

LUDWIG-MAXIMILIANS-UNIVERSITÄT
MÜNCHEN

FORTGESCHRITTENENPRAKTIKUM II
WINTERSEMESTER 22/23

Rheologie

Guido Osterwinter und Jan-Philipp Christ

München, den 10. Dezember 2022

Inhaltsverzeichnis

1	Zielsetzung und Motivation	3
2	Theoretischer Hintergrund	3
2.1	Elastizität und Viskosität	3
2.2	Klassifizierung von Flüssigkeiten anhand ihres Fließverhaltens .	3
2.3	Viskoelastizität	3
2.4	Rotationsrheometer	4
3	Versuchsdurchführung	5
3.1	Wasser-Saccharose	5
3.1.1	Anmischen der Lösungen	5
3.1.2	Scherratenmessungen	5
3.2	Wasser-Guaran	5
3.2.1	Anmischen der Lösungen	5
3.2.2	Scherratenmessungen	6
3.2.3	Frequenzversuch	6
4	Ergebnisse und Diskussion	7
4.1	Wasser-Saccharose	7
4.1.1	Scherratenmessungen	7
4.1.2	Fehlerbetrachtung	9
4.2	Wasser-Guaran	9
4.2.1	Scherratenmessungen	9
4.2.2	Fehlerbetrachtung Scherratenmessungen	15
4.2.3	Frequenzversuch	17
4.2.4	Fehlerbetrachtung Frequenzversuch	18
5	Zusammenfassung	18
A	Ergänzende Plots	21
A.1	Bestimmung der Konzentrationsabhängigkeit der Viskosität bei Saccharose	21
A.2	Bestimmung der Überlappkonzentration für verschiedene Scher- raten bei Guarán	26
B	Python-Skripte zur Auswertung	32
B.1	Bestimmung des Potenzgesetzes für Saccharose	32
B.2	Bestimmung des Potenzgesetzes für Guarán	41
B.3	Bestimmung der Konzentrationsabhängigkeit der Viskosität bei Saccharose	50
B.4	Bestimmung der Konzentrationabhängigkeit der Viskosität bei Guaran	55
B.5	Frequenzversuch	60

1. Zielsetzung und Motivation

Die Rheologie befasst sich mit dem Fließen und der Verformung von Materie, also mit dem Verhalten von Flüssigkeiten und Feststoffen unter dem Einfluss äußerer Kräfte. Die Viskoelastizität ist ein Teilgebiet der Rheologie, das die mechanischen Eigenschaften von Materialien untersucht, die sich wie eine Kombination aus einer viskosen Flüssigkeit und einem elastischen Festkörper verhalten. Viskoelastische Materialien zeigen sowohl viskoses als auch elastisches Verhalten als Reaktion auf Scherung oder andere mechanische Belastungen und sind beispielsweise in medizinischen oder industriellen Anwendungen von Relevanz.

Konkret sollen im Rahmen des hier vorgestellten Versuchs die viskoelastischen Eigenschaften von wässrigen Saccharose-Lösungen (Zuckerwasser) und von wässrigen Guaran-Lösungen verschiedener Konzentrationen untersucht werden. Für die genannten Lösungen werden unter Verwendung eines Rotationsrheometers die Scherrate und der Scherstress in Abhängigkeit der Scherrate vermessen.

Zuletzt wird eine 0.5-prozentige Wasser-Guaran-Lösung einer oszillatorischen Scherverformung, wobei die Oszillationsfrequenz bei fester Amplitude variiert wird, ausgesetzt, um damit die Abhängigkeit der visko-elastischen Materialantwort von der Frequenz zu beleuchten.

2. Theoretischer Hintergrund

2.1. Elastizität und Viskosität

2.2. Klassifizierung von Flüssigkeiten anhand ihres Fließverhaltens

Fließgesetz nach Ostwald und de Waele -1[1]

2.3. Viskoelastizität

Zeigt ein Material sowohl elastisches und viskoses Verhalten, so kann dessen Antwort für hinreichend kleine Scherungen/ Deformationen/ Dehnungen als linear angenommen werden. Dies führt für rein elastische Materialien zum Modell des Hookschen Körpers und für rein viskose Flüssigkeiten zu dem Modell der newtonschen Flüssigkeiten, die als Netzwerk von kleinen, bei Auslenkung Energie dissipierenden, Kolben beschrieben werden können. Das Maxwell-Modell kombiniert diese beiden Modelle, indem Materialien sowohl viskosen als auch elastischen Eigenschaften als Netzwerke von in Serie geschalteten Federn und Kolben interpretiert werden.

Legt man an solche Körper eine oszillatorische Verformung

$$\gamma(t) = \text{Re} \left(\gamma_0 e^{i\omega t} \right) \quad (1)$$

an, so teilt sich die Stressantwort des Materials nach dem Maxwell-Modell in einen Term proportional zur Scherung und in einen Term proportional zur Scherrate auf, sodass

$$\sigma(t) = \text{Re} (G\gamma(t) + \eta\dot{\gamma}(t)) = \text{Re} (G\gamma_0 + \gamma_0\eta \cdot i\omega e^{i\omega t}) =: \text{Re} (\sigma_0 e^{i\omega t + \delta}) \quad (2)$$

$$= \frac{\sigma_0}{\gamma_0} \text{Re} (\gamma(t)(\cos \delta + i \cdot \sin \delta)) =: \text{Re} ((G' + iG'') \cdot \gamma(t)) =: \text{Re} (G^* \cdot \gamma(t)) \quad (3)$$

gilt mit dem Verlustmodul $G'' = \eta\omega$ und dem Speichermodul $G' = G$. Die Phase von G^* definiert den Verlustfaktor $\tan \delta = G''/G'$.

Das Verlust- und das Speichermodul sind beide i.A. abhängig von der Art der Verformung. Legt man beispielsweise eine oszillatorische Verschiebung variabler Frequenz mit fester Amplitude an, so ergibt sich für Polymere in Wasserlösung typischerweise das folgende viskoelastische Spektrum:

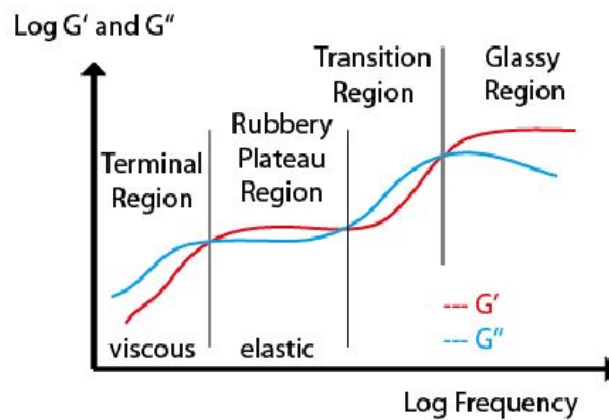


Abbildung 1 Viskoelastisches Spektrum einer Polymerlösung (aus [2])

Für kleine Frequenzen ist $G'' > G'$. In diesem Bereich hat der Stoff den Charakter einer viskoelastischen Flüssigkeit. Nachdem sich Speicher- und Verlustmodul gekreuzt haben, sind über einen gewissen Frequenzbereich (die Rubbery Plateau Region) die viskoelastischen Stoffeigenschaften weitestgehend unabhängig von der Frequenz, und der Stoff verhält sich gelartig oder wie ein viskoelastischer Festkörper. In der sich daran anschließenden Übergangsregion dominieren die viskosen Eigenschaften des Körpers. Für höhere Frequenzen wird die Glassy Region erreicht, in der das Speichermodul plateauartig wird, wohingegen das Verlustmodul abnimmt. Der Stoff verhält sich also zunehmend wie ein (elastischer) Festkörper.

2.4. Rotationsrheometer

Im Versuch wurde das Rotationsrheometer KINEXUS ultra+ von Malvern Panalytical verwendet. Die Probe wird zwischen zwei runden Metallplatten aufgetragen, von denen die obere drehbar gelagert ist und mit variabler Winkelgeschwindigkeit Ω rotieren kann.

Der Plattenabstand ist variabel, wird bei den Messungen hier fest auf $d = 1$ mm gesetzt. Bei einer viskosen Probe wird durch das Rotieren eine Stressantwort $\sigma \propto \eta$ resultieren, welche proportional zum Drehmoment M ist, welches auf die obere Platte wirkt. Dieses Drehmoment wird mit einer Auflösung von 0.05 nNm gemessen (vgl. [3]), was bei Kenntnis der Normalkraft, die auf die Platte wirkt, und einiger geometrieabhängiger Kenngrößen, das Berechnen der Stressantwort und damit der Viskosität zulässt. Wegen $\eta \propto M$ ist $\frac{\Delta\eta}{\eta} = \frac{\Delta M}{M}$ und damit der relative Fehler bei der Viskositätsbestimmung für weniger viskose Flüssigkeiten größer.

3. Versuchsdurchführung

3.1. Wasser-Saccharose

3.1.1. Anmischen der Lösungen

3.1.2. Scherratenmessungen

3.2. Wasser-Guaran

3.2.1. Anmischen der Lösungen

Prinzipiell gleicht das Vorgehen beim Anmischen und Auftragen der Lösungen auf das Rheometer dem in Abschnitt 3.1.1. Da Guaran bei Wasserkontakt schnell eine viskose Flüssigkeit bildet, die nur mit großen Schwierigkeiten pipettiert werden kann, wurde die jeweilige Probe unmittelbar vor Messbeginn mit Wasser versetzt. Dabei war auf ein möglichst schnelles Vermischen von Guaran und Wasser zu achten, da Guaran die Tendenz hat, bei Wasserkontakt Klumpen auszubilden, die ein vollständiges Durchmischen der gesamten Pulversubstanz erheblich erschweren. Eine solche Klumpenbildung war bei der 1%igen-Lösung zu beobachten. Durch mehrfaches Pipettieren konnten die Klumpen so weit aufgelöst werden, dass keine mehr mit bloßem Auge zu erkennen waren. Im Prozess bildeten sich in der Flüssigkeit jedoch kleine Luftblasen, die nicht vollständig von der Flüssigkeit getrennt werden konnten. Auf deren Einfluss auf die Messergebnisse wird in Abschnitt 4.2.2 eingegangen.

Eine weitere Schwierigkeit beim Anmischen der Lösungen ist, dass gerade bei höheren Konzentrationen ein nicht unerheblicher Anteil der Substanz wegen der hohen Viskosität nach dem Einziehen in der Pipette verbleibt. In besonderem Maße wurde deshalb beim Loaden des Samples darauf geachtet, dass eine ausreichende Abdeckung des Messstempels am Rheometer erzielt wurde, um Underfilling zu vermeiden. So musste beispielsweise für die 2.3%ige Lösung nachträglich noch Flüssigkeit hinzugefügt werden, da anfangs nicht das gesamte Volumen unter der Geometry-Platte von Flüssigkeit ausgefüllt wurde. Auf diese Weise wurde ein Zustand erreicht, bei dem die Flüssigkeit sich so aus dem Spalt herauskrümmte, dass der Krümmungsradius unterhalb der Geometry lag.

3.2.2. Scherratenmessungen

Die sich anschließende Messung wurde nach Vornehmen der entsprechenden Einstellungen unter Toolkit_V001 Shear Rate Table mit $\dot{\gamma} \in [1, 100]\text{s}^{-1}$, $T = 25^\circ\text{C}$, 5 Messpunkten pro Dekade und dem Parameter Tskip nach 1 min vom Rheometer automatisiert durchgeführt, woraufhin die Messwerte als Tabelle exportiert werden konnten.

Es wurden Konzentrationen $c \in [0, 0.25, 0.5, 1.0, 1.4, 2.0, 2.3]\%$ untersucht, wobei die Messwerte für die reine Wasserlösung mit $c = 0.0\%$ aus der vorangegangenen Saccharose-Messreihe übernommen werden konnten.

3.2.3. Frequenzversuch

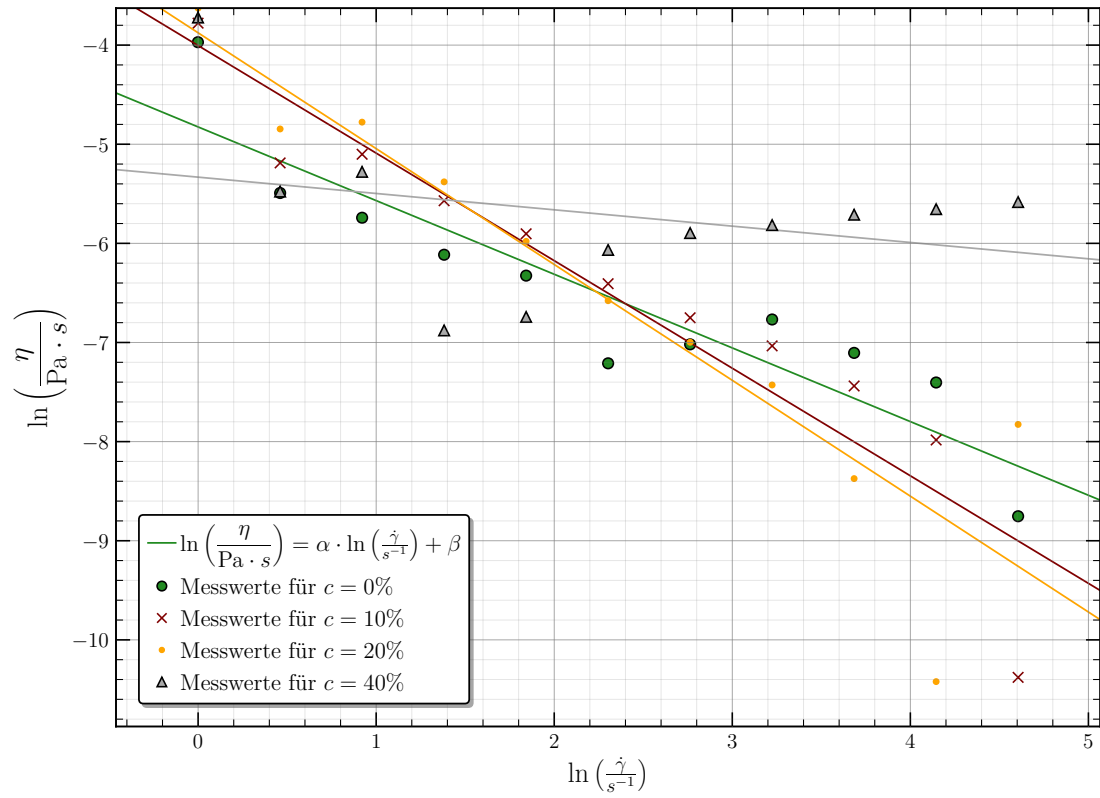
In diesem Versuchsteil sollte eine 0.5%ige Guaran-Lösung einer oszillatorischen Scherverformung ausgesetzt worden. Es wurde dieselbe Probe, mit der auch die entsprechende Messung in Abschnitt 3.2.1 durchgeführt wurde, verwendet. Der eigentliche Messprozess lief automatisiert nach Aufrufen des entsprechenden Programms Toolkit_O002 Frequency Table in der Rheometer-Software und dem Einstellen des Messbereichs von 0.01–200 Hz, der Soll-Temperatur $T = 25^\circ\text{C}$ und der Anzahl (5) von Messpunkten pro Dekade. Tskip wurde auf 1 min gesetzt, die Stärke der Scherverformung betrug 0.5%.

Aufgezeichnet wurden das Speichermodul G' und das Verlustmodul G'' als Funktion der Oszillationsfrequenz bei konstanter Amplitude der Scherverformung.

4. Ergebnisse und Diskussion

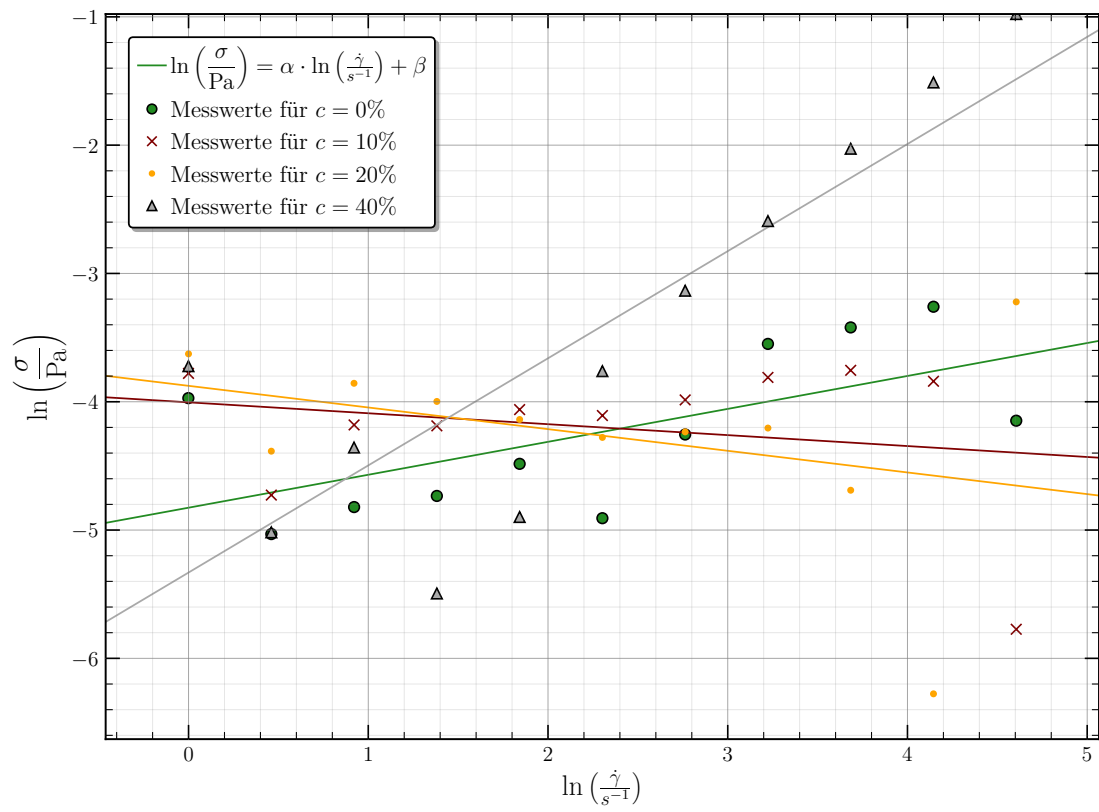
4.1. Wasser-Saccharose

4.1.1. Scherratenmessungen



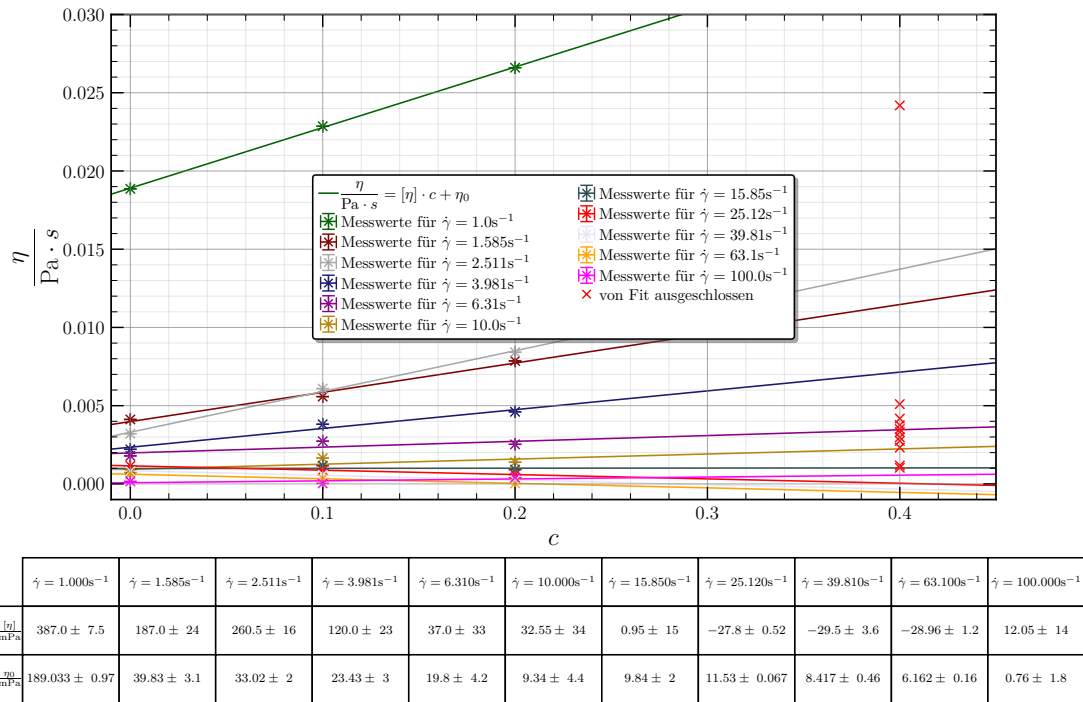
	$c = 0\%$	$c = 10\%$	$c = 20\%$	$c = 40\%$
α	-0.743 ± 0.11	-1.085 ± 0.13	-1.169 ± 0.16	-0.165 ± 0.17
β	-4.825 ± 0.29	-4.005 ± 0.34	-3.876 ± 0.43	-5.332 ± 0.47

Plot 1 Auftragung von η gegen $\dot{\gamma}$ bei Saccharose



	$c = 0\%$	$c = 10\%$	$c = 20\%$	$c = 40\%$
α	0.257 ± 0.11	-0.085 ± 0.13	-0.169 ± 0.16	0.835 ± 0.17
β	-4.826 ± 0.29	-4.004 ± 0.34	-3.876 ± 0.43	-5.331 ± 0.47

Plot 2 Auftragung von σ gegen $\dot{\gamma}$ bei Saccharose



Plot 3 Auftragung von η gegen c bei Saccharose

4.1.2. Fehlerbetrachtung

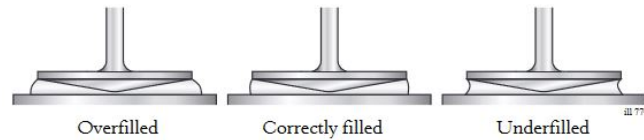
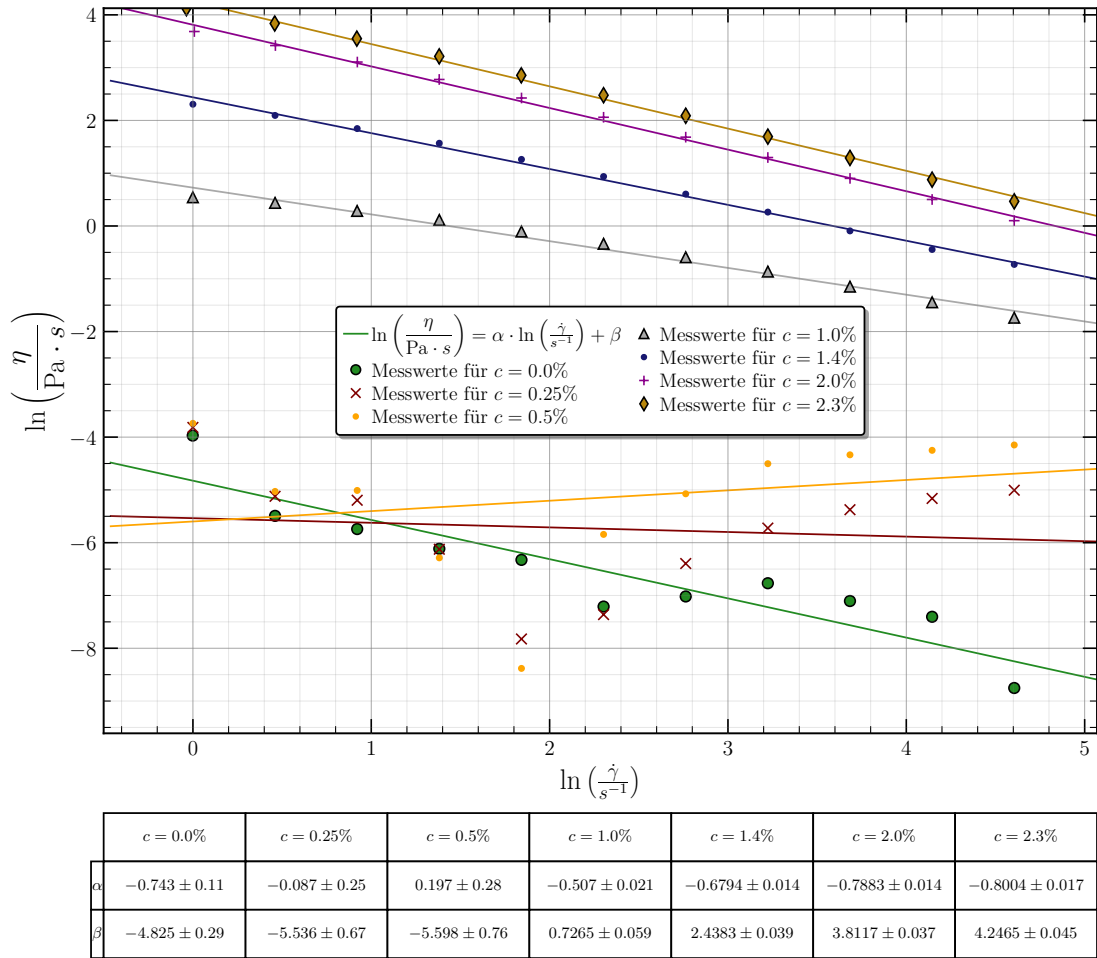


Abbildung 2 Zur korrekten Platzierung des Samples (aus [2])

4.2. Wasser-Guaran

4.2.1. Scherratenmessungen

Trägt man die für die verschiedenen Konzentrationen und Scherraten aufgenommenen Messwerte gegeneinander auf, so können die Werte $(\eta_i, \dot{\gamma}_i)$ für feste Konzentration c nach dem Fließgesetz von Ostwald und de Waele (vgl. [1]) bei einem doppelt-logarithmischen Plot durch Geraden beschrieben werden.



Plot 4 Doppelt logarithmische Auftragung von η gegen $\dot{\gamma}$ bei Guaran

Rein qualitativ stellt man durch Inspektion von Plot 4 fest, dass die Messwerte für $c \in [1.0, 1.4, 2.0, 2.3]\%$ gut durch eine Gerade beschrieben werden können, was die Gültigkeit des Fließgesetzes von Ostwald und de Waele stützt und sich in einer geringen Standardabweichung der Fitparameter α („Steigung“) und β („Verschiebung entlang der Ordinate“) niederschlägt. In guter Näherung parallel zu den vier genannten Geraden verläuft die Fitgerade, die die Messwerte für die reine Wasserlösung beschreiben soll. Es fällt die große Streuung der Messwerte auf, was auf die Bemerkung in 2.4, wonach der relative Fehler in der Viskosität bei weniger viskosen Flüssigkeiten größer sein sollte, zurückzuführen ist. Durch den größeren Fehler bei jedem Messwert wird die statistische Streuung um die Fitgerade und damit auch die Unsicherheit der Fitparameter α, β größer.

Unerwartet hingegen ist das Verhalten, das sich für $c = 0.25\%$ und $c = 0.5\%$ zeigt. Für die 0.5%ige Lösung legt der Fit gar ein $\alpha > 0$ nahe, was bedeuten würde, dass die untersuchte Lösung *scherverdickendes* Verhalten zeigen würde. Dies steht im Widerspruch dazu,

dass Guaran nach [4] stark *scherverdünnende* Eigenschaften hat. Tatsächlich ist aber ein lineares Modell höchst unzureichend zur Beschreibung der betreffenden Daten. Nur für die letzten 4-5 Messwerte (bei hohen Scherraten) lässt sich ein linearer Zusammenhang (aber mit positiver Steigung!) erahnen. Für kleinere Scherraten sieht es vielmehr so aus, als ob die Daten zwei nach unten geöffneten Hyperbeln folgen würden, die sich etwa bei $\ln\left(\frac{\dot{\gamma}}{s-1}\right) \approx 1.8$ treffen. Entsprechend schwach in der Aussagekraft sind die jeweiligen Fitparameter α , die je mit einer mehr als hundertprozentigen Unsicherheit behaftet sind. Insbesondere kann also anhand von Plot 4 nicht darauf geschlossen werden, dass Guar für manche Konzentrationen scherverdickendes Verhalten zeigt. Da der Mechanismus, der bestimmt, ob ein Stoff scherverdünnende oder -dickende Eigenschaften hat, prinzipiell für alle Konzentrationen derselbe ist (bei Polymeren wie Guaran: „Verhaken“ der langkettigen Moleküle in der Lösung), muss auch eine 0.5%ige Guar-Wasser-Lösung eindeutig scherverdünnende Eigenschaften haben und das erläuterte Verhalten bei $c = 0.25\%, 0.5\%$ auf systematische Messfehler zurückzuführen sein (siehe dazu Abschnitt 4.2.2).

Weiterhin muss angemerkt werden, dass Wasser als Standardbeispiel für ein Newtonsches Fluid eigentlich eine scherratenunabhängige Viskosität haben müsste.

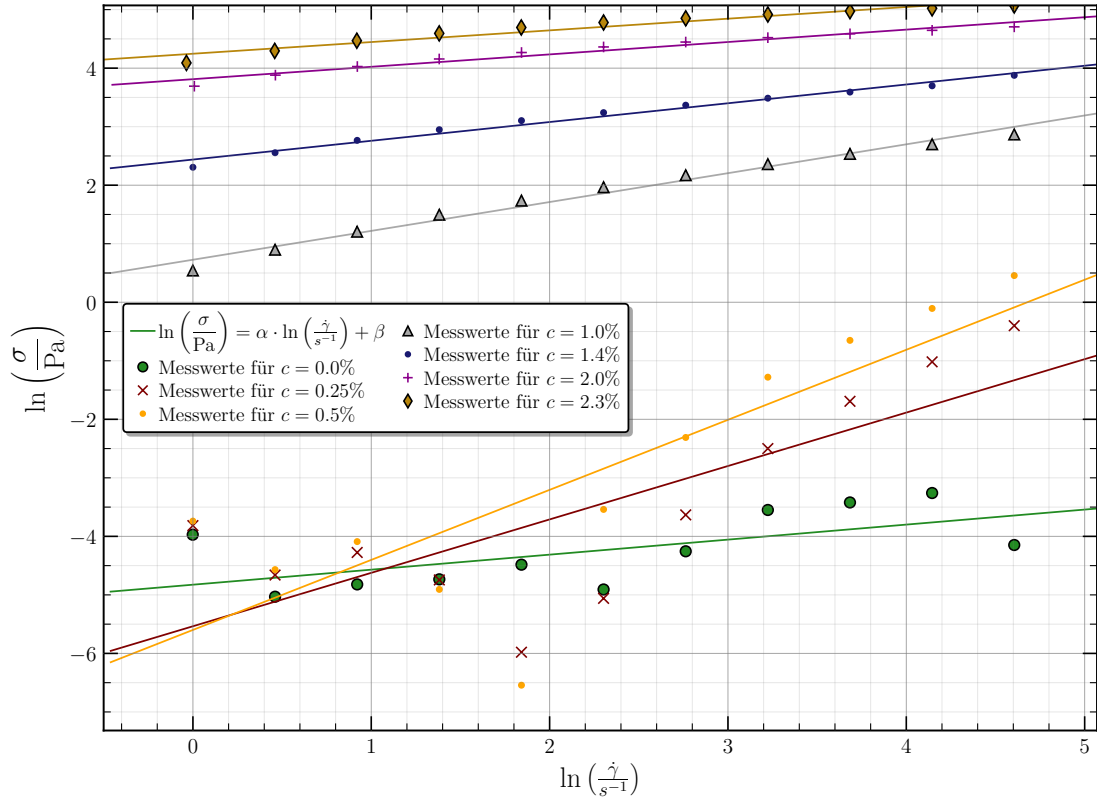
Erklärungsansätze, warum dennoch scherverdünnendes Verhalten beobachtet wurde, wurden bereits in Abschnitt 4.1.2 gefunden.

Für $c = [1.0, 1.4, 2.0, 2.3]\%$ entspricht das beobachtete Verhalten der Erwartung, dass Guaran in Wasserlösung ein scherverdünnendes Fluid ist. Die Potenzabhängigkeit ist also von der Form

$$\eta \propto \dot{\gamma}^\alpha \quad (4)$$

mit $c \in [-0.51 \pm 0.03, -0.68 \pm 0.02, -0.79 \pm 0.02, -0.80 \pm 0.02]$. Die Güte und Aussagekraft dieser Werte ist hoch; lediglich eine Verschiebung dieser Werte durch systematische Fehlereinflüsse ist denkbar. Diese werden in Abschnitt 4.2.2 eingeschätzt. Für die drei niedrigsten Konzentrationen ist aus den oben erläuterten Gründen ein abschließendes Festhalten des Fitparameters α zur Bestimmung der Potenzgesetz-Beziehung nicht sinnvoll.

Zuletzt kann der Scherstress gegen die Scherrate aufgetragen werden. Plot 5 enthält aber im Vergleich zu Plot 0 keinerlei neue Information, da die vom Rheometer ausgegebenen Messwerte für σ und η beide aus dem gemessenen Drehmoment berechnet wurden und zwischen ihnen die Abhängigkeit $\sigma = \dot{\gamma} \cdot \eta$ gilt.



	$c = 0.0\%$	$c = 0.25\%$	$c = 0.5\%$	$c = 1.0\%$	$c = 1.4\%$	$c = 2.0\%$	$c = 2.3\%$
α	0.257 ± 0.11	0.913 ± 0.25	1.197 ± 0.28	0.493 ± 0.021	0.3206 ± 0.014	0.2117 ± 0.014	0.1996 ± 0.017
β	-4.826 ± 0.29	-5.536 ± 0.67	-5.599 ± 0.76	0.7265 ± 0.059	2.4383 ± 0.039	3.8115 ± 0.037	4.2465 ± 0.045

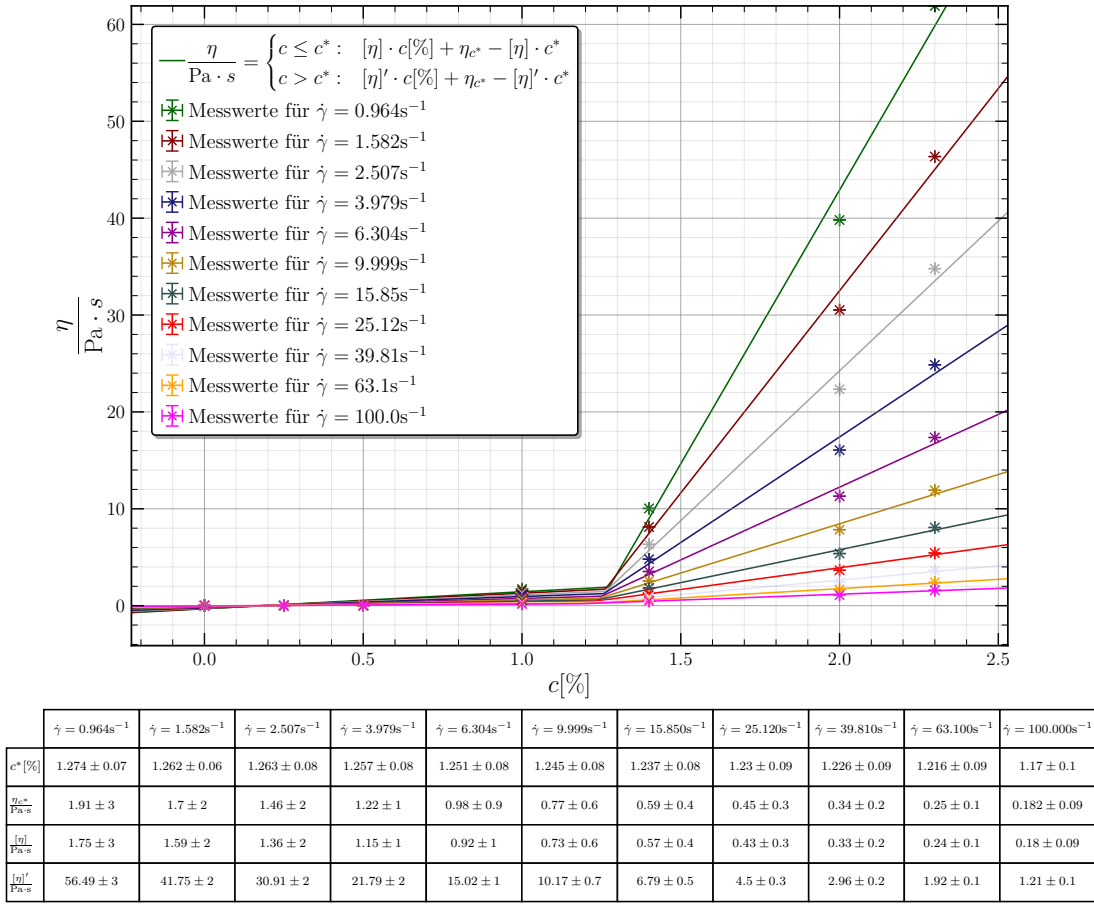
Plot 5 Doppelt logarithmische Auftragung von σ gegen $\dot{\gamma}$ bei Guaran

Entsprechend dieser Abhängigkeit sind die für die Parameter α_σ und α_η für die aus Plot 5 bzw. Plot 4 gewonnenen Parameter numerisch exakt $\alpha_\sigma - \alpha_\eta = 1$. Da $0 < \alpha_\sigma < 1$ kann erneut bestätigt werden, dass scherverdünnendes Verhalten vorliegt.

Schon rein visuell fällt in Plot 4 auf, dass für die Viskositäten zwischen den Konzentrationen $c = 1.0\%$ und $c = 1.4\%$ gemessen an der kleinen Konzentrationsdifferenz ein recht großer Sprung in der Viskosität zu beobachten ist. Und tatsächlich modelliert man die Abhängigkeit der Viskosität von der Konzentration bei Polymerlösungen häufig folgt:

$$\eta(c) = \begin{cases} c \leq c^* : & [\eta] \cdot (c - c^*) + \eta_{c^*} \\ c > c^* : & [\eta]' \cdot (c - c^*) + \eta_{c^*} \end{cases} \quad (5)$$

Für eine feste Scherrate kann somit durch Variation der Parameter $([\eta], [\eta]', c^*, \eta_{c^*})$ ein Fit der Daten vorgenommen werden. Es ergibt sich der folgende Plot:



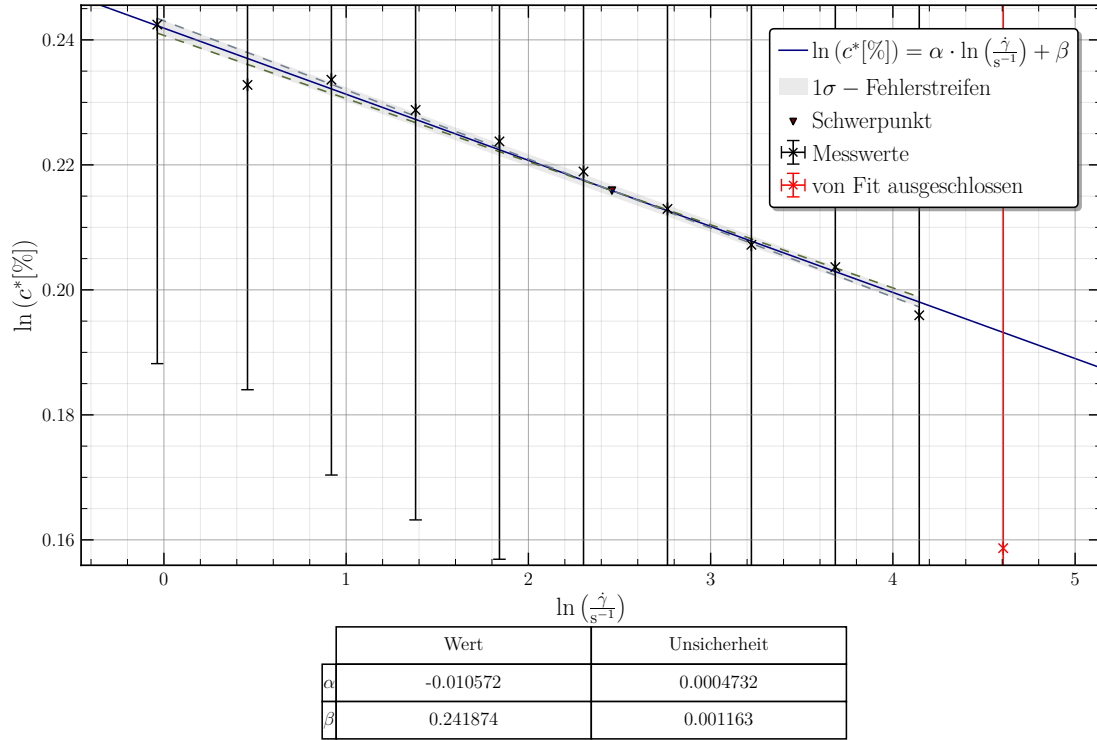
Plot 6 Auftragung von η gegen c bei Guaran

In der Appendix A.2 sind für jede Scherrate dieselben Daten nochmals in einzelnen Figures aufgetragen, um die Übersichtlichkeit zu verbessern. Qualitativ kann festgehalten werden, dass das verwendete Modell die Messdaten gut zu beschreiben vermag, also tatsächlich eine Konzentration c^* existiert, ab der der Anstieg der Viskosität mit der Konzentration schneller wird.

Die Güte der Fits mag überraschend sein, da bei der Betrachtung der Viskosität als Funktion der Scherrate zumindest für zwei Konzentrationen äußerst unzufriedenstellende Messwerte aufgenommen wurden. Da aber nun $\dot{\gamma}$ festgehalten wird, haben wir bei jedem der 11 Fits in Plot 6 mindestens vier Messwerte mit guter Aussagekraft, sodass diese den Einfluss der beobachteten Abweichung der zwei bis drei anderen Messwerte abschwächen und zu einer relativ hohen Aussagekraft der Fitparameter führt.

Ein Blick auf die Fitparameter zeigt, dass c^* monoton mit der Scherrate $\dot{\gamma}$ abzunehmen scheint. Tatsächlich zeigt sich in einer doppelt logarithmischen Darstellung der kritischen Konzentration gegen die Scherrate ein klarer linearer Trend (\rightarrow Plot 7). Beim Wert zu $\dot{\gamma} = 100\text{ s}^{-1}$ handelt es sich um einen statistischen Ausreißer, der deshalb

nicht bei der Regression berücksichtigt wurde. Ein Berücksichtigen dieses Messwertes hätte zu einer Verschlechterung in der Beschreibung des ansonsten eindeutig linearen Zusammenhangs geführt.



Plot 7 Doppelt logarithmische Auftragung von c^* gegen $\dot{\gamma}$ bei Guaran

In der Tat erscheint es plausibel, dass ein Potenzgesetz zwischen der kritischen Konzentration und der Scherrate vorliegt: Die Viskosität nimmt gemäß eines Potenzgesetzes mit der Scherrate ab. Das heißt auch, dass die Kurven in Plot 6 für festes c gemäß

$$\frac{\dot{\gamma}_1}{\dot{\gamma}_2} = \frac{\eta_1^{\alpha(c)}}{\eta_2^{\alpha(c)}} \quad (6)$$

ineinander überführt werden können. Stellt man sich nun eine Kurve (η, c) für festes $\dot{\gamma} = \dot{\gamma}_2$ vor und möchte diese in eine Kurve mit $\dot{\gamma} = \dot{\gamma}_1 > \dot{\gamma}_2$ überführen, so gilt $\eta_1 = \left(\frac{\dot{\gamma}_1}{\dot{\gamma}_2}\right)^{1/\alpha(c)} \cdot \eta_2$. Nun kann man Plot 4 entnehmen, dass $|\alpha|$ für größere Konzentrationen zunimmt. Das bedeutet auch, dass die η -Werte für kleinere c (relativ gesehen) weniger gestaucht werden als die Werte mit größere c . Damit flacht die Kurve $\eta(c)|_{\dot{\gamma}_1}$ im Vergleich zur Kurve $\eta(c)|_{\dot{\gamma}_2}$ ab, und insbesondere wird der Punkt der kritischen Konzentration nach links verschoben (zu kleineren c). Für die konkrete Form des Potenzgesetzes, wie c^* von $\dot{\gamma}$ abhängt, kommt es nun auf die konkrete Form der Abhängigkeit $\alpha(c)$ an. Klar ist aber, dass zwischen c^* und $\dot{\gamma}$ ein Potenzzusammenhang gelten muss.

Da also, wie gerade dargelegt, die kritische Konzentration nicht nur eine Stoffeigenschaft ist, sondern auch von der Scherrate abhängt, ist es nicht sinnvoll, abschließend *eine* kritische Konzentration für Guaran-Wasser-Lösungen festzuhalten. Stattdessen bemerken wir also, dass der folgende funktionale Zusammenhang empirisch gefunden wurde:

$$c^*[\%] = e^{0.242 \pm 0.002} \cdot \dot{\gamma}^{-0.0106 \pm 0.0005} \quad (7)$$

4.2.2. Fehlerbetrachtung Scherratenmessungen

Wie in im vorigen Abschnitt 4.2.1 bereits angemerkt, muss das beobachtete Verhalten bei $c = 0.25\%$, 0.5% auf systematische Messfehler zurückzuführen sein. Es sind folgende Fehlereinflüsse denkbar:

- Under- oder Overfilling: In besonderem Maße wurde bei dem Loaden des Samples bei den Guar-Messungen darauf geachtet, dass eine ausreichende Lösungsmenge aufgebracht wird, da stets ein Teil der mittels einer Pipette abgemessenen Menge in der Pipette zurückblieb. Hellström (2015) merkt an: „*Small changes in the radius of a rheometer sample can cause significant errors in the measured apparent viscosity*“ ([5]). Er empfiehlt ein Verfahren, bei dem mittels Bildgebung der reale Sample-Radius bestimmt wird, um auf diesen Fehler hin zu korrigieren. Dies ist auch bei dem hier verwendeten Setup leicht umzusetzen und verspricht erhebliche Verbesserungen in der Messgenauigkeit.
- Nicht-Konstanz der Sample-Temperatur während der Messung (vgl. dazu auch Abschnitt 4.1.2): Kleine Variationen in der Temperatur führen zu vernachlässigbaren Messfehlern. Bei den Saccharose-Messungen wurde jedoch schon die Beobachtung gemacht, dass - entgegen der Erwartung- die Saccharose-Lösungen und sogar die reine Wasserlösung scherverdünnendes Verhalten zeigen. Da die Temperatur nicht direkt am Sample gemessen wird (sonder in der Platte darunter), und da die Anfangstemperatur des Samples etwas 12 K unter der Soll-Temperatur während des Versuchs lag, bleibt die Frage offen, ob das Sample bei der ersten Scherrate für die bereits die Soll-Temperatur erreicht hat. Würde sich die Temperatur des Samples im Verlauf der weiteren Messungen weiter erhöhen, würde ein verstärkt scherverdünnendes Verhalten beobachtet werden, da höhere Temperaturen zu einer Abnahme der Viskosität führen.
- (kleine) Luftblaseneinschlüsse im Sample, der beim Anmischen der Lösungen entstanden sein könnte: Stadler spekuliert, dass die wegen ihrer Oberflächenspannung sehr elastischen Luftblasen bei Elastizitätsmessungen deformiert werden und zu einer erhöhten Elastizitätsmessung führen ([6]). Analoge Überlegungen könnten nahelegen, dass Luftblaseneinschlüsse zu einer geringeren Viskosität führen, da die Luftblasen lokal des Zusammenhalt der Polymermoleküle schwächen würden. Denkbar ist auch, dass sich bei einer gewissen Scherrate zwei eingeschlossene Luftblasen vereinigen, da sie sich durch die Deformation näher kommen. Ein solches Vereinigen könnte zu einem unstetigen Verhalten bei der Viskositätsmessung führen. Die

Bewegung von Luftblasen in Flüssigkeiten ist hydrodynamisch jedoch sehr komplex, sodass der konkrete Einfluss schwer abzuschätzen ist. Klar ist jedoch, dass Luftblaseneinschlüsse im Sample als wahrscheinlich eingeschätzt werden müssen, da sie beim Schütteln des Samples oder dem Pipettieren entstanden sein könnten.

- nicht vollständig gelöstes Guaran-Pulver, das ausklumpt: Solche Klumpen würden eine Inhomogenität in der Lösung bedeuten, die lokal die Viskosität (stark) erhöht. Insgesamt wäre die gelöste Pulvermenge durch solche Klumpen verringert, da das Pulver innerhalb der Klumpen natürlich nicht gelöst werden kann. Das Auftreten solcher Klumpen steht außer Frage, jedoch wurden sie stets so weit ausgemerzt, bis keine mehr mit bloßem Auge sichtbar waren. Bei der 1%igen Lösung sind solche Einflüsse wegen Schwierigkeiten beim Anmischen wahrscheinlich. Dort ist jedoch im Gegensatz zu den Lösungen mit $c = 0.25\%, 0.5\%$ ein klar linearer Zusammenhang erkennbar, weshalb Klumpenbildung vermutlich nicht alleine das Verhalten bei den besagten zwei Konzentrationen erklären kann.
- Erschütterungen im Raum während der Versuchsdurchführung: Einflüsse dadurch sind wegen der hohen Genauigkeit bei der Drehmomentbestimmung prinzipiell möglich, beispielsweise wenn die Tür rapide zugezogen werden würde. Außern würde es sich durch einzelne Messausreißer. Für die Messungen mit $c = 0.25\%, 0.5\%$ kann ein solcher Einfluss nicht als Erklärung herangezogen werden, da die beiden Messungen in dem Sinn korreliert erscheinen, dass sie das gleiche qualitative Verhalten (aus zwei nach unten gerichteten Hyperbeln zu bestehen) zeigen.
- Falschausrichtung der Geometry (vgl. dazu [6]): Wurde die Geometry in der Vergangenheit einmal fallen gelassen oder hat sie sich anderweitig deformiert, ist es denkbar, dass die beiden Platten nicht mehr parallel sind. Dies würde den Plattenabstand (Gap) verändern, was die ausgegebenen Scherraten wegen $\dot{\gamma} \propto d^{-1}$ systematisch verfälschen würde. Da das verwendete Rheometer jedoch häufig in Benutzung ist, ist es wahrscheinlich, dass solche Veränderungen sehr frühzeitig bemerkt werden würden und deshalb der Einfluss von dadurch induzierten Fehlern als unwahrscheinlich einzuschätzen.

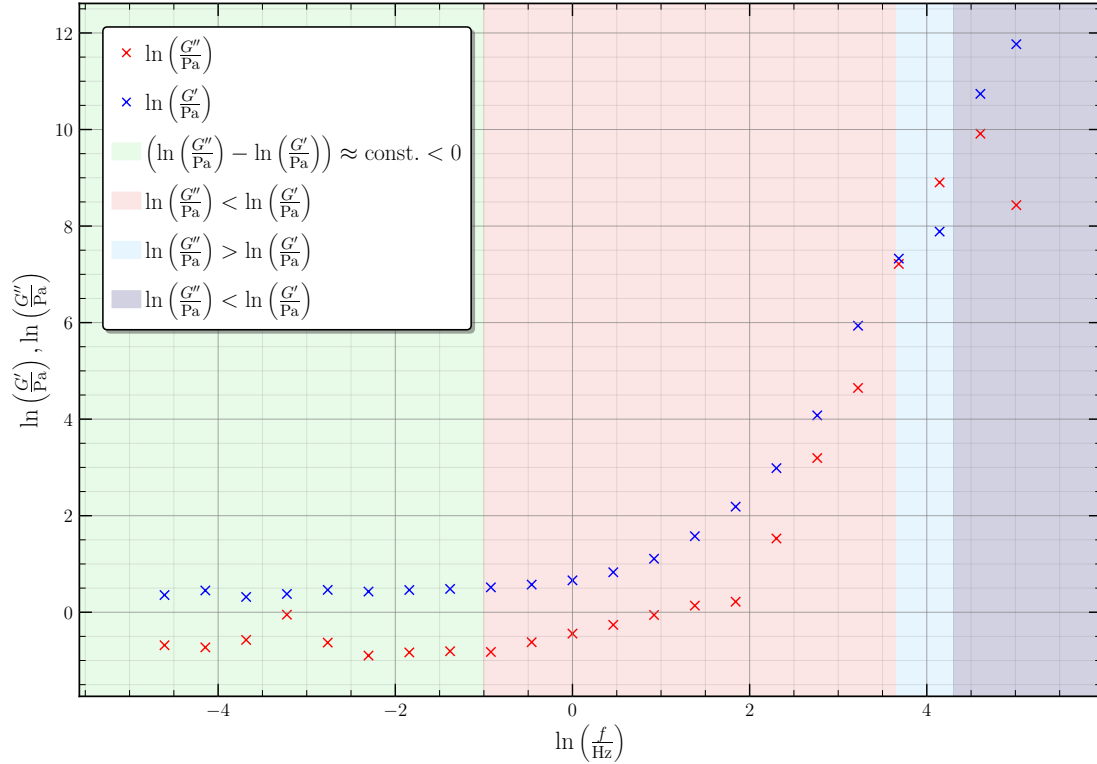
Außerdem gilt weiterhin die Bemerkung aus Abschnitt 2.4, dass prinzipiell der relative Fehler bei der Viskositätsbestimmung für kleiner Viskositäten und somit kleinere Stoffkonzentrationen größer ist. Dies Zusammen mit der Nicht-Konstanz der Temperatur und der Gefahr von Under- oder Overfilling sind als die wahrscheinlichsten Fehlereinflüsse zu nennen. Zuletzt sei folgende Bemerkung von Stadler ([6]) erwähnt, wonach ein erheblicher Anteil der beobachteten Anomalitäten auf falsche Anwendung seitens des Benutzers zurückzuführen sind:

„many inexperienced users obtain data, which is significantly error afflicted in nontrivial ways“

Denkbar hier sind das unzureichende Reinigen der Platten am Rheometer vor dem Loaden des Samples oder sonstige Flüchtigkeitsfehler.

4.2.3. Frequenzversuch

Trägt man das Speicher- bzw. Verlustmodul doppellogarithmisch gegen die Oszillationsfrequenz auf, erhält man den folgenden Plot:



Plot 8 Doppelt logarithmische Auftragung von G' , G'' gegen die Frequenz f bei Guaran

Es zeigt sich, dass für kleine Frequenzen sowohl das Speicher- als auch das Verlustmodul in sehr guter Näherung konstant sind, wobei das Speichermodul größer ist als das Verlustmodul (grün hinterlegt). In Abbildung 1 kann dies der Rubbery Plateau Region zugeordnet werden, in der sich der Stoff gelartig verhält.

Im sich anschließenden rot hinterlegten Bereich nehmen die viskosen Anteile weiter zu, bis sich das Speicher- und das Verlustmodul bei $f^* \approx 40$ Hz kreuzen. Für einen kurzen Frequenzbereich (hellblau hinterlegt) dominiert das Verlustmodul gegenüber dem Speichermodul. Dieser Bereich kann der Transition Region in Abbildung 1 zugeordnet werden. In diesem Bereich wäre eine höhere Messwertdichte wünschenswert, um eine definitivere Aussage darüber treffen zu können, in welchem Bereich $G'' > G'$ ist. Anschließend nimmt das Speichermodul weiter zu und das Verlustmodul ab, sodass $G'' > G'$ (dunkelblau hinterlegt). Gemäß Abbildung 1 ist für noch höhere Frequenzen ein Abflachen von G' und Abnehmen von G'' zu erwarten. Es wäre schön, noch weitere Messwerte zu sehen, um das Verhalten, das vermutlich der Glossy Region zugeordnet werden kann, weiter zu beleuchten.

Ferner konnte für keinen Frequenzbereich ein Verhalten wie in der Terminal Region beobachtet werden. Es ist möglich, dass für kleinere Frequenzen (< 0.01 Hz) als im betrachteten Messbereich ein solches auftreten würde.

Insgesamt konnten also drei der vier Regionen, die im viskoelastischen Spektrum einer Polymerlösung zu erwarten sind, bei der Analyse der 0.5%igen Guaran-Wasser-Lösung beobachtet werden.

4.2.4. Fehlerbetrachtung Frequenzversuch

Da die Beobachtungen, die zum Frequenzversuch gemacht wurden, nur qualitativer Natur sind, ist der Einfluss von systematischen Fehlern als äußerst gering einzustufen. Eine von der Soll-Temperatur verschiedene Temperatur würde die Breite und Höhe der einzelnen viskoelastischen Bereiche verändern, aber nichts an deren qualitativen Natur ändern. Selbst eine sich zeitlich (monoton) ändernde Temperatur des Samples, wie in Abschnitt 4.1.2 diskutiert, würde den qualitativen Verlauf kaum ändern, sondern nur zu einer Stauchung entlang der Abszisse und auch Ordinate führen und damit zwar u.U. das Bestimmen von f^* systematisch verfälschen, aber an der prinzipiellen Abfolge der viskoelastischen Bereiche nichts ändern. Der Einfluss durch Temperaturvariation ist jedoch (im Gegensatz zu den Messungen mit Saccharose) ohnehin als unerheblich einzustufen, da die 0.5%ige Lösung wegen der zuvor mit ihr vorgenommenen Messungen schon lange vor dem Beginn der eigentlichen Oszillationsmessung auf das Rheometer gegeben wurde, also auf jeden Fall genug Zeit hatte, die Temperatur Soll-Temperatur von 25°C anzunehmen.

Auch der Einfluss eines Underfills wäre nur gering, da dadurch die elastischen und viskosen Eigenschaften gleichermaßen als zu gering gemessen worden wären, der Graph dadurch also nur entlang der Ordinate gestaucht werden würde.

Der größte Fehler bei der Bestimmung der einzelnen viskoelastischen Bereiche ist die recht grobe Auflösung in der Frequenz, da dadurch beispielsweise die Schnittpunkte vom Verlust- und vom Speichermodul nur äußerst grob abgeschätzt werden konnten. Es wäre sinnvoll, die Messpunkte pro Dekade bei einer Wiederholung des Versuchs zu Kosten einer längeren Versuchsdurchführung auf einen Wert > 5 zu setzen. Auch ist es empfehlenswert, den Messbereich selbst zu vergrößern. Bei einigen hundert Hz maximaler Messfrequenz mehr könnte die Glossy Region deutlich detaillierter beleuchtet werden. Änderte man zudem die minimale Messfrequenz, ist es denkbar, dass die Terminal Region noch zu beobachten wäre.

[7] [6] [5] [8]

5. Zusammenfassung

Es gelingt uns nicht, eine zufriedenstellende Erklärung für die Messwerte bei den Konzentrationen $c = 0.25\%$, 0.5% zu finden. Ein Wiederholen dieser Messungen wird ausdrücklich empfohlen, um zu prüfen, ob das beobachtete Verhalten reproduzierbar ist. Zu erwarten ist jedoch, dass sich bei einer Wiederholung „schönere“ Werte ergeben würden, die erstens stützen, dass auch bei diesen Konzentrationen zwischen der Viskosität und

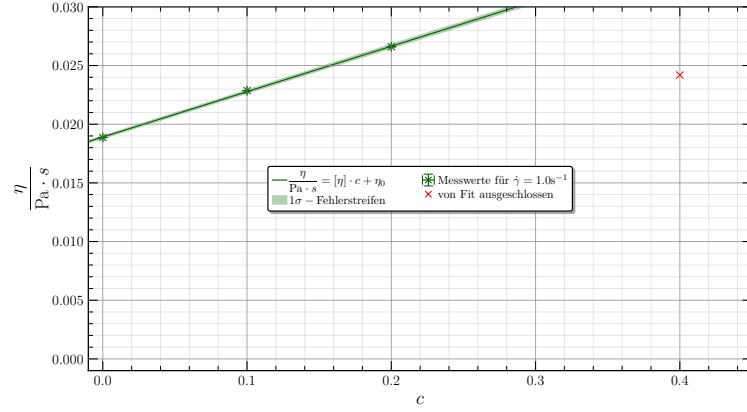
der Scherrate ein Potenzgesetzzusammenhang gilt, und zweitens, das Guaran-Wasser-Lösungen für beliebige Konzentrationen scherverdünnend sind. Weiterhin empfehlen wir eine Versuchsdurchführung in einem wärmeren Raum. Wahrscheinlich würde dann kein scherverdünnendes Verhalten bei den Saccharose-Lösungen beobachtet werden.

Literatur

- [1] Wikipedia, “Potenzgesetz (Flüssigkeit).” [https://de.wikipedia.org/w/index.php?title=Potenzgesetz_\(Fl%C3%BCssigkeit\)&oldid=192899581](https://de.wikipedia.org/w/index.php?title=Potenzgesetz_(Fl%C3%BCssigkeit)&oldid=192899581), 2019. [Online; Stand 3. Dezember 2022].
- [2] Malvern, “Kinexus series user manual,” Jan 2014. [Online unter <https://www.equipx.net/uploads/Malvern%20Instruments/MalvernKinexus-usermanual.pdf>, Stand 08.12.2022].
- [3] Malvern, “Kinexus Brochure,” 2017. [Online unter https://www.malvernpanalytical.com/de/assets/MRK1089-06-DE_Kinexus_Brochure_LRA4.tcm57-17219.pdf, Stand 08.12.2022].
- [4] Wikipedia contributors, “Guar gum.” https://en.wikipedia.org/w/index.php?title=Guar_gum&oldid=1124053245, 2022. [Online; Zugriff am 08. Dezember 2022].
- [5] L. H. O. Hellström, M. A. Samaha, K. M. Wang, A. J. Smits, and M. Hultmark, “Errors in parallel-plate and cone-plate rheometer measurements due to sample underfill,” *Measurement Science and Technology*, vol. 26, p. 015301, nov 2014.
- [6] F. J. Stadler, “What are typical sources of error in rotational rheometry of polymer melts?,” *Korea-Australia Rheology Journal*, vol. 26, pp. 277–291, Aug 2014.
- [7] Autor unbekannt, “Praktikumsversuch Rheologie Bachelor.” [Online unter <https://www.softmatter.physik.uni-muenchen.de/teaching/fortgeschrittenenpraktikum/r3-rheologie/fpraktikumrheologiebdeutsch.pdf>; Stand 03. Dezember 2022].
- [8] E. Arian and W. Pauer, “Sucrose solution as a new viscous test fluid with tunable viscosities up to 2 pas for micromixing characterization by the villermaux–dushman reaction,” *Journal of Flow Chemistry*, vol. 11, pp. 579–588, Sep 2021.

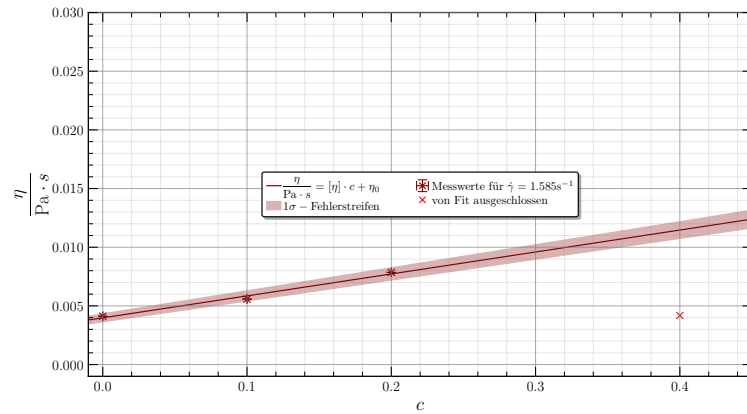
A. Ergänzende Plots

A.1. Bestimmung der Konzentrationsabhängigkeit der Viskosität bei Saccharose



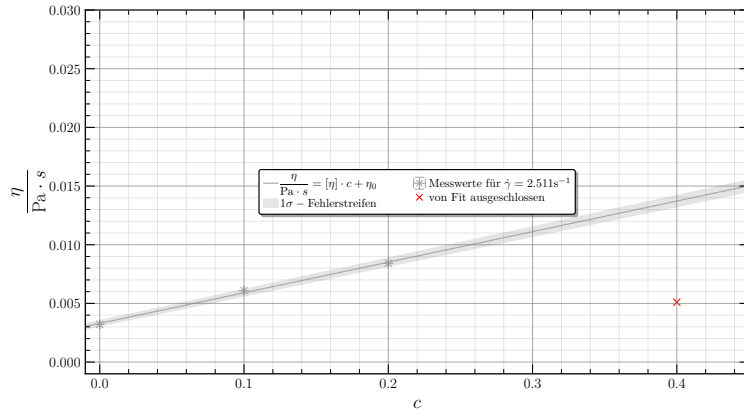
	$\dot{\gamma} = 1.000\text{s}^{-1}$	$\dot{\gamma} = 1.585\text{s}^{-1}$	$\dot{\gamma} = 2.511\text{s}^{-1}$	$\dot{\gamma} = 3.981\text{s}^{-1}$	$\dot{\gamma} = 6.310\text{s}^{-1}$	$\dot{\gamma} = 10.000\text{s}^{-1}$	$\dot{\gamma} = 15.850\text{s}^{-1}$	$\dot{\gamma} = 25.120\text{s}^{-1}$	$\dot{\gamma} = 39.810\text{s}^{-1}$	$\dot{\gamma} = 63.100\text{s}^{-1}$	$\dot{\gamma} = 100.000\text{s}^{-1}$
$\frac{ \eta }{\text{mPa}}$	387.0 ± 7.5	187.0 ± 24	260.5 ± 16	120.0 ± 23	37.0 ± 33	32.55 ± 34	0.95 ± 15	-27.8 ± 0.52	-29.5 ± 3.6	-28.96 ± 1.2	12.05 ± 14
$\frac{\eta}{\text{mPa}}$	189.033 ± 0.97	39.83 ± 3.1	33.02 ± 2	23.43 ± 3	19.8 ± 4.2	9.34 ± 4.4	9.84 ± 2	11.53 ± 0.067	8.417 ± 0.46	6.162 ± 0.16	0.76 ± 1.8

Plot 9 Auftragung von η gegen c bei Saccharose für Scherrate 1



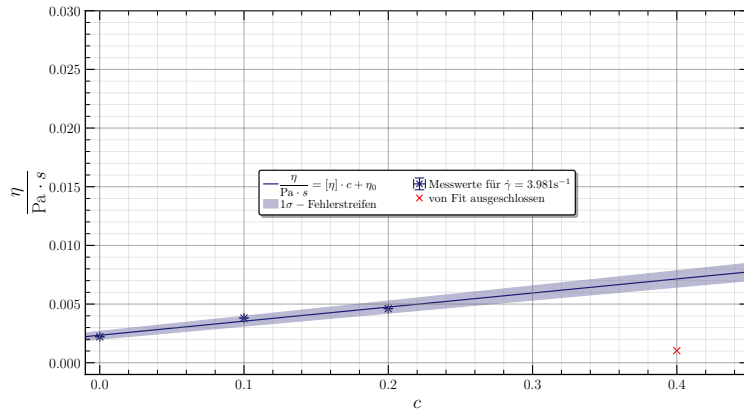
	$\dot{\gamma} = 1.000\text{s}^{-1}$	$\dot{\gamma} = 1.585\text{s}^{-1}$	$\dot{\gamma} = 2.511\text{s}^{-1}$	$\dot{\gamma} = 3.981\text{s}^{-1}$	$\dot{\gamma} = 6.310\text{s}^{-1}$	$\dot{\gamma} = 10.000\text{s}^{-1}$	$\dot{\gamma} = 15.850\text{s}^{-1}$	$\dot{\gamma} = 25.120\text{s}^{-1}$	$\dot{\gamma} = 39.810\text{s}^{-1}$	$\dot{\gamma} = 63.100\text{s}^{-1}$	$\dot{\gamma} = 100.000\text{s}^{-1}$
$\frac{ \eta }{\text{mPa}}$	387.0 ± 7.5	187.0 ± 24	260.5 ± 16	120.0 ± 23	37.0 ± 33	32.55 ± 34	0.95 ± 15	-27.8 ± 0.52	-29.5 ± 3.6	-28.96 ± 1.2	12.05 ± 14
$\frac{\eta}{\text{mPa}}$	189.033 ± 0.97	39.83 ± 3.1	33.02 ± 2	23.43 ± 3	19.8 ± 4.2	9.34 ± 4.4	9.84 ± 2	11.53 ± 0.067	8.417 ± 0.46	6.162 ± 0.16	0.76 ± 1.8

Plot 10 Auftragung von η gegen c bei Saccharose für Scherrate 2



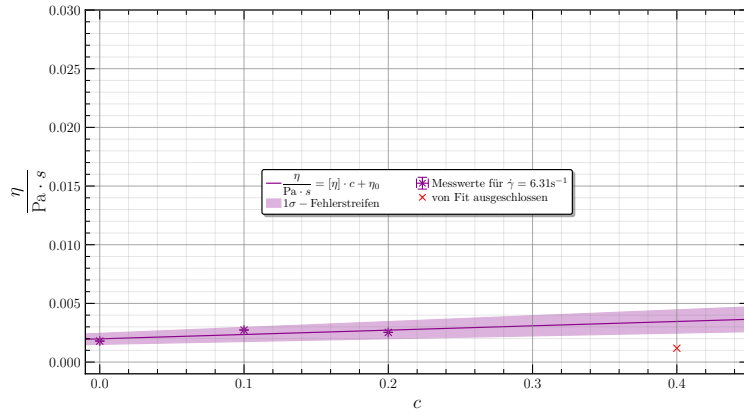
	$\dot{\gamma} = 1.000\text{s}^{-1}$	$\dot{\gamma} = 1.585\text{s}^{-1}$	$\dot{\gamma} = 2.511\text{s}^{-1}$	$\dot{\gamma} = 3.981\text{s}^{-1}$	$\dot{\gamma} = 6.310\text{s}^{-1}$	$\dot{\gamma} = 10.000\text{s}^{-1}$	$\dot{\gamma} = 15.850\text{s}^{-1}$	$\dot{\gamma} = 25.120\text{s}^{-1}$	$\dot{\gamma} = 39.810\text{s}^{-1}$	$\dot{\gamma} = 63.100\text{s}^{-1}$	$\dot{\gamma} = 100.000\text{s}^{-1}$
$\frac{[\eta]}{\text{ml/g}}$	387.0 ± 7.5	187.0 ± 24	260.5 ± 16	120.0 ± 23	37.0 ± 33	32.55 ± 34	0.95 ± 15	-27.8 ± 0.52	-29.5 ± 3.6	-28.96 ± 1.2	12.05 ± 14
$\frac{\eta_0}{\text{mPa}}$	189.033 ± 0.97	39.83 ± 3.1	33.02 ± 2	23.43 ± 3	19.8 ± 4.2	9.34 ± 4.4	9.84 ± 2	11.53 ± 0.067	8.417 ± 0.46	6.162 ± 0.16	0.76 ± 1.8

Plot 11 Auftragung von η gegen c bei Saccharose für Scherrate 3



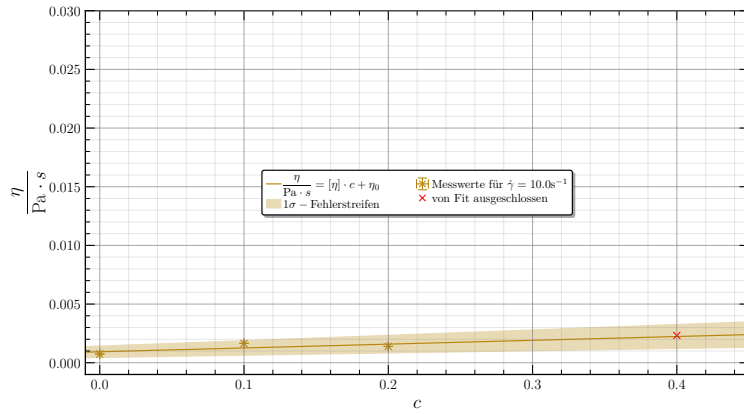
	$\dot{\gamma} = 1.000\text{s}^{-1}$	$\dot{\gamma} = 1.585\text{s}^{-1}$	$\dot{\gamma} = 2.511\text{s}^{-1}$	$\dot{\gamma} = 3.981\text{s}^{-1}$	$\dot{\gamma} = 6.310\text{s}^{-1}$	$\dot{\gamma} = 10.000\text{s}^{-1}$	$\dot{\gamma} = 15.850\text{s}^{-1}$	$\dot{\gamma} = 25.120\text{s}^{-1}$	$\dot{\gamma} = 39.810\text{s}^{-1}$	$\dot{\gamma} = 63.100\text{s}^{-1}$	$\dot{\gamma} = 100.000\text{s}^{-1}$
$\frac{[\eta]}{\text{ml/g}}$	387.0 ± 7.5	187.0 ± 24	260.5 ± 16	120.0 ± 23	37.0 ± 33	32.55 ± 34	0.95 ± 15	-27.8 ± 0.52	-29.5 ± 3.6	-28.96 ± 1.2	12.05 ± 14
$\frac{\eta_0}{\text{mPa}}$	189.033 ± 0.97	39.83 ± 3.1	33.02 ± 2	23.43 ± 3	19.8 ± 4.2	9.34 ± 4.4	9.84 ± 2	11.53 ± 0.067	8.417 ± 0.46	6.162 ± 0.16	0.76 ± 1.8

Plot 12 Auftragung von η gegen c bei Saccharose für Scherrate 4



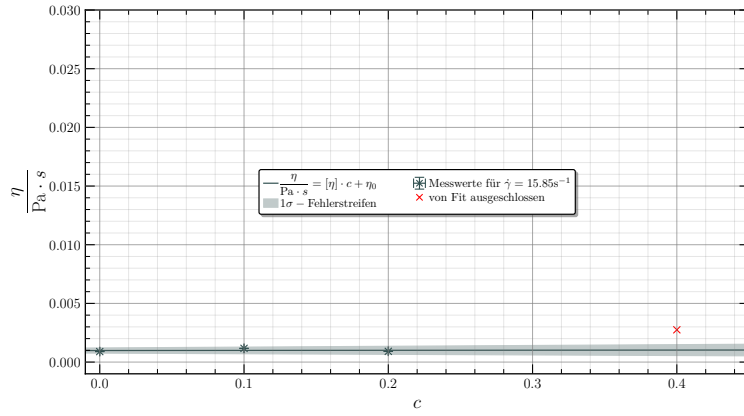
	$\dot{\gamma} = 1.000\text{s}^{-1}$	$\dot{\gamma} = 1.585\text{s}^{-1}$	$\dot{\gamma} = 2.511\text{s}^{-1}$	$\dot{\gamma} = 3.981\text{s}^{-1}$	$\dot{\gamma} = 6.310\text{s}^{-1}$	$\dot{\gamma} = 10.000\text{s}^{-1}$	$\dot{\gamma} = 15.850\text{s}^{-1}$	$\dot{\gamma} = 25.120\text{s}^{-1}$	$\dot{\gamma} = 39.810\text{s}^{-1}$	$\dot{\gamma} = 63.100\text{s}^{-1}$	$\dot{\gamma} = 100.000\text{s}^{-1}$
$\frac{[\eta]}{\text{m}^3/\text{kg}}$	387.0 ± 7.5	187.0 ± 24	260.5 ± 16	120.0 ± 23	37.0 ± 33	32.55 ± 34	0.95 ± 15	-27.8 ± 0.52	-29.5 ± 3.6	-28.96 ± 1.2	12.05 ± 14
$\frac{\eta_0}{\text{mPa}}$	189.033 ± 0.97	39.83 ± 3.1	33.02 ± 2	23.43 ± 3	19.8 ± 4.2	9.34 ± 4.4	9.84 ± 2	11.53 ± 0.067	8.417 ± 0.46	6.162 ± 0.16	0.76 ± 1.8

Plot 13 Auftragung von η gegen c bei Saccharose für Scherrate 5



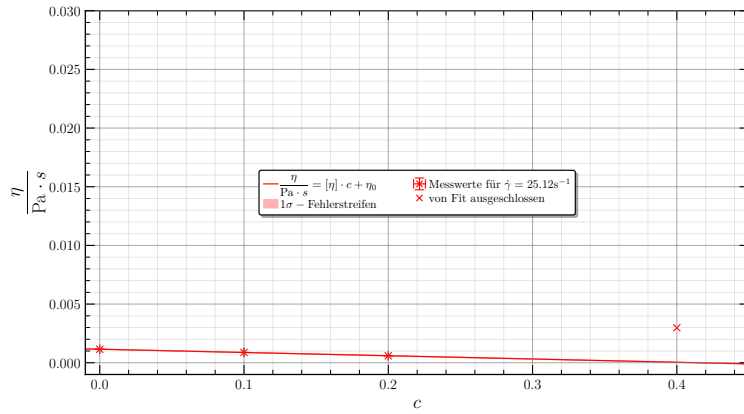
	$\dot{\gamma} = 1.000\text{s}^{-1}$	$\dot{\gamma} = 1.585\text{s}^{-1}$	$\dot{\gamma} = 2.511\text{s}^{-1}$	$\dot{\gamma} = 3.981\text{s}^{-1}$	$\dot{\gamma} = 6.310\text{s}^{-1}$	$\dot{\gamma} = 10.000\text{s}^{-1}$	$\dot{\gamma} = 15.850\text{s}^{-1}$	$\dot{\gamma} = 25.120\text{s}^{-1}$	$\dot{\gamma} = 39.810\text{s}^{-1}$	$\dot{\gamma} = 63.100\text{s}^{-1}$	$\dot{\gamma} = 100.000\text{s}^{-1}$
$\frac{[\eta]}{\text{m}^3/\text{kg}}$	387.0 ± 7.5	187.0 ± 24	260.5 ± 16	120.0 ± 23	37.0 ± 33	32.55 ± 34	0.95 ± 15	-27.8 ± 0.52	-29.5 ± 3.6	-28.96 ± 1.2	12.05 ± 14
$\frac{\eta_0}{\text{mPa}}$	189.033 ± 0.97	39.83 ± 3.1	33.02 ± 2	23.43 ± 3	19.8 ± 4.2	9.34 ± 4.4	9.84 ± 2	11.53 ± 0.067	8.417 ± 0.46	6.162 ± 0.16	0.76 ± 1.8

Plot 14 Auftragung von η gegen c bei Saccharose für Scherrate 6



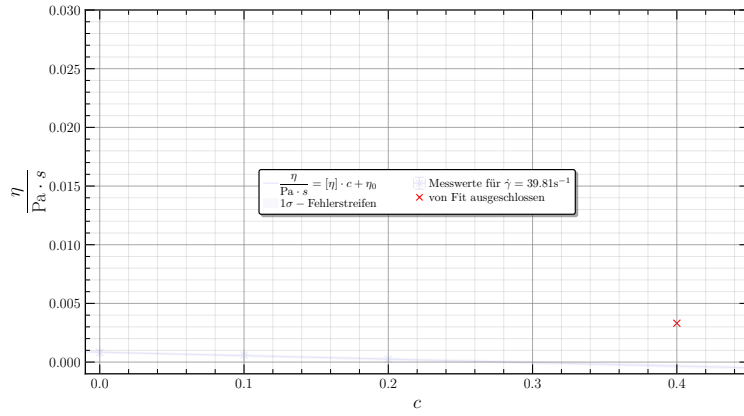
	$\dot{\gamma} = 1.000\text{s}^{-1}$	$\dot{\gamma} = 1.585\text{s}^{-1}$	$\dot{\gamma} = 2.511\text{s}^{-1}$	$\dot{\gamma} = 3.981\text{s}^{-1}$	$\dot{\gamma} = 6.310\text{s}^{-1}$	$\dot{\gamma} = 10.000\text{s}^{-1}$	$\dot{\gamma} = 15.850\text{s}^{-1}$	$\dot{\gamma} = 25.120\text{s}^{-1}$	$\dot{\gamma} = 39.810\text{s}^{-1}$	$\dot{\gamma} = 63.100\text{s}^{-1}$	$\dot{\gamma} = 100.000\text{s}^{-1}$
$\frac{[\eta]}{\text{mL/g}}$	387.0 ± 7.5	187.0 ± 24	260.5 ± 16	120.0 ± 23	37.0 ± 33	32.55 ± 34	0.95 ± 15	-27.8 ± 0.52	-29.5 ± 3.6	-28.96 ± 1.2	12.05 ± 14
$\frac{\eta_0}{\text{mPa}}$	189.033 ± 0.97	39.83 ± 3.1	33.02 ± 2	23.43 ± 3	19.8 ± 4.2	9.34 ± 4.4	9.84 ± 2	11.53 ± 0.067	8.417 ± 0.46	6.162 ± 0.16	0.76 ± 1.8

Plot 15 Auftragung von η gegen c bei Saccharose für Scherrate 7



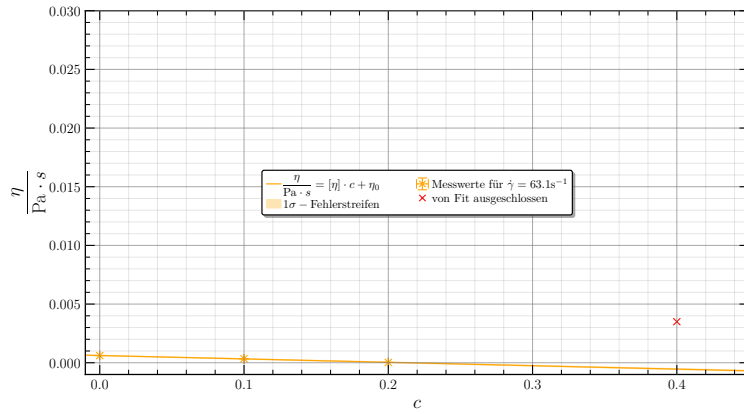
	$\dot{\gamma} = 1.000\text{s}^{-1}$	$\dot{\gamma} = 1.585\text{s}^{-1}$	$\dot{\gamma} = 2.511\text{s}^{-1}$	$\dot{\gamma} = 3.981\text{s}^{-1}$	$\dot{\gamma} = 6.310\text{s}^{-1}$	$\dot{\gamma} = 10.000\text{s}^{-1}$	$\dot{\gamma} = 15.850\text{s}^{-1}$	$\dot{\gamma} = 25.120\text{s}^{-1}$	$\dot{\gamma} = 39.810\text{s}^{-1}$	$\dot{\gamma} = 63.100\text{s}^{-1}$	$\dot{\gamma} = 100.000\text{s}^{-1}$
$\frac{[\eta]}{\text{mL/g}}$	387.0 ± 7.5	187.0 ± 24	260.5 ± 16	120.0 ± 23	37.0 ± 33	32.55 ± 34	0.95 ± 15	-27.8 ± 0.52	-29.5 ± 3.6	-28.96 ± 1.2	12.05 ± 14
$\frac{\eta_0}{\text{mPa}}$	189.033 ± 0.97	39.83 ± 3.1	33.02 ± 2	23.43 ± 3	19.8 ± 4.2	9.34 ± 4.4	9.84 ± 2	11.53 ± 0.067	8.417 ± 0.46	6.162 ± 0.16	0.76 ± 1.8

Plot 16 Auftragung von η gegen c bei Saccharose für Scherrate 8



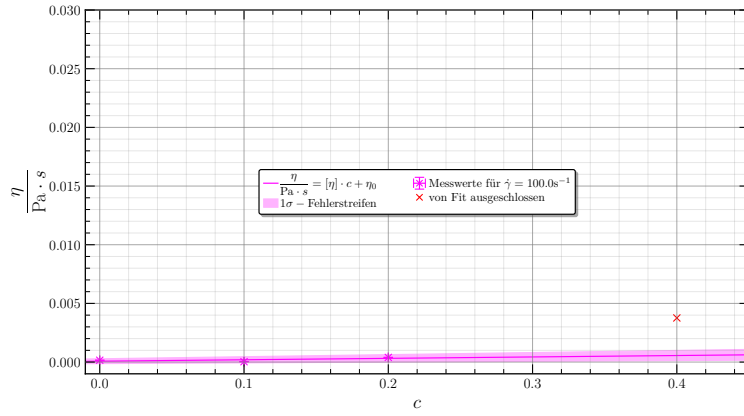
	$\dot{\gamma} = 1.000\text{s}^{-1}$	$\dot{\gamma} = 1.585\text{s}^{-1}$	$\dot{\gamma} = 2.511\text{s}^{-1}$	$\dot{\gamma} = 3.981\text{s}^{-1}$	$\dot{\gamma} = 6.310\text{s}^{-1}$	$\dot{\gamma} = 10.000\text{s}^{-1}$	$\dot{\gamma} = 15.850\text{s}^{-1}$	$\dot{\gamma} = 25.120\text{s}^{-1}$	$\dot{\gamma} = 39.810\text{s}^{-1}$	$\dot{\gamma} = 63.100\text{s}^{-1}$	$\dot{\gamma} = 100.000\text{s}^{-1}$
$\frac{[\eta]}{\text{mPa}}$	387.0 ± 7.5	187.0 ± 24	260.5 ± 16	120.0 ± 23	37.0 ± 33	32.55 ± 34	0.95 ± 15	-27.8 ± 0.52	-29.5 ± 3.6	-28.96 ± 1.2	12.05 ± 14
$\frac{\eta_0}{\text{mPa}}$	189.033 ± 0.97	39.83 ± 3.1	33.02 ± 2	23.43 ± 3	19.8 ± 4.2	9.34 ± 4.4	9.84 ± 2	11.53 ± 0.067	8.417 ± 0.46	6.162 ± 0.16	0.76 ± 1.8

Plot 17 Auftragung von η gegen c bei Saccharose für Scherrate 9



	$\dot{\gamma} = 1.000\text{s}^{-1}$	$\dot{\gamma} = 1.585\text{s}^{-1}$	$\dot{\gamma} = 2.511\text{s}^{-1}$	$\dot{\gamma} = 3.981\text{s}^{-1}$	$\dot{\gamma} = 6.310\text{s}^{-1}$	$\dot{\gamma} = 10.000\text{s}^{-1}$	$\dot{\gamma} = 15.850\text{s}^{-1}$	$\dot{\gamma} = 25.120\text{s}^{-1}$	$\dot{\gamma} = 39.810\text{s}^{-1}$	$\dot{\gamma} = 63.100\text{s}^{-1}$	$\dot{\gamma} = 100.000\text{s}^{-1}$
$\frac{[\eta]}{\text{mPa}}$	387.0 ± 7.5	187.0 ± 24	260.5 ± 16	120.0 ± 23	37.0 ± 33	32.55 ± 34	0.95 ± 15	-27.8 ± 0.52	-29.5 ± 3.6	-28.96 ± 1.2	12.05 ± 14
$\frac{\eta_0}{\text{mPa}}$	189.033 ± 0.97	39.83 ± 3.1	33.02 ± 2	23.43 ± 3	19.8 ± 4.2	9.34 ± 4.4	9.84 ± 2	11.53 ± 0.067	8.417 ± 0.46	6.162 ± 0.16	0.76 ± 1.8

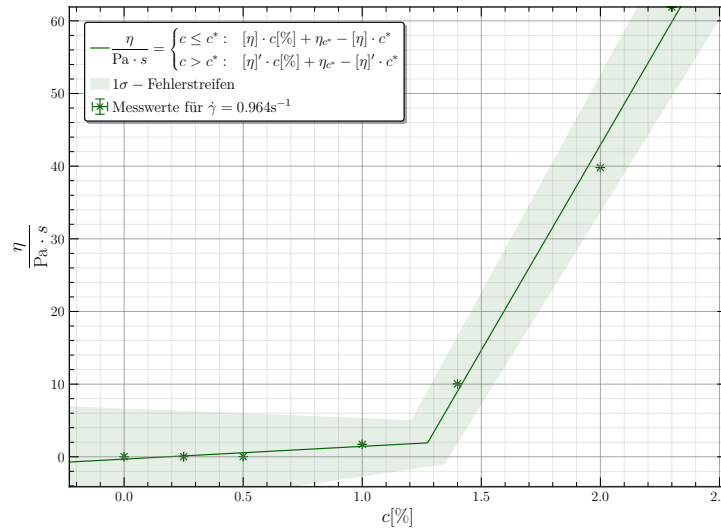
Plot 18 Auftragung von η gegen c bei Saccharose für Scherrate 10



	$\dot{\gamma} = 1.000\text{s}^{-1}$	$\dot{\gamma} = 1.585\text{s}^{-1}$	$\dot{\gamma} = 2.511\text{s}^{-1}$	$\dot{\gamma} = 3.981\text{s}^{-1}$	$\dot{\gamma} = 6.310\text{s}^{-1}$	$\dot{\gamma} = 10.000\text{s}^{-1}$	$\dot{\gamma} = 15.850\text{s}^{-1}$	$\dot{\gamma} = 25.120\text{s}^{-1}$	$\dot{\gamma} = 39.810\text{s}^{-1}$	$\dot{\gamma} = 63.100\text{s}^{-1}$	$\dot{\gamma} = 100.000\text{s}^{-1}$
$\frac{[\eta]}{\text{mL/g}}$	387.0 ± 7.5	187.0 ± 24	260.5 ± 16	120.0 ± 23	37.0 ± 33	32.55 ± 34	0.95 ± 15	-27.8 ± 0.52	-29.5 ± 3.6	-28.96 ± 1.2	12.05 ± 14
$\frac{\eta_0}{\text{Pa}\cdot\text{s}}$	189.033 ± 0.97	39.83 ± 3.1	33.02 ± 2	23.43 ± 3	19.8 ± 4.2	9.34 ± 4.4	9.84 ± 2	11.53 ± 0.067	8.417 ± 0.46	6.162 ± 0.16	0.76 ± 1.8

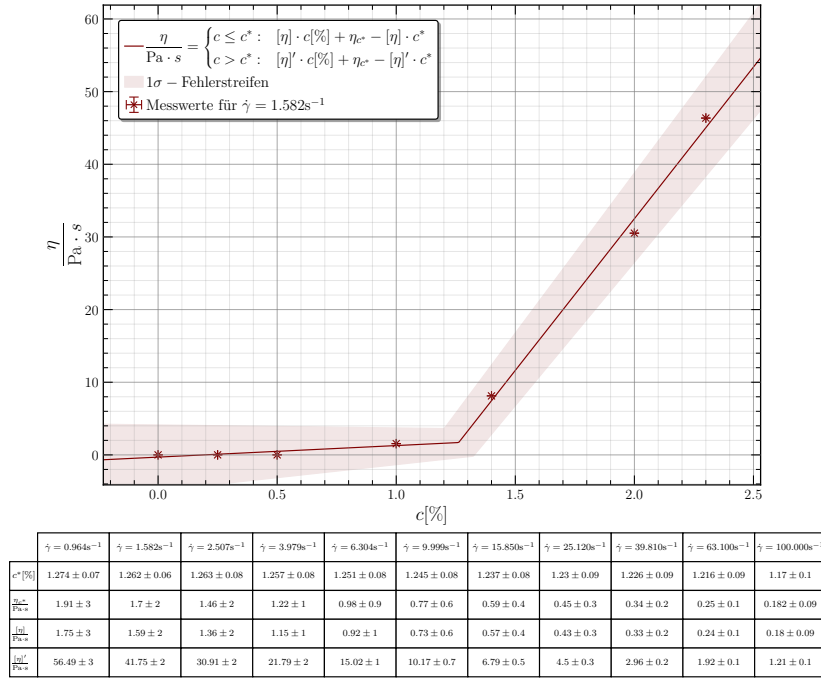
Plot 19 Auftragung von η gegen c bei Saccharose für Scherrate 11

A.2. Bestimmung der Überlappkonzentration für verschiedene Scherraten bei Guaran

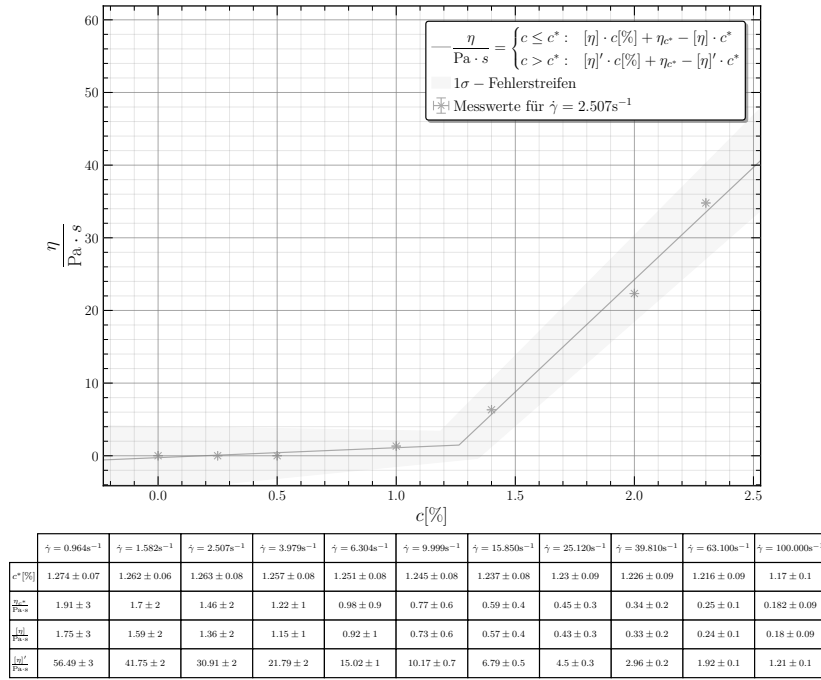


	$\dot{\gamma} = 0.964\text{s}^{-1}$	$\dot{\gamma} = 1.582\text{s}^{-1}$	$\dot{\gamma} = 2.507\text{s}^{-1}$	$\dot{\gamma} = 3.979\text{s}^{-1}$	$\dot{\gamma} = 6.304\text{s}^{-1}$	$\dot{\gamma} = 9.999\text{s}^{-1}$	$\dot{\gamma} = 15.850\text{s}^{-1}$	$\dot{\gamma} = 25.120\text{s}^{-1}$	$\dot{\gamma} = 39.810\text{s}^{-1}$	$\dot{\gamma} = 63.100\text{s}^{-1}$	$\dot{\gamma} = 100.000\text{s}^{-1}$
$c^* [\%]$	1.274 ± 0.07	1.262 ± 0.06	1.263 ± 0.08	1.257 ± 0.08	1.251 ± 0.08	1.245 ± 0.08	1.237 ± 0.08	1.23 ± 0.09	1.226 ± 0.09	1.216 ± 0.09	1.17 ± 0.1
$\frac{[\eta]}{\text{Pa}\cdot\text{s}}$	1.91 ± 3	1.7 ± 2	1.46 ± 2	1.22 ± 1	0.98 ± 0.9	0.77 ± 0.6	0.59 ± 0.4	0.45 ± 0.3	0.34 ± 0.2	0.25 ± 0.1	0.182 ± 0.09
$\frac{[\eta]'}{\text{Pa}\cdot\text{s}}$	1.75 ± 3	1.59 ± 2	1.36 ± 2	1.15 ± 1	0.92 ± 1	0.73 ± 0.6	0.57 ± 0.4	0.43 ± 0.3	0.33 ± 0.2	0.24 ± 0.1	0.18 ± 0.09
$\frac{[\eta]_0'}{\text{Pa}\cdot\text{s}}$	56.49 ± 3	41.75 ± 2	30.91 ± 2	21.79 ± 2	15.02 ± 1	10.17 ± 0.7	6.79 ± 0.5	4.5 ± 0.3	2.96 ± 0.2	1.92 ± 0.1	1.21 ± 0.1

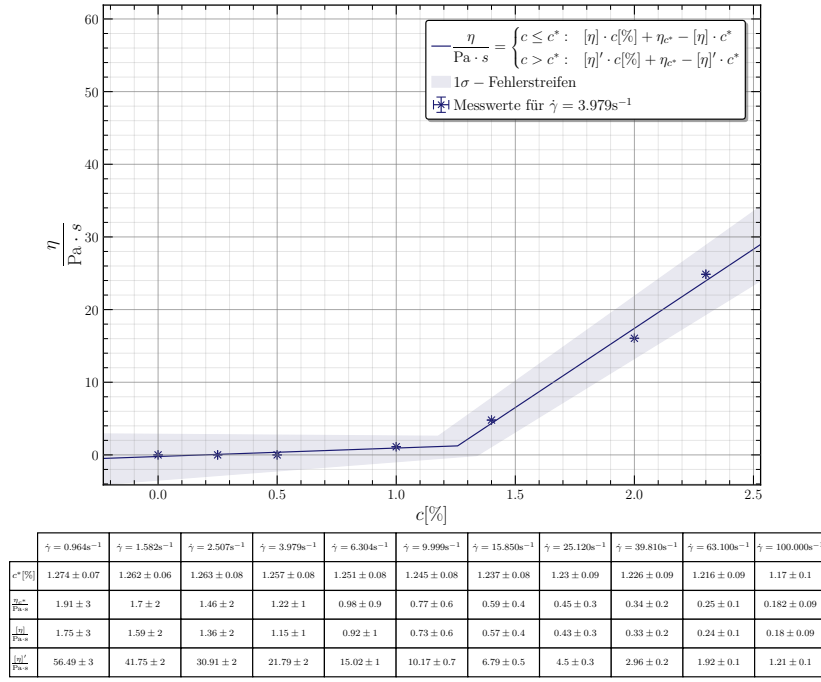
Plot 20 Auftragung von η gegen c bei Guaran für Scherrate 1



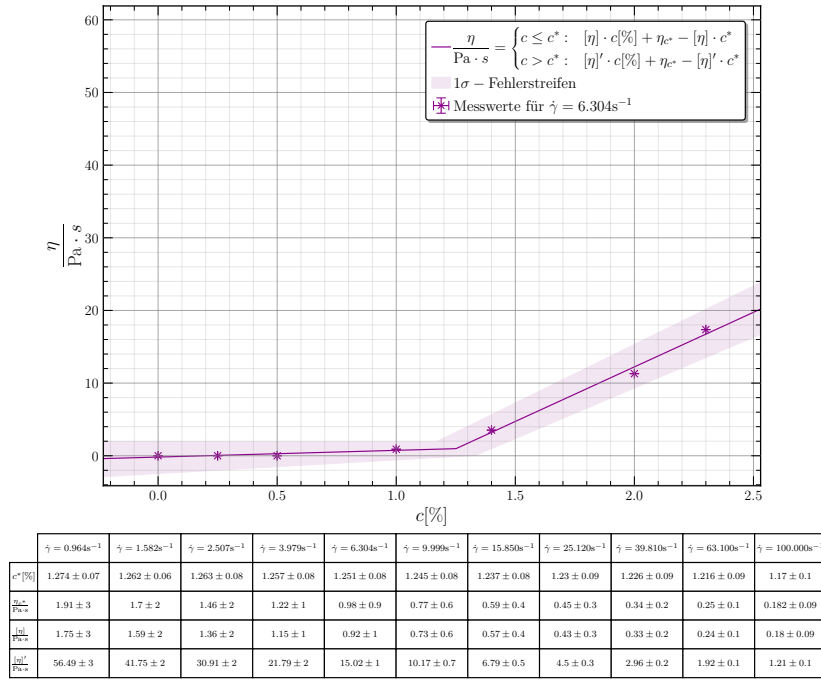
Plot 21 Auftragung von η gegen c bei Guaran für Scherrate 2



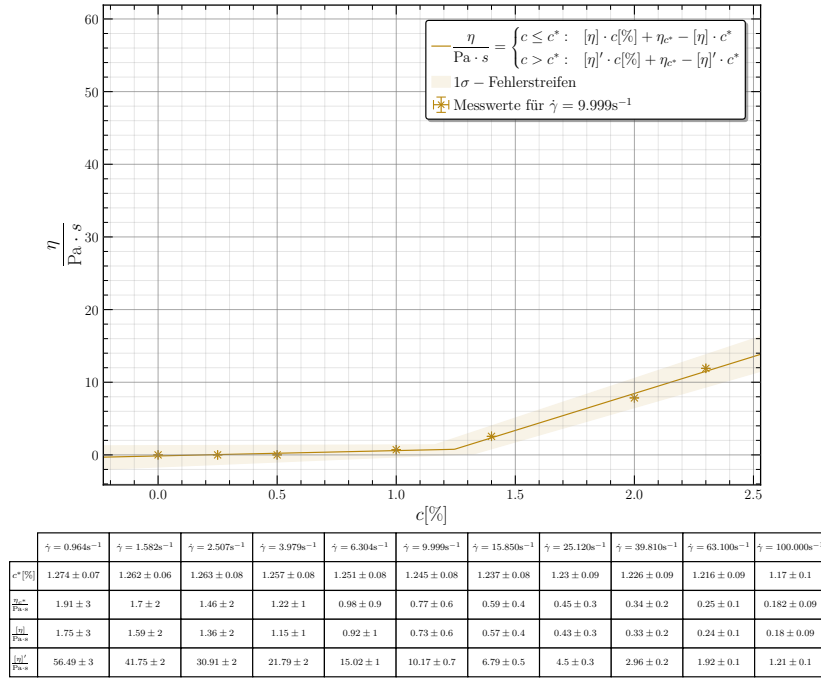
Plot 22 Auftragung von η gegen c bei Guaran für Scherrate 3



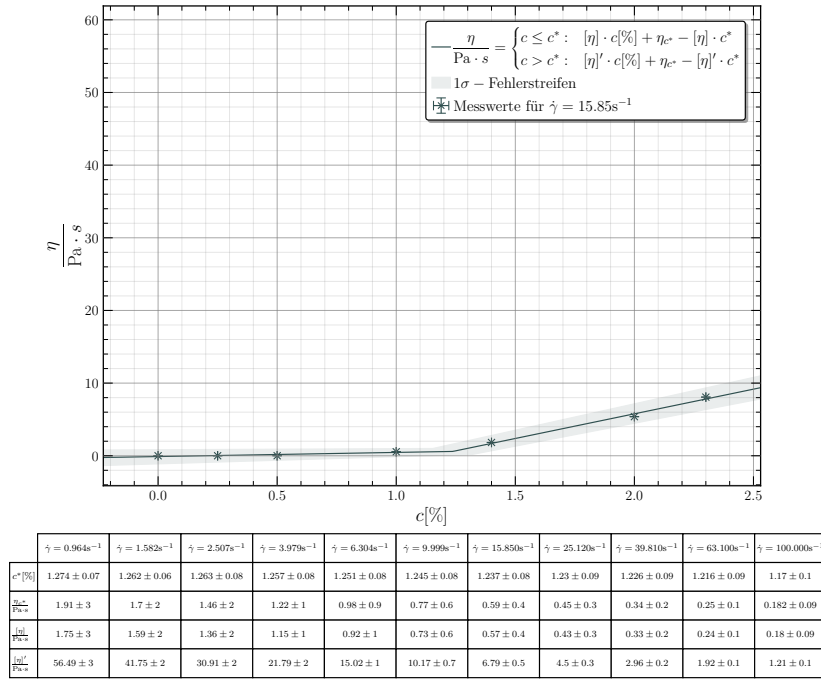
Plot 23 Auftragung von η gegen c bei Guaran für Scherrate 4



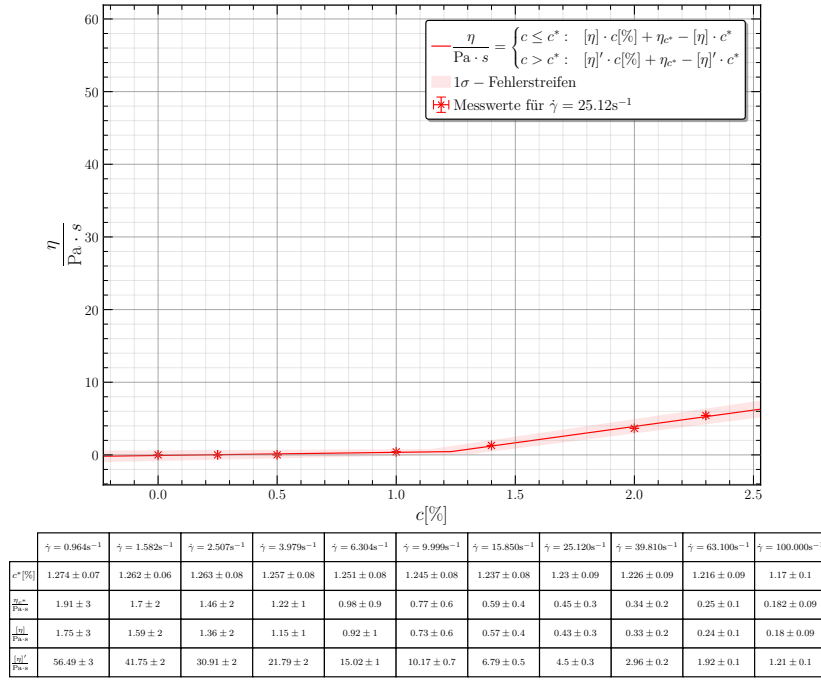
Plot 24 Auftragung von η gegen c bei Guaran für Scherrate 5



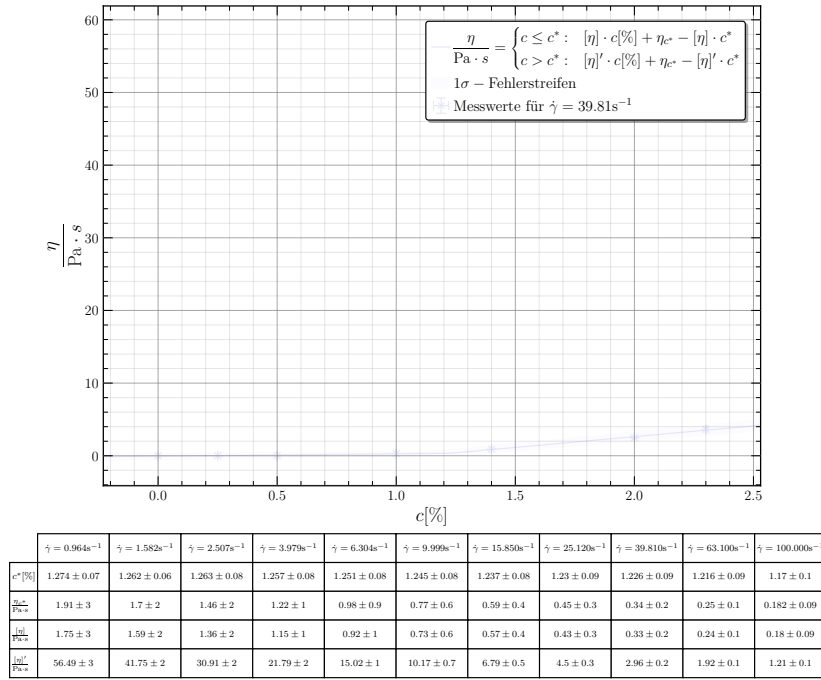
Plot 25 Auftragung von η gegen c bei Guaran für Scherrate 6



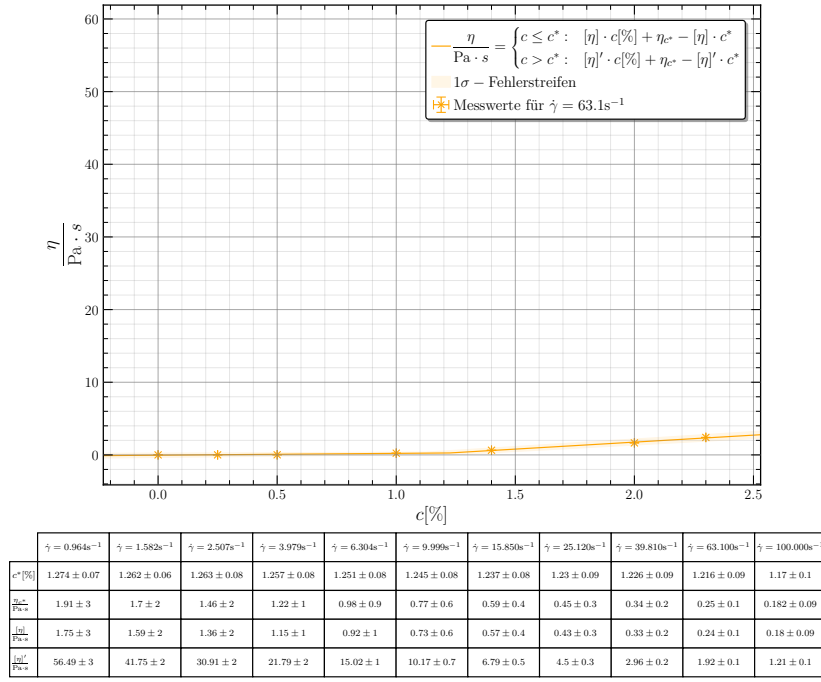
Plot 26 Auftragung von η gegen c bei Guaran für Scherrate 7



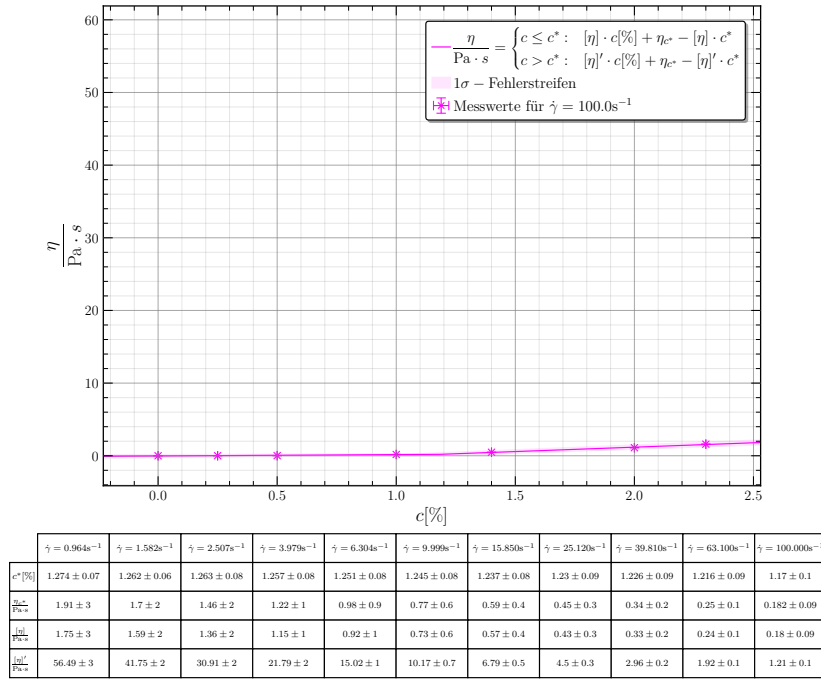
Plot 27 Auftragung von η gegen c bei Guaran für Scherrate 8



Plot 28 Auftragung von η gegen c bei Guaran für Scherrate 9



Plot 29 Auftragung von η gegen c bei Guaran für Scherrate 10



Plot 30 Auftragung von η gegen c bei Guaran für Scherrate 11

B. Python-Skripte zur Auswertung

B.1. Bestimmung des Potenzgesetzes für Saccharose

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Sat Dec 3 14:51:49 2022
4
5  @author: Jan-Philipp
6  """
7
8  import numpy as np #math functions, arrays
9  import matplotlib.pyplot as plt #visualizing
10 from scipy.optimize import curve_fit
11 from itertools import product, combinations
12 import matplotlib
13 import pandas as pd
14
15 matplotlib.style.use('JaPh') #merely contains basic style
   information
16 plt.ioff()
17
18
19 def LinXY(x,y):
20     """Scale x and y to obtain an linear relation between
       ordinate and abscissa"""
21     return np.log(x),np.log(y)
22
23 def LinXYerr(xerr,yerr,x,y):
24     """Scale xerr and yerr according to LinXY"""
25     return xerr/np.abs(x),yerr/np.abs(y)
26
27 def LinRegr1(x,a,b):
28     """Model of a linear function for fitting and plotting"""
29     return a*x+b
30
31 def LinRegr(tup, x):
32     """Model of a linear function for fitting and plotting"""
33     a,b = tup
34     return a*x+b
35
36 def LineIntersection(m1,m2,b1,b2):
37     """determining the geometric center of the 1-sigma-range"""
38     x = (b2-b1)/(m1-m2)
```



```

39     y = m1*x+b1
40     return x,y
41
42
43 def plot(CSVNAMES):
44
45     fig ,ax=plt.subplots(1,1,figsize=(10,10/np.sqrt(2)))
46
47     colorlist = ['forestgreen','maroon','orange','darkgray','
48                 midnightblue','darkmagenta','darkgoldenrod','
49                 darkslategray','red','lavender','magenta']
50
51     shear_ind = 2
52     cs = [0,10,20,40]
53     cs = np.array(cs)
54     maxy = -np.inf
55     miny = np.inf
56     stdList = []
57     poptList = []
58     markers = ['o','x','.', '^', '. ', '+', 'd']
59     ylabel = r'$\mathrm{ln}\left(\frac{\eta}{\mathrm{Pa}}\cdot\right)$'
60     xlabel = r'$\mathrm{ln}\left(\frac{\dot{\gamma}}{s^{-1}}\right)$'
61
62 for i,CSVNAME in enumerate(CSVNAMES):
63     PATH = CSVNAME + '.csv'
64     df = pd.read_csv(PATH, sep=',', header=None)
65     y=np.array(df.iloc[:,8])
66     x=np.array(df.iloc[:,7])
67
68     xerr_rel= xerr_abs= yerr_rel= yerr_abs = np.zeros(len(x))
69
70     xerr = x*xerr_rel+xerr_abs
71     yerr = y*yerr_rel+yerr_abs
72
73     X,Y = LinXY(x,y)
74     Xerr,Yerr = LinXYerr(xerr,yerr,x,y)
75
76     maxy = max(Y) if maxy<max(Y) else maxy
77     miny = min(Y) if miny>min(Y) else miny

```

```

78
79 dx = (max(X)-min(X))/10
80 dy = (maxy-miny)/15
81 xlim = (min(X)-dx,max(X)+dx)
82 ylim = (miny-dy,maxy+dy)
83
84
85 pop_t, pcov = curve_fit(LinRegr1,X,Y,maxfev=10000)
86 stdDev=np.sqrt(np.diag(pcov))
87
88 t1 = np.linspace(xlim[0],xlim[1], 500)
89
90
91
92 #ax.axvline(pop_t[0],marker='None',linestyle='dotted',
93             color='navajowhite',label=r'$c^*$')
94 #ax.axvspan(pop_t[0]-stdDev[0],pop_t[0]+stdDev[0],
95             linestyle='None',alpha=.2,color='navajowhite')
96 #ax.annotate(r'$c^*$',(pop_t[0],(pop_t[1]+min(Y))/2),
97             xytext=((pop_t[0]*1.5+max(X)*.5)/(2),(pop_t[1]+min(Y))/2),
98             arrowprops=dict(arrowstyle=matplotlib.patches.
99                             ArrowStyle("Fancy", head_length=.2, head_width=.2,
100                                     tail_width=.1),color='black'))
101
102 PointWise_LowerBound = np.zeros(len(t1))
103 PointWise_UpperBound = np.zeros(len(t1))
104 for tind in range(len(t1)):
105     t = t1[tind]
106     Min = LinRegr(pop_t,t)
107     Max = LinRegr(pop_t,t)
108     for comb1, comb2 in combinations(product([-1,1],
109                                         repeat=len(stdDev)),2):
110         p1 = pop_t+stdDev*comb1
111         p2 = pop_t+stdDev*comb2
112         Min = min(LinRegr(p1,t),Min)
113         Max = max(LinRegr(p2,t),Max)
114     PointWise_LowerBound[tind] = Min
115     PointWise_UpperBound[tind] = Max
116
117 if i == 0:
118     ax.plot(t1,LinRegr(pop_t,t1), marker = 'None',
119             linestyle = '-', color = colorlist[i], label=
120             ylabel[:-1]+'=\alpha_\cdot'+xlabel[1:-1]+'+\beta$')

```

```

112         #ax.fill_between(t1,PointWise_LowerBound,
113                           PointWise_UpperBound,color=colorlist[i], alpha =
114                           .1, label = r'$1\sigma-\text{Fehlerstreifen}$')
115     else:
116         ax.plot(t1,LinRegr(popt,t1), marker = 'None',
117                  linestyle = '-', color = colorlist[i])
118         #ax.fill_between(t1,PointWise_LowerBound,
119                           PointWise_UpperBound,color=colorlist[i], alpha =
120                           .1)
121         ax.plot(X,Y,marker=markers[i], color = colorlist[i],
122                  linestyle='None',label=r'Messwerte_für_<math>\beta</math>='+str(int(
123                  cs[i]))+'%$')
124         popList.append(popt)
125         stdList.append(stdDev)
126
127     cell_text = np.core.defchararray.add(np.core.defchararray.
128         add(np.round(np.transpose(popList)).astype(str),np.full
129         (np.transpose(popList).shape,'+')), np.round(np.
130         transpose(stdList)).astype(str))
131     #cell_text = np.core.defchararray.add(np.core.defchararray.
132         add(np.full(np.full(np.transpose(popList).shape,'$'),
133         cell_text)),np.full(np.transpose(popList).shape,'$'))
134     sigfig = 2
135     stdDevFlat = np.array(stdList).flatten()
136     popFlat = np.array(popList).flatten()
137
138     stdDev_rounded = ['{:g}'.format(float('{:.{p}g}'.format(
139         stdDevFlat[i], p=sigfig)))+'$' for i in range(0,len(
140         stdDevFlat)))]
141     decimals = [len(str(stdDev_rounded[i].split('.')[1])) for
142         i in range(0,len(stdDev_rounded))]
143     popFlat_rounded = [r'$'+str(round(popFlat[i],decimals[i]))+'\\
144         pm' for i in range(len(popFlat))]
145     popList = np.reshape(np.array(popFlat_rounded),np.array(
146         popList).shape)
147     stdList = np.reshape(np.array(stdDev_rounded),np.array(
148         stdList).shape)
149     cell_text = np.core.defchararray.add(np.transpose(popList)
150         , np.transpose(stdList).astype(str))
151
152     the_table = the_table = plt.table(cellText=cell_text ,
153         #rowLabels = np.full(7,'a'),
154         rowLabels=[r'$\alpha$',r'$\beta$'],

```

```

136         colLabels=[r '$c='+str(i)+'\%-$', for i in
137                     cs],
138         loc='bottom',
139         cellLoc='center',
140         bbox=[.1, -0.31, .8, 0.2],
141         edges='closed')
142     #the_table.auto_set_font_size(False)
143     #the_table.set_fontsize(8)
144     ax.set_ylim(ylim[0], maxy)
145     ax.set_xlabel(xlabel, size=16)
146     ax.set_ylabel(ylabel, size=16)
147     ax.set_xlim(xlim[0], xlim[1])
148     ax.grid(visible=True, which='minor', color="grey",
149             linestyle='-', linewidth=.008, alpha=.2)
150     ax.legend(loc='best', fontsize=13)
151
152     #fig.tight_layout()
153
154     #-----Save Figure-----
155     plt.savefig("
156         Water_Sucrose_Viscosity_vs_Shearrate_all_Concentrations.
157         pdf", dpi=1200)
158 if __name__ == "__main__":
159     plot(['sac-00-percent', 'sac-10-percent', 'sac-20-percent', '
160         sac-40-percent'])

```

```

1  # -*- coding: utf-8 -*-
2  """
3  Created on Sat Dec 3 16:40:27 2022
4
5  @author: Jan-Philipp
6  """
7
8
9  import numpy as np #math functions, arrays
10 import matplotlib.pyplot as plt #visualizing
11 from scipy.optimize import curve_fit
12 from itertools import product, combinations
13 import matplotlib
14 import pandas as pd

```

```

15
16 matplotlib.style.use('JaPh') #merely contains basic style
   information
17 plt.ioff()
18
19
20 def LinXY(x,y):
21     """Scale x and y to obtain an linear relation between
   ordinate and abscissa"""
22     return np.log(x),np.log(y)
23
24 def LinXYerr(xerr,yerr,x,y):
25     """Scale xerr and yerr according to LinXY"""
26     return xerr/np.abs(x),yerr/np.abs(y)
27
28 def LinRegr1(x,a,b):
29     """Model of a linear function for fitting and plotting"""
30     return a*x+b
31
32 def LinRegr(tup, x):
33     """Model of a linear function for fitting and plotting"""
34     a,b = tup
35     return a*x+b
36
37 def LineIntersection(m1,m2,b1,b2):
38     """determining the geometric center of the 1-sigma-range"""
39     x = (b2-b1)/(m1-m2)
40     y = m1*x+b1
41     return x,y
42
43
44 def plot(CSVNAMES):
45
46     fig,ax=plt.subplots(1,1,figsize=(10,10/np.sqrt(2)))
47
48     colorlist = ['forestgreen','maroon','orange','darkgray','
   midnightblue','darkmagenta','darkgoldenrod','
   darkslategray','red','lavender','magenta']
49
50     shear_ind = 2
51     cs = [0,10,20,40]
52     cs = np.array(cs)
53     maxy = -np.inf
54     miny = np.inf

```

```

55 stdList = []
56 poptList = []
57 markers = ['o', 'x', '.', '^', '. ', '+', 'd']
58 ylabel = r'$\mathrm{ln}\left(\frac{\sigma}{\mathrm{Pa}}\right)$'
59 xlabel = r'$\mathrm{ln}\left(\frac{\dot{\gamma}}{s^{-1}}\right)$'
60
61 for i, CSVNAME in enumerate(CSVNAMES):
62     PATH = CSVNAME + '.csv'
63     df = pd.read_csv(PATH, sep=',', header=None)
64     y=np.array(df.iloc[:,6])
65     x=np.array(df.iloc[:,7])
66
67
68
69     xerr_rel= xerr_abs= yerr_rel= yerr_abs = np.zeros(len(x))
70     xerr = x*xerr_rel+xerr_abs
71     yerr = y*yerr_rel+yerr_abs
72
73
74     X,Y = LinXY(x,y)
75     Xerr,Yerr = LinXYerr(xerr,yerr,x,y)
76
77     maxy = max(Y) if maxy<max(Y) else maxy
78     miny = min(Y) if miny>min(Y) else miny
79
80     dx = (max(X)-min(X))/10
81     dy = (maxy-miny)/15
82     xlim = (min(X)-dx,max(X)+dx)
83     ylim = (miny-dy,maxy+dy)
84
85
86     popt, pcov = curve_fit(LinRegr1,X,Y,maxfev=10000)
87     stdDev=np.sqrt(np.diag(pcov))
88
89     t1 = np.linspace(xlim[0],xlim[1], 500)
90
91
92
93     #ax.axvline(popt[0], marker='None', linestyle='dotted',
94               color='navajowhite', label=r'$c^*$')
95     #ax.axvspan(popt[0]-stdDev[0],popt[0]+stdDev[0],

```

```

95         linestyle='None', alpha=.2, color='navajowhite')
#ax.annotate(r'$c^*$', (popt[0], (popt[1]+min(Y))/2),
        xytext=((popt[0]*1.5+max(X)*.5)/(2), (popt[1]+min(Y))/2),
        arrowprops=dict(arrowstyle=matplotlib.patches.
        ArrowStyle("Fancy", head_length=.2, head_width=.2,
        tail_width=.1), color='black'))

96
97 PointWise_LowerBound = np.zeros(len(t1))
98 PointWise_UpperBound = np.zeros(len(t1))
99 for tind in range(len(t1)):
100     t = t1[tind]
101     Min = LinRegr(popt, t)
102     Max = LinRegr(popt, t)
103     for comb1, comb2 in combinations(product([-1,1],
        repeat=len(stdDev)), 2):
104         p1 = popt+stdDev*comb1
105         p2 = popt+stdDev*comb2
106         Min = min(LinRegr(p1, t), Min)
107         Max = max(LinRegr(p2, t), Max)
108     PointWise_LowerBound[tind] = Min
109     PointWise_UpperBound[tind] = Max
110
111 if i == 0:
112     ax.plot(t1, LinRegr(popt, t1), marker = 'None',
        linestyle = '-', color = colorlist[i], label=
        ylabel[: -1] + '\alpha\cdot' + xlabel[1: -1] + '+' + '\beta$')
113     #ax.fill_between(t1, PointWise_LowerBound,
        PointWise_UpperBound, color=colorlist[i], alpha =
        .1, label = r'$1\sigma-\text{Fehlerstreifen}$')
114 else:
115     ax.plot(t1, LinRegr(popt, t1), marker = 'None',
        linestyle = '-', color = colorlist[i])
116     #ax.fill_between(t1, PointWise_LowerBound,
        PointWise_UpperBound, color=colorlist[i], alpha =
        .1)
117 ax.plot(X, Y, marker=markers[i], color = colorlist[i],
        linestyle='None', label=r'Messwerte für $c$'+str(int(
        cs[i])) + '\%$')
118 poptList.append(popt)
119 stdList.append(stdDev)
120
121 cell_text = np.core.defchararray.add(np.core.defchararray.
        add(np.round(np.transpose(poptList)).astype(str), np.full

```

```

122     (np.transpose(poptList).shape, '+-')), np.round(np.
        transpose(stdList)).astype(str))
123 #cell_text = np.core.defchararray.add(np.core.defchararray.
        add(np.full(np.full(np.transpose(poptList).shape, '$'),
        cell_text)), np.full(np.transpose(poptList).shape, '$'))
124 sigfig = 2
125 stdDevFlat = np.array(stdList).flatten()
126 poptFlat = np.array(poptList).flatten()
127 stdDev_rounded = [ '{:g}'.format(float(' {:.{p}g}'.format(
        stdDevFlat[i], p=sigfig)))+ '$' for i in range(0, len(
        stdDevFlat))]
128 decimals = [len(str(stdDev_rounded[i].split('.')[1])) for
        i in range(0, len(stdDev_rounded))]
129 popt_rounded = [r '$'+str(round(poptFlat[i], decimals[i]))+' \
        pm' for i in range(len(poptFlat))]
130 poptList = np.reshape(np.array(popt_rounded), np.array(
        poptList).shape)
131 stdList = np.reshape(np.array(stdDev_rounded), np.array(
        stdList).shape)
132 cell_text = np.core.defchararray.add(np.transpose(poptList)
        , np.transpose(stdList).astype(str))
133
134 the_table = the_table = plt.table(cellText=cell_text ,
135     #rowLabels = np.full(7, 'a'),
136     rowLabels=[r '$\alpha$', r '$\beta$'],
137     colLabels=[r '$c='+str(i)+'\%_-$' for i in
        cs],
138     loc='bottom',
139     cellLoc='center',
140     bbox=[.1, -0.31, .8, 0.2],
141     edges='closed')
142 #the_table.auto_set_font_size(False)
143 #the_table.set_fontsize(8)
144
145 ax.set_ylim(ylim[0], maxy)
146 ax.set_xlabel(xlabel, size=16)
147 ax.set_ylabel(ylabel, size=16)
148 ax.set_xlim(xlim[0], xlim[1])
149 ax.grid(visible=True, which='minor', color="grey",
        linestyle='-', linewidth=.008, alpha=.2)
150
151 ax.legend(loc='best', fontsize=13)
152

```



```

153
154     #fig.tight_layout()
155
156     #——Save Figure——
157     plt.savefig("
158         Water_Sucrose_Stress_vs_Shearrate_all_Concentrations.pdf
159         ",dpi=1200)
160
161 if __name__ == "__main__":
162     plot(['sac_00_percent','sac_10_percent','sac_20_percent','sac_40_percent'])

```

B.2. Bestimmung des Potenzgesetzes für Guarán

```

1  # -*- coding: utf-8 -*-
2  """
3  Created on Sat Dec 3 16:47:25 2022
4
5  @author: Jan-Philipp
6  """
7
8  import numpy as np #math functions, arrays
9  import matplotlib.pyplot as plt #visualizing
10 from scipy.optimize import curve_fit
11 from itertools import product, combinations
12 import matplotlib
13 import pandas as pd
14
15 matplotlib.style.use('JaPh') #merely contains basic style
16     information
17 plt.ioff()
18
19 def LinXY(x,y):
20     """Scale x and y to obtain an linear relation between
21     ordinate and abscissa"""
22     return np.log(x),np.log(y)
23
24 def LinXYerr(xerr,yerr,x,y):
25     """Scale xerr and yerr according to LinXY"""
26     return xerr/np.abs(x),yerr/np.abs(y)
27
28 def LinRegr1(x,a,b):

```

```

28     """Model of a linear function for fitting and plotting"""
29     return a*x+b
30
31 def LinRegr(tup, x):
32     """Model of a linear function for fitting and plotting"""
33     a,b = tup
34     return a*x+b
35
36 def LineIntersection(m1,m2,b1,b2):
37     """determining the geometric center of the 1-sigma-range"""
38     x = (b2-b1)/(m1-m2)
39     y = m1*x+b1
40     return x,y
41
42
43 def plot(CSVNAMES):
44
45     fig,ax=plt.subplots(1,1,figsize=(10,10/np.sqrt(2)))
46
47     colorlist = ['forestgreen','maroon','orange','darkgray','
48                 midnightblue','darkmagenta','darkgoldenrod','
49                 darkslategray','red','lavender','magenta']
50
51     shear_ind = 2
52     cs = [0,.25,.5,1,1.4,2,2.3]
53     cs = np.array(cs)
54     maxy = -np.inf
55     miny = np.inf
56     stdList = []
57     poptList = []
58     markers = ['o','x','.', '^', '.', '+', 'd']
59     ylabel = r'$\mathrm{ln}\left(\frac{\sigma}{\mathrm{Pa}}\right)$'
60     xlabel = r'$\mathrm{ln}\left(\frac{\dot{\gamma}}{s^{-1}}\right)$'
61
62     for i,CSVNAME in enumerate(CSVNAMES):
63         PATH = CSVNAME + '.csv'
64         df = pd.read_csv(PATH, sep=',', header=None)
65         y=np.array(df.iloc[:,6])
66         x=np.array(df.iloc[:,7])
67

```

```

68     xerr_rel= xerr_abs= yerr_rel= yerr_abs = np.zeros(len(x
        ))
69     xerr = x*xerr_rel+xerr_abs
70     yerr = y*yerr_rel+yerr_abs
71
72
73     X,Y = LinXY(x,y)
74     Xerr ,Yerr = LinXYerr(xerr ,yerr ,x,y)
75
76     maxy = max(Y) if maxy<max(Y) else maxy
77     miny = min(Y) if miny>min(Y) else miny
78
79     dx = (max(X)-min(X)) /10
80     dy = (maxy-miny) /15
81     xlim = (min(X)-dx,max(X)+dx)
82     ylim = (miny-dy ,maxy+dy)
83
84
85     popt , pcov = curve_fit(LinRegr1,X,Y,maxfev=10000)
86     stdDev=np.sqrt(np.diag(pcov))
87
88     t1 = np.linspace(xlim[0],xlim[1], 500)
89
90
91
92     #ax.axvline(popt[0],marker='None',linestyle='dotted',
        color='navajowhite',label=r'$c^*$')
93     #ax.axvspan(popt[0]-stdDev[0],popt[0]+stdDev[0],
        linestyle='None',alpha=.2,color='navajowhite')
94     #ax.annotate(r'$c^*$',(popt[0],(popt[1]+min(Y))/2),
        xytext=((popt[0]*1.5+max(X)*.5)/(2),(popt[1]+min(Y))
        /2),arrowprops=dict(arrowstyle= matplotlib.patches.
        ArrowStyle("Fancy", head_length=.2, head_width=.2,
        tail_width=.1),color='black'))
95     """
96     PointWise_LowerBound = np.zeros(len(t1))
97     PointWise_UpperBound = np.zeros(len(t1))
98     for tind in range(len(t1)):
99         t = t1[tind]
100         Min = LinRegr(popt,t)
101         Max = LinRegr(popt,t)
102         for comb1, comb2 in combinations(product([-1,1],
            repeat=len(stdDev)),2):
103             p1 = popt+stdDev*comb1

```

```

104         p2 = poprt+stdDev*comb2
105         Min = min(LinRegr(p1, t), Min)
106         Max = max(LinRegr(p2, t), Max)
107         PointWise_LowerBound[tind] = Min
108         PointWise_UpperBound[tind] = Max
109     """
110     if i == 0:
111         ax.plot(t1, LinRegr(poprt, t1), marker = 'None',
112                 linestyle = '-', color = colorlist[i], label=
113                 ylabel[: -1] + '\alpha \cdot ' + xlabel[1: -1] + ' + \
114                 beta$')
115         #ax.fill_between(t1, PointWise_LowerBound,
116                         PointWise_UpperBound, color=colorlist[i], alpha =
117                         .1, label = r'$1\sigma - \text{Fehlerstreifen}$')
118     else:
119         ax.plot(t1, LinRegr(poprt, t1), marker = 'None',
120                 linestyle = '-', color = colorlist[i])
121         #ax.fill_between(t1, PointWise_LowerBound,
122                         PointWise_UpperBound, color=colorlist[i], alpha =
123                         .1)
124     ax.plot(X, Y, marker=markers[i], color = colorlist[i],
125             linestyle='None', label=r'Messwerte für $c=$'+str(np.
126             round(cs[i], 2))+'%$')
127     poprtList.append(poprt)
128     stdList.append(stdDev)
129
130     cell_text = np.core.defchararray.add(np.core.defchararray.
131         add(np.round(np.transpose(poprtList)).astype(str), np.full
132         (np.transpose(poprtList).shape, '+')), np.round(np.
133         transpose(stdList)).astype(str))
134     #cell_text = np.core.defchararray.add(np.core.defchararray.
135         add(np.full(np.full(np.transpose(poprtList).shape, '$'),
136         cell_text)), np.full(np.transpose(poprtList).shape, '$'))
137     sigfig = 2
138     stdDevFlat = np.array(stdList).flatten()
139     poprtFlat = np.array(poprtList).flatten()
140
141     stdDev_rounded = ['{:g}'.format(float('{:.{p}g}'.format(
142         stdDevFlat[i], p=sigfig)))+ '$' for i in range(0, len(
143         stdDevFlat))]
144     decimals = [len(str(stdDev_rounded[i].split('.')[ -1])) for
145         i in range(0, len(stdDev_rounded))]
146     poprt_rounded = [r'$'+str(round(poprtFlat[i], decimals[i]))+' \
147         pm' for i in range(len(poprtFlat))]

```

```

129     poptList = np.reshape(np.array(popt_rounded),np.array(
        poptList).shape)
130     stdList = np.reshape(np.array(stdDev_rounded),np.array(
        stdList).shape)
131     cell_text = np.core.defchararray.add(np.transpose(poptList)
        , np.transpose(stdList).astype(str))
132
133     the_table = the_table = plt.table(cellText=cell_text ,
134                                     #rowLabels = np.full(7, 'a'),
135                                     rowLabels=[r'$\alpha$',r'$\beta$'],
136                                     colLabels=[r'$c='+str(np.round(i,2))+'\%_
137                                     $' for i in cs],
138                                     loc='bottom',
139                                     cellLoc='center',
140                                     bbox=[0,-0.31,1,0.2],
141                                     edges='closed')
142     #the_table.auto_set_font_size(False)
143     #the_table.set_fontsize(8)
144
145     ax.set_ylim(ylim[0],maxy)
146     ax.set_xlabel(xlabel, size=16)
147     ax.set_ylabel(ylabel, size=16)
148     ax.set_xlim(xlim[0],xlim[1])
149     ax.grid(visible=True, which='minor', color="grey",
150           linestyle='-',linewidth=.008, alpha=.2)
151
152     ax.legend(loc='best',fontsize=11,ncol=2,columnspacing=0.6,
153           labelspace=.3)
154
155     #fig.tight_layout()
156
157     #-----Save Figure-----
158     plt.savefig("Guar-Stress-vs-Shearrate-all-Concentrations.
159           pdf",dpi=1200)
160
161 if __name__ == "__main__":
162     plot(['guar_00000_percent','guar_00025_percent','
163           guar_00050_percent','guar_00100_percent','
164           guar_00140_percent','guar_00200_percent','
165           guar_00230_percent'])

```

```

1  # -*- coding: utf-8 -*-
2  """

```

```

3  Created on Sat Dec 3 15:33:05 2022
4
5  @author: Jan-Philipp
6  """
7
8  import numpy as np #math functions, arrays
9  import matplotlib.pyplot as plt #visualizing
10 from scipy.optimize import curve_fit
11 from itertools import product, combinations
12 import matplotlib
13 import pandas as pd
14
15 matplotlib.style.use('JaPh') #merely contains basic style
   information
16 plt.ioff()
17
18
19 def LinXY(x,y):
20     """Scale x and y to obtain an linear relation between
   ordinate and abscissa"""
21     return np.log(x),np.log(y)
22
23 def LinXYerr(xerr,yerr,x,y):
24     """Scale xerr and yerr according to LinXY"""
25     return xerr/np.abs(x),yerr/np.abs(y)
26
27 def LinRegr1(x,a,b):
28     """Model of a linear function for fitting and plotting"""
29     return a*x+b
30
31 def LinRegr(tup, x):
32     """Model of a linear function for fitting and plotting"""
33     a,b = tup
34     return a*x+b
35
36 def LineIntersection(m1,m2,b1,b2):
37     """determining the geometric center of the 1-sigma-range"""
38     x = (b2-b1)/(m1-m2)
39     y = m1*x+b1
40     return x,y
41
42
43 def plot(CSVNAMES):
44

```

```

45 fig ,ax=plt.subplots(1,1,figsize=(10,10/np.sqrt(2)))
46
47 colorlist = ['forestgreen','maroon','orange','darkgray','
               midnightblue','darkmagenta','darkgoldenrod','
               darkslategray','red','lavender','magenta']
48
49 shear_ind = 2
50 cs = [0,.25,.5,1,1.4,2,2.3]
51 cs = np.array(cs)
52 maxy = -np.inf
53 miny = np.inf
54 stdList = []
55 poptList = []
56 markers = ['o','x','.', '^', 'v', '+', 'd']
57 ylabel = r'$\mathrm{ln}\left(\frac{\eta}{\mathrm{Pa}}\dot{\mathrm{Pa}}\cdot\right)$'
58 xlabel = r'$\mathrm{ln}\left(\frac{\dot{\gamma}}{s^{-1}}\right)$'
59
60 for i,CSVNAME in enumerate(CSVNAMES):
61     PATH = CSVNAME + '.csv'
62     df = pd.read_csv(PATH,sep=',',header=None)
63     y=np.array(df.iloc[:,8])
64     x=np.array(df.iloc[:,7])
65
66
67
68     xerr_rel= xerr_abs= yerr_rel= yerr_abs = np.zeros(len(x))
69
70     xerr = x*xerr_rel+xerr_abs
71     yerr = y*yerr_rel+yerr_abs
72
73     X,Y = LinXY(x,y)
74     Xerr,Yerr = LinXYerr(xerr,yerr,x,y)
75
76     maxy = max(Y) if maxy<max(Y) else maxy
77     miny = min(Y) if miny>min(Y) else miny
78
79     dx = (max(X)-min(X))/10
80     dy = (maxy-miny)/15
81     xlim = (min(X)-dx,max(X)+dx)
82     ylim = (miny-dy,maxy+dy)
83

```

```

84
85     popt, pcov = curve_fit(LinRegr1,X,Y,maxfev=10000)
86     stdDev=np.sqrt(np.diag(pcov))
87
88     t1 = np.linspace(xlim[0],xlim[1], 500)
89
90
91
92     #ax.axvline(popt[0],marker='None',linestyle='dotted',
93               color='navajowhite',label=r'$c^*$')
94     #ax.axvspan(popt[0]-stdDev[0],popt[0]+stdDev[0],
95               linestyle='None',alpha=.2,color='navajowhite')
96     #ax.annotate(r'$c^*$',(popt[0],(popt[1]+min(Y))/2),
97               xytext=((popt[0]*1.5+max(X)*.5)/(2),(popt[1]+min(Y))/2),
98               arrowprops=dict(arrowstyle=matplotlib.patches.
99                             ArrowStyle("Fancy", head_length=.2, head_width=.2,
100                                tail_width=.1),color='black'))
101
102     """
103     PointWise_LowerBound = np.zeros(len(t1))
104     PointWise_UpperBound = np.zeros(len(t1))
105     for tind in range(len(t1)):
106         t = t1[tind]
107         Min = LinRegr(popt,t)
108         Max = LinRegr(popt,t)
109         for comb1, comb2 in combinations(product([-1,1],
110                                repeat=len(stdDev)),2):
111             p1 = popt+stdDev*comb1
112             p2 = popt+stdDev*comb2
113             Min = min(LinRegr(p1,t),Min)
114             Max = max(LinRegr(p2,t),Max)
115         PointWise_LowerBound[tind] = Min
116         PointWise_UpperBound[tind] = Max
117     """
118     if i == 0:
119         ax.plot(t1,LinRegr(popt,t1), marker = 'None',
120               linestyle = '-', color = colorlist[i], label=
121               ylabel[: -1] + '\alpha\cdot'+xlabel[1: -1] + '+\beta$')
122         #ax.fill_between(t1,PointWise_LowerBound,
123               PointWise_UpperBound,color=colorlist[i], alpha =
124               .1, label = r'$1\sigma-\text{Fehlerstreifen}$')
125     else:
126         ax.plot(t1,LinRegr(popt,t1), marker = 'None',
127               linestyle = '-', color = colorlist[i])

```



```

115         #ax.fill_between(t1,PointWise_LowerBound,
            PointWise_UpperBound,color=colorlist[i], alpha =
            .1)
116     ax.plot(X,Y,marker=markers[i], color = colorlist[i],
            linestyle='None',label=r'Messwerte_für_$(c='+str(np.
            round(cs[i],2))+'\%$')
117     popList.append(popt)
118     stdList.append(stdDev)
119
120     cell_text = np.core.defchararray.add(np.core.defchararray.
            add(np.round(np.transpose(popList)).astype(str),np.full
            (np.transpose(popList).shape,'+')), np.round(np.
            transpose(stdList)).astype(str))
121     #cell_text = np.core.defchararray.add(np.core.defchararray.
            add(np.full(np.full(np.transpose(popList).shape,'$'),
            cell_text)),np.full(np.transpose(popList).shape,'$'))
122     sigfig = 2
123     stdDevFlat = np.array(stdList).flatten()
124     popFlat = np.array(popList).flatten()
125
126     stdDev_rounded = ['{:g}'.format(float('{:.{p}g}'.format(
            stdDevFlat[i], p=sigfig)))+'$' for i in range(0,len(
            stdDevFlat))]
127     decimals = [len(str(stdDev_rounded[i].split('.')[1])) for
            i in range(0,len(stdDev_rounded))]
128     pop_rounded = [r'$'+str(round(popFlat[i],decimals[i]))+'\
            pm' for i in range(len(popFlat))]
129     popList = np.reshape(np.array(pop_rounded),np.array(
            popList).shape)
130     stdList = np.reshape(np.array(stdDev_rounded),np.array(
            stdList).shape)
131     cell_text = np.core.defchararray.add(np.transpose(popList)
            , np.transpose(stdList).astype(str))
132
133     the_table = the_table = plt.table(cellText=cell_text ,
134         #rowLabels = np.full(7,'a'),
135         rowLabels=[r'$\alpha$',r'$\beta$'],
136         colLabels=[r'$c='+str(np.round(i,2))+'\%_
            $' for i in cs],
137         loc='bottom',
138         cellLoc='center',
139         bbox=[0,-0.31,1,0.2],
140         edges='closed')
141     #the_table.auto_set_font_size(False)

```

```

142     #the_table.set_fontsize(8)
143
144     ax.set_ylim(ylim[0],maxy)
145     ax.set_xlabel(xlabel,size=16)
146     ax.set_ylabel(ylabel,size=16)
147     ax.set_xlim(xlim[0],xlim[1])
148     ax.grid(visible=True, which='minor', color="grey",
149             linestyle='-',linewidth=.008, alpha=.2)
150
151     ax.legend(loc='best',fontsize=11,ncol=2,columnspacing=0.6,
152             labelspace=.3)
153
154     #fig.tight_layout()
155
156     #-----Save Figure-----
157     plt.savefig("Guar_Viscosity_vs_Shearrate_all-Concentrations
158                 .pdf",dpi=1200)
159
160 if __name__ == "__main__":
161     plot(['guar_00000-percent','guar_00025-percent','
162           guar_00050-percent','guar_00100-percent','
163           guar_00140-percent','guar_00200-percent','
164           guar_00230-percent'])

```

B.3. Bestimmung der Konzentrationsabhängigkeit der Viskosität bei Saccharose

```

1  # -*- coding: utf-8 -*-
2  """
3  Created on Sat Dec 3 16:40:27 2022
4
5  @author: Jan-Philipp
6  """
7
8
9  import numpy as np #math functions, arrays
10 import matplotlib.pyplot as plt #visualizing
11 from scipy.optimize import curve_fit
12 from itertools import product, combinations
13 import matplotlib
14 import pandas as pd
15

```

```

16 matplotlib.style.use('JaPh') #merely contains basic style
   information
17 plt.ioff()
18
19
20 def LinXY(x,y):
21     """Scale x and y to obtain an linear relation between
   ordinate and abscissa"""
22     return np.log(x),np.log(y)
23
24 def LinXYerr(xerr,yerr,x,y):
25     """Scale xerr and yerr according to LinXY"""
26     return xerr/np.abs(x),yerr/np.abs(y)
27
28 def LinRegr1(x,a,b):
29     """Model of a linear function for fitting and plotting"""
30     return a*x+b
31
32 def LinRegr(tup, x):
33     """Model of a linear function for fitting and plotting"""
34     a,b = tup
35     return a*x+b
36
37 def LineIntersection(m1,m2,b1,b2):
38     """determining the geometric center of the 1-sigma-range"""
39     x = (b2-b1)/(m1-m2)
40     y = m1*x+b1
41     return x,y
42
43
44 def plot(CSVNAMES):
45
46     fig,ax=plt.subplots(1,1,figsize=(10,10/np.sqrt(2)))
47
48     colorlist = ['forestgreen','maroon','orange','darkgray','
   midnightblue','darkmagenta','darkgoldenrod','
   darkslategray','red','lavender','magenta']
49
50     shear_ind = 2
51     cs = [0,10,20,40]
52     cs = np.array(cs)
53     maxy = -np.inf
54     miny = np.inf
55     stdList = []

```

```

56     poptList = []
57     markers = ['o', 'x', '.', '^', '._', '+', 'd']
58     ylabel = r'$\mathrm{ln}\left(\frac{\sigma}{\mathrm{Pa}}\right)$'
59     xlabel = r'$\mathrm{ln}\left(\frac{\dot{\gamma}}{s^{-1}}\right)$'
60
61     for i, CSVNAME in enumerate(CSVNAMES):
62         PATH = CSVNAME + '.csv'
63         df = pd.read_csv(PATH, sep=',', header=None)
64         y=np.array(df.iloc[:,6])
65         x=np.array(df.iloc[:,7])
66
67
68
69         xerr_rel= xerr_abs= yerr_rel= yerr_abs = np.zeros(len(x))
70         xerr = x*xerr_rel+xerr_abs
71         yerr = y*yerr_rel+yerr_abs
72
73
74         X,Y = LinXY(x,y)
75         Xerr,Yerr = LinXYerr(xerr,yerr,x,y)
76
77         maxy = max(Y) if maxy<max(Y) else maxy
78         miny = min(Y) if miny>min(Y) else miny
79
80         dx = (max(X)-min(X))/10
81         dy = (maxy-miny)/15
82         xlim = (min(X)-dx,max(X)+dx)
83         ylim = (miny-dy,maxy+dy)
84
85
86         popt, pcov = curve_fit(LinRegr1,X,Y,maxfev=10000)
87         stdDev=np.sqrt(np.diag(pcov))
88
89         t1 = np.linspace(xlim[0],xlim[1], 500)
90
91
92
93         #ax.axvline(popt[0], marker='None', linestyle='dotted',
94                 color='navajowhite', label=r'$c^*$')
95         #ax.axvspan(popt[0]-stdDev[0],popt[0]+stdDev[0],
96                 linestyle='None', alpha=.2, color='navajowhite')

```

```

95      #ax.annotate(r'$c^*$', (popt[0], (popt[1]+min(Y))/2),
      xytext=((popt[0]*1.5+max(X)*.5)/(2), (popt[1]+min(Y))/2),
      arrowprops=dict(arrowstyle= matplotlib.patches.
      ArrowStyle("Fancy", head_length=.2, head_width=.2,
      tail_width=.1), color='black'))

96
97      PointWise_LowerBound = np.zeros(len(t1))
98      PointWise_UpperBound = np.zeros(len(t1))
99      for tind in range(len(t1)):
100          t = t1[tind]
101          Min = LinRegr(popt, t)
102          Max = LinRegr(popt, t)
103          for comb1, comb2 in combinations(product([-1,1],
      repeat=len(stdDev)), 2):
104              p1 = popt+stdDev*comb1
105              p2 = popt+stdDev*comb2
106              Min = min(LinRegr(p1, t), Min)
107              Max = max(LinRegr(p2, t), Max)
108          PointWise_LowerBound[tind] = Min
109          PointWise_UpperBound[tind] = Max
110
111      if i == 0:
112          ax.plot(t1, LinRegr(popt, t1), marker = 'None',
      linestyle = '-', color = colorlist[i], label=
      ylabel[: -1] + '\alpha \cdot ' + xlabel[1: -1] + ' + \beta $')
113          #ax.fill_between(t1, PointWise_LowerBound,
      PointWise_UpperBound, color=colorlist[i], alpha =
      .1, label = r'$1\sigma - \text{Fehlerstreifen}$')
114      else:
115          ax.plot(t1, LinRegr(popt, t1), marker = 'None',
      linestyle = '-', color = colorlist[i])
116          #ax.fill_between(t1, PointWise_LowerBound,
      PointWise_UpperBound, color=colorlist[i], alpha =
      .1)
117      ax.plot(X, Y, marker=markers[i], color = colorlist[i],
      linestyle='None', label=r'Messwerte für $c$'+str(int(
      cs[i])) + '\% $')
118      poptList.append(popt)
119      stdList.append(stdDev)
120
121      cell_text = np.core.defchararray.add(np.core.defchararray.
      add(np.round(np.transpose(poptList)).astype(str), np.full
      (np.transpose(poptList).shape, '+'))), np.round(np.

```

```

122     transpose(stdList)).astype(str)
123     #cell_text = np.core.defchararray.add(np.core.defchararray.
124         add(np.full(np.full(np.transpose(poptList).shape, '$'),
125             cell_text)), np.full(np.transpose(poptList).shape, '$'))
126     sigfig = 2
127     stdDevFlat = np.array(stdList).flatten()
128     poptFlat = np.array(poptList).flatten()
129     stdDev_rounded = [ '{:g}'.format(float('{:.{p}g}'.format(
130         stdDevFlat[i], p=sigfig)))+ '$' for i in range(0, len(
131         stdDevFlat))]
132     decimals = [len(str(stdDev_rounded[i].split('.')[1])) for
133         i in range(0, len(stdDev_rounded))]
134     popt_rounded = [r '$'+str(round(poptFlat[i], decimals[i]))+'\
135         pm' for i in range(len(poptFlat))]
136     poptList = np.reshape(np.array(popt_rounded), np.array(
137         poptList).shape)
138     stdList = np.reshape(np.array(stdDev_rounded), np.array(
139         stdList).shape)
140     cell_text = np.core.defchararray.add(np.transpose(poptList)
141         , np.transpose(stdList).astype(str))
142
143     the_table = the_table = plt.table(cellText=cell_text ,
144         #rowLabels = np.full(7, 'a'),
145         rowLabels=[r '$\alpha$', r '$\beta$'],
146         colLabels=[r '$c='+str(i)+'\%-' for i in
147             cs],
148         loc='bottom',
149         cellLoc='center',
150         bbox=[.1, -0.31, .8, 0.2],
151         edges='closed')
152     #the_table.auto_set_font_size(False)
153     #the_table.set_fontsize(8)
154
155     ax.set_ylim(ylim[0], maxy)
156     ax.set_xlabel(xlabel, size=16)
157     ax.set_ylabel(ylabel, size=16)
158     ax.set_xlim(xlim[0], xlim[1])
159     ax.grid(visible=True, which='minor', color="grey",
160         linestyle='-', linewidth=.008, alpha=.2)
161
162     ax.legend(loc='best', fontsize=13)

```

```

154     #fig.tight_layout()
155
156     #——Save Figure——
157     plt.savefig("
        Water_Sucrose_Stress_vs_Shearrate_all_Concentrations.pdf
        ",dpi=1200)
158
159 if __name__ == "__main__":
160     plot(['sac_00_percent','sac_10_percent','sac_20_percent','
        sac_40_percent'])

```

B.4. Bestimmung der Konzentrationabhängigkeit der Viskosität bei Guaran

```

1  # -*- coding: utf-8 -*-
2  """
3  Created on Mon Nov 28 10:49:25 2022
4
5  @author: Jan-Philipp
6  """
7
8  import numpy as np #math functions, arrays
9  import matplotlib.pyplot as plt #visualizing
10 from scipy.optimize import curve_fit
11 from itertools import product, combinations
12 import matplotlib
13 import pandas as pd
14
15 matplotlib.style.use('JaPh') #merely contains basic style
    information
16 plt.ioff()
17
18
19 def LinXY(x,y):
20     """Scale x and y to obtain an linear relation between
        ordinate and abscissa"""
21     return x,y
22
23
24 def LinXYerr(xerr,yerr,x,y):
25     """Scale xerr and yerr according to LinXY"""
26     return xerr,yerr
27

```

```

28 def LinRegr1(x, x0, y0, k1, k2):
29     """Model of a piecewise linear function for fitting and
        plotting"""
30     tup = (x0, y0, k1, k2)
31     return LinRegr(tup, x)
32
33 def LinRegr(tup, x):
34     """Model of a piecewise linear function for fitting and
        plotting"""
35     x0, y0, k1, k2 = tup
36     return np.piecewise(x, [x < x0], [lambda x:k1*x + y0-k1*x0,
        lambda x:k2*x + y0-k2*x0])
37
38 def LineIntersection(m1,m2,b1,b2):
39     """determining the geometric center of the 1-sigma-range"""
40     x = (b2-b1)/(m1-m2)
41     y = m1*x+b1
42     return x,y
43
44 def plot(CSVNAMES):
45     global cellcand
46
47     fig ,ax=plt.subplots(1,1,figsize=(10,10/np.sqrt(2)))
48
49     colorlist = ['darkgreen','maroon','darkgray','midnightblue',
        'darkmagenta','darkgoldenrod','darkslategray','red',''
        lavender','orange','magenta']
50
51     shear_ind = 2
52     x = [0,.25,.5,1,1.4,2,2.3]
53     x = np.array(x)
54     maxy = 0
55     stdList = []
56     poptList = []
57     gammadotList = []
58     for shear_ind in range(0,11):
59         y = []
60         for CSVNAME in CSVNAMES:
61             PATH = CSVNAME + '.csv'
62             df = pd.read_csv(PATH,sep=',',header=None)
63             y.append(df.iloc[shear_ind,8])
64
65         y = np.array(y)
66

```



```

67     maxy = max(y) if maxy<max(y) else maxy
68
69     xerr_rel= xerr_abs= yerr_rel= yerr_abs = np.zeros(len(x
70     ))
71     xerr = x*xerr_rel+xerr_abs
72     yerr = y*yerr_rel+yerr_abs
73
74     ylabel = r '$\dfrac{\eta}{\mathrm{Pa}}\cdot s$',
75     xlabel = r '$c[\%]$'
76
77     X,Y = LinXY(x,y)
78     Xerr,Yerr = LinXYerr(xerr,yerr,x,y)
79     dx = (max(X)-min(X))/10
80     dy = (maxy-min(Y))/15
81     xlim = (min(X)-dx,max(X)+dx)
82     ylim = (min(Y)-dy,maxy+dy)
83
84     popt, pcov = curve_fit(LinRegr1,X,Y,p0=(1.3,5,2,35),
85     maxfev=10000)
86     stdDev=np.sqrt(np.diag(pcov))
87
88     t1 = np.linspace(xlim[0],xlim[1], 500)
89
90
91     #ax.axvline(popt[0],marker='None',linestyle='dotted',
92     color='navajowhite',label=r '$c^*$')
93     #ax.axvspan(popt[0]-stdDev[0],popt[0]+stdDev[0],
94     linestyle='None',alpha=.2,color='navajowhite')
95     #ax.annotate(r '$c^*$',(popt[0],(popt[1]+min(Y))/2),
96     xytext=((popt[0]*1.5+max(X)*.5)/(2),(popt[1]+min(Y))
97     /2),arrowprops=dict(arrowstyle= matplotlib.patches.
98     ArrowStyle("Fancy", head_length=.2, head_width=.2,
99     tail_width=.1),color='black'))
100     """
101     PointWise_LowerBound = np.zeros(len(t1))
102     PointWise_UpperBound = np.zeros(len(t1))
103     for tind in range(len(t1)):
104         t = t1[tind]
105         Min = LinRegr(popt,t)
106         Max = LinRegr(popt,t)
107         for comb1, comb2 in combinations(product([-1,1],
108         repeat=len(stdDev)),2):

```

```

102         p1 = poprt+stdDev*comb1
103         p2 = poprt+stdDev*comb2
104         Min = min(LinRegr(p1, t), Min)
105         Max = max(LinRegr(p2, t), Max)
106         PointWise_LowerBound[tind] = Min
107         PointWise_UpperBound[tind] = Max
108     """
109     if shear_ind == 0:
110         ax.plot(t1, LinRegr(popt, t1), marker = 'None',
111                 linestyle = '-', color = colorlist[shear_ind],
112                 label=ylabel[:-1]+'=\begin{cases} c \leq c^*: & \& \\
113                 [\eta] \cdot c + xlabel[1:-1]+'+\eta\{c^*\}-[\eta] \\
114                 [\eta] \cdot c^* \\ c > c^*: & \& [\eta] \cdot \prime \cdot c \\
115                 +xlabel[1:-1]+'+\eta\{c^*\}-[\eta] \cdot \prime \cdot c^* \\
116                 \end{cases}$')
117         #ax.fill_between(t1, PointWise_LowerBound,
118                         PointWise_UpperBound, color=colorlist[shear_ind],
119                         alpha = .1, label = r'$1\sigma$-Fehlerstreifen$')
120     else:
121         ax.plot(t1, LinRegr(popt, t1), marker = 'None',
122                 linestyle = '-', color = colorlist[shear_ind])
123         #ax.fill_between(t1, PointWise_LowerBound,
124                         PointWise_UpperBound, color=colorlist[shear_ind],
125                         alpha = .1)
126     ax.errorbar(x=X, y=Y, xerr=Xerr, yerr=Yerr, marker='x',
127                color = colorlist[shear_ind], linestyle='None', label=
128                r'Messwerte für $\dot{\gamma}$'+str(round(df.iloc[
129                shear_ind, 7], 3))+'\mathrm{s}^{-1}$')
130     poprtList.append(popt)
131     stdList.append(stdDev)
132     gammadotList.append(df.iloc[shear_ind, 7])
133
134     cell_text = np.core.defchararray.add(np.core.defchararray.
135         add(np.round(np.transpose(poprtList)).astype(str), np.full
136         (np.transpose(poprtList).shape, '+')), np.round(np.
137         transpose(stdList)).astype(str))
138     #cell_text = np.core.defchararray.add(np.core.defchararray.
139         add(np.full(np.full(np.transpose(poprtList).shape, '$'),
140             cell_text)), np.full(np.transpose(poprtList).shape, '$'))
141
142     sigfig = 1
143     stdDevFlat = np.array(stdList).flatten()
144     poprtFlat = np.array(poprtList).flatten()
145

```

```

126 np.savetxt('c_star_by_gamma_dot.csv',[np.transpose(poptList
127 ) [0], np.transpose(stdList) [0], gammadotList], delimiter=',
128 ')
127
128 stdDev_rounded = ['{:g}'.format(float('{:.{p}g}'.format(
129     stdDevFlat[i], p=sigfig)))+ '$' for i in range(0, len(
130     stdDevFlat))]
129
130 decimals = [len(str(stdDev_rounded[i].split('.')[1])) for
131     i in range(0, len(stdDev_rounded))]
132
133 popt_rounded = [r'$'+str(round(poptFlat[i], decimals[i]))+' \
134     pm' for i in range(len(poptFlat))]
135
136 poptList = np.reshape(np.array(popt_rounded), np.array(
137     poptList).shape)
138
139 stdList = np.reshape(np.array(stdDev_rounded), np.array(
140     stdList).shape)
141
142 cell_text = np.core.defchararray.add(np.transpose(poptList)
143     , np.transpose(stdList).astype(str))
144
145 the_table = the_table = plt.table(cellText=cell_text ,
146     #rowLabels = np.full(7, 'a'),
147     rowLabels=[r'$c^*[\%_{\dot{c}}]$', r'$\frac{\eta_{c^*}}{\eta_{\dot{c}}}$', r'$\frac{\eta_{c^*}}{\eta_{\dot{c}}}$', r'$\frac{\eta_{c^*}}{\eta_{\dot{c}}}$', r'$\frac{\eta_{c^*}}{\eta_{\dot{c}}}$', r'$\frac{\eta_{c^*}}{\eta_{\dot{c}}}$', r'$\frac{\eta_{c^*}}{\eta_{\dot{c}}}$',
148     ],
149     colLabels=[r'$\dot{\gamma} = %.3f \mathrm{s}^{-1}$' % i for i in gammadotList], #['
150     Wert', 'Unsicherheit'],
151     loc='bottom',
152     cellLoc='center',
153     bbox=[-0.1, -0.4, 1.2, 0.3],
154     edges='closed')
155
156 the_table.auto_set_font_size(False)
157 the_table.set_fontsize(8)
158
159 ax.set_ylim(ylim[0], maxy)
160 ax.set_xlabel(xlabel, size=16)
161 ax.set_ylabel(ylabel, size=16)
162 ax.set_xlim(xlim[0], xlim[1])
163 ax.grid(visible=True, which='minor', color="grey",
164     linestyle='-', linewidth=.008, alpha=.2)
165
166 ax.legend(loc='best', fontsize=13)

```

```

154
155     #fig.tight_layout()
156
157     #-----Save Figure-----
158     plt.savefig("
        Viscosity_by_Concentration_Guar_all_Shearrates_without_1sigma
        .pdf",dpi=1200)
159
160 if __name__ == "__main__":
161     plot(['guar_00000_percent','guar_00025_percent','
        guar_00050_percent','guar_00100_percent','
        guar_00140_percent','guar_00200_percent','
        guar_00230_percent'])

```

B.5. Frequenzversuch

```

1  # -*- coding: utf-8 -*-
2  """
3  Created on Sat Dec  3 11:27:48 2022
4
5  @author: Jan-Philipp
6  """
7
8  import numpy as np #math functions, arrays
9  import matplotlib.pyplot as plt #visualizing
10 from scipy.interpolate import make_interp_spline
11 import matplotlib
12 import pandas as pd
13 from scipy.optimize import curve_fit
14
15 matplotlib.style.use('JaPh') #merely contains basic style
    information
16 plt.ioff()
17
18 def Fit(x,a,b,c,d):
19     return np.tanh(a*(x-b))*c+d
20
21 def plot(CSVNAME):
22     global popt1
23     PATH = CSVNAME + '.csv'
24     fig,ax=plt.subplots(1,1,figsize=(10,10/np.sqrt(2)))
25
26     df = pd.read_csv(PATH,sep=',',header=None)

```

```

27     Y1=df.iloc[:,10]
28     Y2=df.iloc[:,11]
29     X =df.iloc[:,6]
30     print(Y1,Y2,X)
31     X = np.log(X)
32     Y1 = np.log(Y1)
33     Y2 = np.log(Y2)
34
35     ylabel = r'$\mathrm{ln}\left(\frac{G^{\prime}}{\mathrm{Pa}}\right)\mathrm{,}\mathrm{ln}\left(\frac{G^{\prime\prime}}{\mathrm{Pa}}\right)$'
36     xlabel = r'$\mathrm{ln}\left(\frac{f}{\mathrm{Hz}}\right)$'
37
38     Ymax = np.max(np.array([Y1,Y2]),axis=0)
39     Ymin = np.min(np.array([Y1,Y2]),axis=0)
40
41     dx = (max(X)–min(X))/10
42     dy = (max(Ymax)–min(Ymin))/15
43     xlim = (min(X)–dx,max(X)+dx)
44     ylim = (min(Ymin)–dy,max(Ymax)+dy)
45
46     #X_Y1_Spline = make_interp_spline(X, Y1,k=3)
47     #X_Y2_Spline = make_interp_spline(X, Y2,k=3)
48
49     pop1,pcov1 = curve_fit(Fit,X,Y1,maxfev = 100000,p0
50                          =[0.79769266, 3.77394899, 5.19260498, 7.45366197])
51     pop2,pcov2 = curve_fit(Fit,X,Y2,maxfev = 100000,p0
52                          =[0.79769266, 3.77394899, 5.19260498, 7.45366197])
53
54     minind1 = np.argmin(np.round(np.abs(Fit(t1,*pop2)–Fit(t1,*
55                          pop1)),1))
56     minind2 = np.argmin(np.round(np.abs(Fit(t1,*pop2)–Fit(t1,*
57                          pop1)),4))
58
59     #ax.plot(t1,Fit(t1,*pop2),marker='None',linestyle='-',
60             color='red')
61     #ax.plot(t1,Fit(t1,*pop1),marker='None',linestyle='-',
62             color='blue')
63     #ax.plot(t1,X_Y1_Spline(t1),marker='None',color='black',
64             linestyle='--')

```



```

74
75
76     ax.set_ylim(ylim[0],ylim[1])
77     ax.set_xlabel(xlabel,size=16)
78     ax.set_ylabel(ylabel,size=16)
79     ax.set_xlim(xlim[0],xlim[1])
80     ax.grid(visible=True, which='minor', color="grey",
            linestyle='-',linewidth=.008, alpha=.2)
81
82     ax.legend(loc='best',fontsize=15)
83     fig.tight_layout()
84
85     #——Save Figure——
86     plt.savefig("Osc_Guaran.pdf",dpi=1200)
87
88     if __name__ == "__main__":
89         plot('Osc_Guaran')

```