HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

MASTERARBEIT

Herr
Linus Herrich-Schäffer, B. Eng.

Erstellung und Implementierung eines gekoppelten Advektions-Diffusionsmodells in Lagrange-Koordinaten

2023

Fakultät **Ingenieurwissenschaften**

MASTERARBEIT

Erstellung und Implementierung eines gekoppelten Advektions-Diffusionsmodells in Lagrange-Koordinaten

Autor:
Linus Herrich-Schäffer, B. Eng.

Studiengang:
Lasertechnik/Physikalische Technik

Seminargruppe:
LT20w1-M

Erstprüfer:
Prof. Dr. rer. nat. habil. Alexander Horn

Zweitprüfer:
Markus Olbrich, M. Sc.

Mittweida, Juni 2023

Bibliografische Angaben

Herrich-Schäffer, Linus: Erstellung und Implementierung eines gekoppelten Advektions- Diffusionsmodells in Lagrange-Koordinaten, 129 Seiten, 34 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Ingenieurwissenschaften

Masterarbeit, 2023

Referat

In dieser Arbeit wird der Einfluss ultrakurzer Laserpulse auf die Dynamik thermodynamischer Zustandsgrößen in dünnen Gold- und Aluminiumschichten numerisch simuliert. Dazu wird zunächst ein Advektions-Diffusions-Modell in Euler-Koordinaten untersucht, welches eine Kopplung der hydrodynamischen Eulergleichungen mit einem Zwei-Temperatur-Modell darstellt. Anschließend wird ein äquivalentes Modell in Lagrange-Koordinaten entwickelt und die daraus berechneten Simulationen mit den Ergebnissen des Modells in Euler-Koordinaten verglichen. Um die thermodynamischen Materialeigenschaften von Aluminium und Gold zu modellieren, werden Zustandsgleichungen für beide Materialien entwickelt und in das Advektions-Diffusion-Modell integriert. Diese Zustandsgleichungen sind für die verschiedenen stabilen und metastabilen Aggregatzustände von Aluminium und Gold gültig.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungsverzeichnis

(G) ..... engl. gas

(L) ..... engl. liquid

(S) ..... engl. solid

α ..... Absorptionskoeffizient

$\triangle t$ ..... Länge des diskreten Zeitschrittes

$\triangle x$ ..... Länge des diskreten Ortsschrittes

$\dot{q}_{\mathrm{V}}$ ..... Volumenwärmequelle

ε ..... innere Energie

ε_{f} ..... Fermienergie

$\gamma(x,t)$ ..... Charakteristik eines Transportproblems

γ ..... Grüneisenparameter

$\gamma_{\mathrm{e}0}$ ..... Elektronischer Grüneisenparameter

γ_{th} ..... Sommerfeldparameter

γ_0 ..... Grüneisenparameter ohne Kompression

γ_∞ ..... Grüneisenparameter für den Grenzfall großer Kompressionen

$\hbar$ ..... reduziertes Planckssches Wirkungsquantum

κ ..... Adiabatenexponent

λ ..... Wellenlänge

λ_{e} ..... Wärmeleitfähigkeit des Elektronensystems

λ_{th} ..... Wärmeleitfähigkeit

Ω ..... Phononenfrequenz

Ω_D ..... Debye-Frequenz

ϕ ..... Bindungspotential

ρ ..... Dichte

ρ_{crit} ..... Kritische Dichte

ρ_{r} ..... Reduzierte Dichte

σ ..... Kompression

σ_{CFL} ..... CFL-Zahl

τ_{H} ..... Pulsdauer

Θ_{D} ..... Debye-Temperatur

$\vec{r}_j(t)$ ..... Allgemeine Bahnkurve eines Lagrange-Punktes

A ..... Advektionsgröße

a ..... Temperaturleitfähigkeit

B ..... Kompressionsmodul

B_{c0} ..... Kompressionsmodul bei 0 K ohne Kompression

B_{cp0} ..... Änderung des Kompressionsmoduls mit dem Druck bei 0 K ohne Kompression

C ..... Wärmekapazität

C_{e} ..... Wärmekapazität des Elektronensystems

c_{e} ..... spezifische Wärmekapazität des Elektronensystems

C_{ph} ..... Wärmekapazität des Phononensystems

c_{ph} ..... spezifische Wärmekapazität des Phononensystems

C_s ..... Schallgeschwindigkeit

D ..... Zustandsdichte

E ..... Energie

E_n ..... Energieniveaus des quantenmechanischen Oszillators

F ..... Freie Helmholtzenergie

F_{a}^{L} ..... Freien Helmholtzenergie des Atomgitters in der Flüssigphase

F_{m} ..... Korrekturterm für die freie Helmholtzenergie der Flüssigphase

F_{t} ..... Phononenkomponente der freien Helmholtzenergie der Flüssigphase

F_j^n ..... Zustandsgröße auf dem diskreten Ortsschritt j zum diskreten Zeitschritt n

F_{e} ..... Freie Helmholtzenergie des Elektronensystems

F_{e} ..... Freie Helmholtzenergie des Phononensystems

Fo ..... Fourrier-Zahl

G ..... Kopplungsfaktor

h ..... Plancksches Wirkungsquantum

I_0 ..... Spitzenintzensität

I_0 ..... Spitzenintensität

k ..... Extinktionskoeffizient

k_{B} ..... Boltzmann-Konstante

m ..... Masse

m_{e}^{*} ..... effektive Masse der Elektronen

m_{e} ..... Masse der freien Elektronen

N ..... Anzahl der diskreten Ortsschritte

p ..... Druck

p_{crit} ..... Kritischer Druck

p_{maxwell} ..... konstanter Druck aus Maxwell-Konstruktion

p_r ..... Reduzierter Druck

p_c ..... Druckkomponente des Bindungspotentials

p_e ..... Elektronendruck

p_m ..... Schmelzdruck

p_ph ..... Phononendruck

q ..... Volumenbezogene Wärmeenergie

R ..... universelle Gaskonstante

S ..... Entropie

T ..... Temperatur

t ..... Zeitkoordinate

T_crit ..... Kritische Temperatur

T_e ..... Elektronentemperatur

T_ph ..... Phononentemperatur

T_r ..... Reduzierte Temperatur

T_m0 ..... Schmelztemperatur bei Normaldruck

u ..... Geschwindigkeit

u_f ..... Fermigeschwindigkeit

v ..... spezifisches Volumen

v_0 ..... Spezifisches Volumen bei Normalbedingungen

v_crit ..... Kritisches Volumen

v_r ..... Reduziertes Volumen

v_0c ..... Spezifisches Volumen bei 0 K

x ..... Ortskoordinate

$x_j(t)$ ..... Eindimensionale Bahnkurve eines Lagrange-Punktes

Al ..... Aluminium

Au ..... Gold

bin ..... Binodale

DOS ..... engl. density of states

M ..... Molare Masse

spin ..... Spinodale

1 Einleitung

Die Entwicklung von Ultrakurzpulslasern mit Pulsdauern im Femto- bis niedrigen Pikosekundenbereich eröffnete im letzten Jahrzehnt eine Vielzahl neuartiger Anwendungsmöglichkeiten im Bereich der mikroskopischen Materialbearbeitung [1][2][3][4]. So bewirken die hohen Spitzenintensitäten zusammen mit den kurzen Pulsdauern einen zeitlich konzentrierteren Energieeintrag mit großen Energiedichten in das Material, was bei Bestrahlung mit kurzen Laserpulsen im hohen ps bis ns- Bereich nicht der Fall ist. Bei Metallbearbeitungsprozessen, wie beispielsweise dem Feinschneiden [4] oder auch der 3D-Mikrostrukturierung durch schichtweises Abtragen [3] führt die Erhöhung der Spitzenintensität bei gleichzeitiger Verkürzung des Energieeintrags zu deutlich höheren Abtragsraten und einer signifikanten Verkleinerung der Wärmeeinflusszonen. Das Ergebnis sind Bearbeitungsergebnisse hoher Präzision und Qualität.

Um dies in der Praxis umzusetzen, ist die Verwendung von Laserstrahlung mit geeignet ausgewählten Parametern wie Pulsdauer, Wellenlänge und Fluenz sowie die Kenntnis der zu erwartenden Materialantworten, beispielsweise der thermischen Einflusszonen oder auch der Abtragsrate bzw. -tiefe notwendig. Zu diesem Zweck ist die Parameterabhängigkeit der genannten Materialantworten Gegenstand experimenteller Untersuchungen, so beispielsweise in [5][6].

Auf der anderen Seite sind infolge dieser Entwicklung auch grundlagenphysikalische theoretische Untersuchungen im Bereich der Laserstrahlung-Materie-Wechselwirkung in den Fokus des Interesses gerückt. Genaue Kenntnisse der in Festkörpern stattfindenden Prozesse bei der Wechselwirkung mit Laserstrahlung erlauben die Eingrenzung von Prozessparametern im Vorfeld der experimentellen Untersuchungen, sowie die anschließende Verifizierung der Ergebnisse. Aus diesem Grund stellen ebenfalls zahlreiche Veröffentlichungen, beispielsweise [7][8][9][10], die theoretische Untersuchung und Beschreibung dieser auftretenden physikalischen Effekte dar.

Die bei der Laserstrahlung-Metall-Wechselwirkung konkret zu betrachtenden Prozesse sind zum einen die Advektion der Zustandsgrößen Dichte, Energie und Impuls und zum anderen die Diffusion, die eine zusätzliche Dynamik der inneren Energie in Elektronengas und Ionengitter bewirkt und sich bei makroskopischer Betrachtung in der Wärmeleitung des Materials zeigt. Der Begriff Advektion (bzw. Transport) beschreibt dabei die Bewegungsdynamik einer Zustandsgröße aufgrund eines gegebenen Geschwindigkeitsfeldes, wobei die Bewegung dessen Feldlinien folgt. Da die Ursache des Geschwindigkeitsfeldes wiederum in den Einflüssen anderer Zustandsgrößen liegt, kann die Advektion auch als Bewegungsdynamik einer Zustandsgröße infolge des Einflusses anderer Zustandsgrößen aufgefasst werden. Beispielsweise bewirkt ein Druckgradient innerhalb des Festkörpers die Induzierung eines Impulses. Dieser kann als Geschwin-

digkeitsfeld betrachtet werden, entlang dessen ein Massetransport stattfindet. Diffusion dagegen bezeichnet die Dynamik einer Zustandsgröße dessen Ursache im Gradienten der selben Größe liegt.

Gegenstand dieser Arbeit ist die Simulation der Dynamik von Zustandsgrößen innerhalb einer dünnen Metallschicht, deren Tiefe bzw. deren Oberflächennormale in die Schicht hinein den eindimensionalen Definitionsbereich der Simulation bildet. Der Laserpuls, dessen Pulsdauer im fs-Bereich liegt, trifft senkrecht auf diese Schicht. Die Simulation soll die orts- und zeitabhängige Temperaturverteilung innerhalb der Schicht infolge des Energieeintrages darstellen, wobei ein wesentliches Kriterium die separate Darstellung von Elektronen- und Phononentemperatur des Metalls sein wird. Insbesondere soll dabei die Kopplung von Elektronen- und Phononensystem hinsichtlich ihrer Temperatur im ps-Bereich simuliert werden. Desweiteren soll die Zeitentwicklung der Zustandsgrößen Dichte, Energie, Druck und Geschwindigkeit infolge der in der Schicht angeregten Transportprozesse aus dem Modell berechnet und dargestellt werden. Genauer betrachtet wird an dieser Stelle die Ausbildung einer Dichtewelle, die durch den Energieeintrag des Laserpulses angeregt wird und den Festkörper infolge der Advektion im ps-Zeitbereich nach der Anregung durchläuft.

Die Entwicklung und Lösung physikalischer Modellierungen kann in zwei unterschiedlichen Betrachtungsweisen erfolgen. Die erste stellt die Betrachtung des Modells in Euler-Koordinaten dar, bei dem das Modell bezüglich einer zeitinvarianten Koordinatenbasis entwickelt und demzufolge auf einem stationären Diskretisierungsgitter gelöst wird. Dem gegenüber steht die Lagrange'sche Betrachtungsweise, oder auch Betrachtung in Lagrange-Koordinaten. Diese basiert auf Materialpunkten, welche bei Transportprozessen ihre Positionen auf dem Material behalten und damit eine zeitvariante Basis darstellen. Der Übergang in Lagrange-Koordinaten stellt bei der Simulation von fluiddynamischen sowie feskörpermechanischen Prozessen ein häufig gewählter Ansatz dar. Dies liegt darin begründet, dass sich die Beschreibung von Zustandsänderungen auf Koordinatenpunkten deutlich vereinfacht, wenn die Koordinatenpunkte der Advektionsdynamik des betrachteten Fluids bzw. Festkörpers folgen.

Ein gekoppeltes Advektions-Diffusionsmodell in Euler-Koordinaten zur Beschreibung der laserinduzierten Zustandsdynamik für den Fall der oben beschriebenen Schicht liegt in [11] bereits als Impementierung in Python vor. Dort wird das konkrete Beispiel einer dünnen Goldschicht betrachtet. Die zugrunde liegenden physikalischen Modellierungen, sowie dessen Ergebnisse werden in Abschnitt 3 erläutert.

Ziel des ersten Teils dieser Arbeit ist aufbauend darauf die Formulierung eines physikalisch äquivalenten Modells in Lagrange-Koordinaten. Die damit berechneten Ergebnisse werden anschließend mit den Ergebnissen des Modells in Euler-Koordinaten verglichen.

Sowohl die Modellierung der Advektion in Euler- als auch in Lagrange-Koordinaten macht die Formulierung einer Zustandsgleichung für die Abhängigkeit des Druckes von der Temperatur und der Dichte notwendig. Die Zustandsgleichung muss dafür für einen sehr großen Parameterbereich und verschiedene Aggregatzustände gültig sein, was durch die korrekte Beschreibung von verschiedenen thermophysikalischen Eigenschaften des Materials in Abhängigkeit von der Temperatur und der Dichte ausgedrückt wird. Eben diese Abhängigkeiten der thermophysikalischen Eigenschaften können dabei zur Aufstellung und Bestimmung der Parameter der Zustandsgleichung verwendet werden.

Als stark vereinfachte Modellierung kann an dieser stelle zwar die Zustandsgleichung des idealen Gases in das Advektionsmodell eingesetzt werden, diese gibt jedoch die thermodynamischen Eigenschaften des bei der Simulation betrachteten Materials nicht hinreichend genau wieder. Ziel des zweiten Teils dieser Arbeit ist aus diesem Grund die Zusammenstellung einer thermodynamischen Zustandsgleichung für verschiedene Aggregatzustände sowie die Bestimmung der darin auftretenden Parameter auf Grundlage theoretischer Zusammenhänge und experimentell gewonnener Daten. Dies erfolgt zunächst am Beispiel von Aluminium, da für dieses Material bereits Vergleichswerte für den dichte- und temperaturabhängigen Druck vorliegen, mit denen die Ergebnisse verglichen werden können. Anschließend werden die entwickelten Zustandsgleichungen und Parameterapproximationen auf das Materialbeispiel Gold angewendet.

2 Physikalische Grundlagen

2.1 Eulergleichungen

Als Advektions- oder auch Transportprozess wird die Bewegung der Dichteverteilung einer physikalischen Größe entlang eines gegebenen Geschwindigkeitsfeldes u bezeichnet. Die Modellierung des Erhaltungssatzes für die Dichte A einer betrachteten Transportgröße führt im Allgemeinen zu einer Gleichung der Form [12]

$$\frac{\partial A}{\partial t} + \frac{\partial uA}{\partial x} = 0, \tag{1}$$

die als Advektionsgleichung in expliziter Erhaltungsform [12] bezeichnet wird (ausführliche Herleitungen finden sich in [11][13]) und den reinen Advektionsprozess d. h. ohne den Einfluss anderer physikalischer Größen beschreibt. Tritt eine zusätzliche Anregung des Transportprozesses aufgrund äußerer Einflüsse auf, resultiert diese in einem Quellterm f womit sich die inhomogene Advektionsgleichung in der Form

$$\frac{\partial A}{\partial t} + \frac{\partial uA}{\partial x} = f \tag{2}$$

schreiben lässt [13]. Die in diesem Abschnitt konkret betrachteten Transportgrößen sind die Masse, der Impuls und die Energie, deren Transportdynamiken über drei gekoppelte Advektionsgleichungen, Eulergleichungen genannt, verknüpft werden. Die erste Eulergleichung beschreibt analog zu Gl. 1 den Transport der Massendichte ρ in expliziter Erhaltungsform mit [11][14]

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} = 0. \tag{3}$$

Die Berechnung der Transportgeschwindigkeit u erfolgt aus der Impulstransportgleichung, welche die zweite Eulergleichung darstellt. Die Impulsdichte ist dabei die Massendichte multipliziert mit der Geschwindigkeit $\rho_I = u\rho$. Da der Impuls ebenfalls eine Erhaltungsgröße darstellt, kann dessen Transportdynamik analog zu Gl. 2 mit

$$\frac{\partial \rho u}{\partial t} + \frac{\partial \rho uu}{\partial x} = -\frac{\partial p}{\partial x} \tag{4}$$

formuliert werden. Der Quellterm ergibt sich daraus, dass ein negativer Druckgradient eine zusätzliche Anregung des Impulstransportes bewirkt [11][14], weshalb dies eine inhomogene Advektionsgleichung darstellt.

Die dritte Eulergleichung modelliert den Transportprozess der Energiedichte $\varepsilon\rho$, die sich aus der spezifischen inneren Energie ε des Festkörpers und der Massendichte

zusammensetzt mit

$$\frac{\partial \rho \varepsilon}{\partial t} + \frac{\partial \rho \varepsilon u}{\partial x} = -p \frac{\partial u}{\partial x}, \tag{5}$$

wobei der Term auf der rechten Seite die verrichtete Volumenarbeit als negative Quelle d. h. Senke der Energieadvektion repräsentiert (eine ausführliche Herleitung wird in [11] dargestellt).

2.2 Zwei-Temperatur-Modell

In diesem Abschnitt wird die energetische Wechselwirkung der Laserstrahlung mit dem Festkörper modelliert. Die in dieser Arbeit betrachteten Materialien Aluminium und Gold stellen metallische Festkörper dar und setzen sich demnach aus einem Ionengitter und dem delokalisierten Elektronengas zusammen. Diese mikroskopische Beschaffenheit wird bei der Modellbildung in Form eines Zwei-Temperatur-Modells berücksichtigt, welches eine explizite Unterscheidung zwischen dem Elektronensystem und dem Phononensystem vornimmt. Die Wechselwirkung der Laserstrahlung mit dem Metall führt zunächst zu einer lokal begrenzten energetischen Anregung des Elektronensystems innerhalb der optischen Eindringtiefte. Genauer betrachtet wird hierbei nur ein Teil der Elektronen angeregt, so dass ein lokales thermisches Ungleichgewicht innerhalb des Elektronensystems induziert wird. Durch Kollisionen geben die thermalisierten Elektronen jedoch einen Teil ihrer Energie an unangeregte Elektronen desselben Raumbereichs ab, so dass sich dort ein neues thermisches Gleichgewicht einstellt. Im Vergleich zum Zeitpunkt vor der Anregung hat sich die Elektronentemperatur in diesem Raumbereich erhöht. Desweiteren finden Stöße mit Elektronen angrenzender Raumbereiche und somit eine Energiediffusion statt, die als Wärmeleitung des Elektronensystems bezeichnet wird. Da das Elektronengas im Ionengitter eingelagert ist, werden Elektronen zudem an den Rümpfen inelastisch gestreut, was einen Energieübertrag an das Gitter und somit eine Anregung von Phononen bedeutet. Aufgrund des großen Masseunterschiedes zwischen Elektronen und Ionen wird bei jedem dieser Streuprozesse jedoch deutlich weniger Energie übertragen, als bei der Elektronen-Elektronen-Wechselwirkung, so dass sich die Energieabgabe an das Phononensystem erst in einem deutlich längeren Zeitraum von wenigen Pikosekunden in Form einer Angleichung von Elektronen- und Phononentemperatur bemerkbar macht.

Im Zwei-Temperatur-Modell werden Elektronen- und Phononensystem als energetisch gekoppelte, kontinuierliche Systeme angesehen. Für die in diesen Systemen enthaltene Wärmeenergien gelten ebenfalls die Erhaltungsgesetze, die zur Wärmeleitungsgleichung [13] der Form

$$C \frac{\partial T}{\partial t} - \frac{\partial T}{\partial x} \left(\lambda_{\text{th}} \frac{\partial T}{\partial x} \right) = f \tag{6}$$

führen. Diese wird separat zum einen für das Elektronen- und zum anderen für das Phononensystem formuliert [15][16]:

$$C_{\mathrm{e}}\frac{\partial T_{\mathrm{e}}}{\partial t}-\frac{\partial T_{\mathrm{e}}}{\partial x}\left(\lambda_{\mathrm{e}}\frac{\partial T_{\mathrm{e}}}{\partial x}\right)=-G(T_{\mathrm{e}}-T_{\mathrm{ph}})+\dot{q}_{\mathrm{V}} \tag{7}$$

und

$$C_{\mathrm{ph}}\frac{\partial T_{\mathrm{ph}}}{\partial t}=G\left(T_{\mathrm{e}}-T_{\mathrm{ph}}\right). \tag{8}$$

Der Quellterm der Elektronengleichung setzt sich aus dem Energieeintrag $\dot{q}_{\mathrm{V}}$ durch die Laserstrahlung und aus dem Energieabfluss an das Phononensystem (Senke) zusammen.

Die gepulste Laserstrahlung wird mit einer gaußförmige Intensitätsverteilung angenommen. Die Strahlungsintensität innerhalb der dünnen Schicht kann über das Lambert Beer'sche Gesetz bestimmt werden. Für die absorbierte Strahlungsenergiedichte in Abhängigkeit von der Schichttiefe x gilt somit

$$\dot{q}_V = I_0 \cdot e^{-4log(2)\left(\frac{t-t_0}{\tau_{\mathrm{H}}}\right)^2}(1-R)\alpha e^{-\alpha x}. \tag{9}$$

Der Absorptionskoeffizient α hängt über $\alpha=\frac{4k\pi}{\lambda}$ vom Extinktionskoeffizienten k und der Wellenlänge λ ab. Die Laserwellenlänge beträgt in allen nachfolgenden Betrachtungen $\lambda = 800$ nm und der Extinktionskoeffizient $k = 6,05$ für Au und $k = 1,89$ für Al [17]. R stellt hier den Reflektionsgrad dar, der über die Fresnel-Gleichungen aus den komplexen Brechungsindizes berechnet wird, in den nachfolgenden Simulationen jedoch mit $R = 0$ angenommen wird, so dass die Fluenz des Laserpulses der absorbierten Fluenz entspricht. Da die Energiediffusion innerhalb des Gitters im Allgemeinen sehr viel kleiner als im Elektronensystem ist, soll diese hier vernachlässigt werden, weshalb die Phononengleichung keine zweite Ortsableitung enthält.

Der Energieabfluss an des Phononensystem ist über den Kopplungsfaktor G proportional zur Temperasturdifferenz $T_{\mathrm{e}} - T_{\mathrm{ph}}$ [15][16]. Dieser Energieabfluss bildet wiederum den Quellterm der Phononengleichung. Der Kopplungsfaktor G kann bei hohen Elektronentemperaturen nicht mehr als konstant angenommen werden [16], vielmehr zeigt $G(T_{\mathrm{e}})$ die in Abb. 2.1 dargestellten Verläufe für Aluminium und Gold nach [16]. Sie wurden über polynomiale Approximationen in das Zwei-Temperatur-Modell integriert.

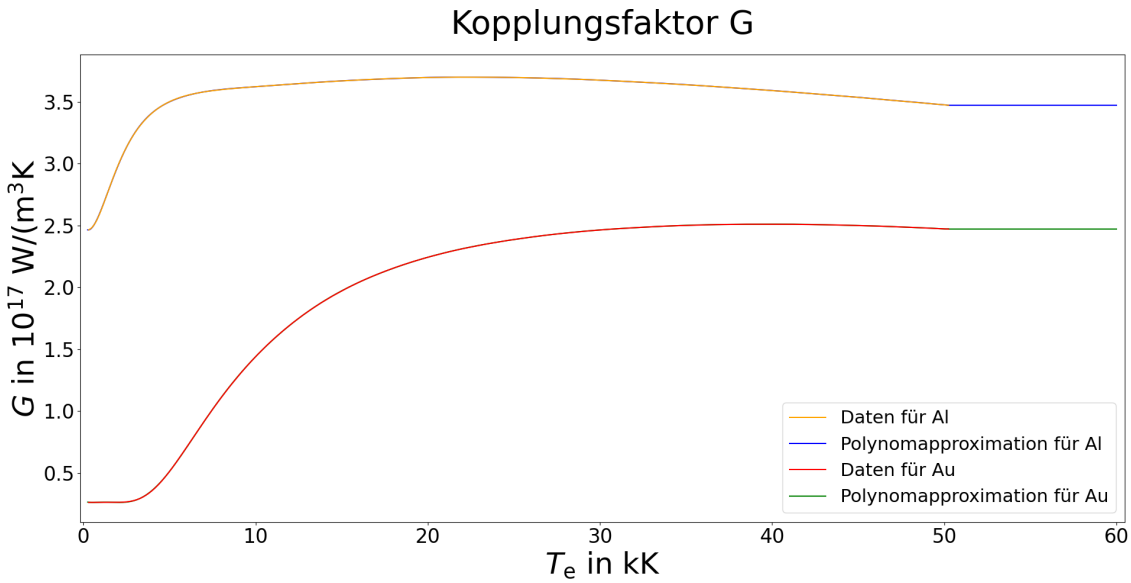

Abbildung 2.1: Kopplungskoeffizient für Aluminium und Gold als Funktion von der Elektronentemperatur nach [16]. Da die Daten aus [16] den Zusammenhang $G(T_e)$ nur bis zu einer Elektronentemperatur von 50 kK wiedergeben, wurden die Verläufe der Polynomapproximationen (blau, grün) mit konstanten Werten verlängert.

Grundlage der temperaturabhängigen Phononenenergie und damit der Wärmekapazität des Phononensystems bildet das Debye-Modell, welches von einer Separierung der kollektiven Gitterschwingungen, sog. Phononen, in harmonische quantenmechnanische Oszillatoren der Energie $E_n = \hbar\Omega(1/2+n)$ ausgeht [18]. $\hbar\Omega$ ist dabei die Energie eines einzelnen Phonons. Die Besetzungszahlen n der Phononen mit der Frequenz Ω sind temperaturabhängig und folgen der Bose-Einstein-Statistik

$$n(\Omega) = \frac{1}{e^{\frac{\hbar\Omega}{k_B T}} - 1}. \tag{10}$$

Die temperaturabhängige innere Energie der Phononen ergibt sich durch Integration über alle Frequenzen

$$E(T) = \int_0^{\Omega_D} \frac{\hbar\Omega}{e^{\frac{\hbar\Omega}{k_B T}} - 1} D(\Omega) d\Omega, \tag{11}$$

wobei Ω_D die höchste besetzte Frequenz darstellt und als Debye-Frequenz bezeichnet wird. Für die Zustandsdichte $D(\Omega)$ gilt $D(\Omega) \propto \Omega^2$ [18]. Die Temperaturableitung von $E(T)$ ergibt die molare Wärmekapazität. Das Integral ist analytisch nicht lösbar. Numerische Lösungen zeigen den in Abb. 2.2 dargestellten Verlauf für die Wärmekapazität, und der Grenzwert für hohe Temperaturen liefert den linearen Zusammenhang

$$E(T) = 3RT \tag{12}$$

mit der universellen Gaskonstante R. Dies führt zur konstanten molaren Wärmekapazität $C_{\text{ph}} = 3R = 24{,}9$ J/(mol K) nach Dulong-Petit [18]. Bei molaren Massen von $M = 196{,}97$ g/mol (Au) und $M = 26{,}98$ g/mol (Al) ergibt sich eine spezifische Wärmeka-

pazität von $c_{\mathrm{ph}} = 126,4$ J/(kg K) für Gold bzw. von $c_{\mathrm{ph}} = 922,91$ J/(kg K) für Aluminium. Die volumetrische Wärmekapazität ist dann jeweils $C_{\mathrm{ph}} = \rho c_{\mathrm{ph}}$.

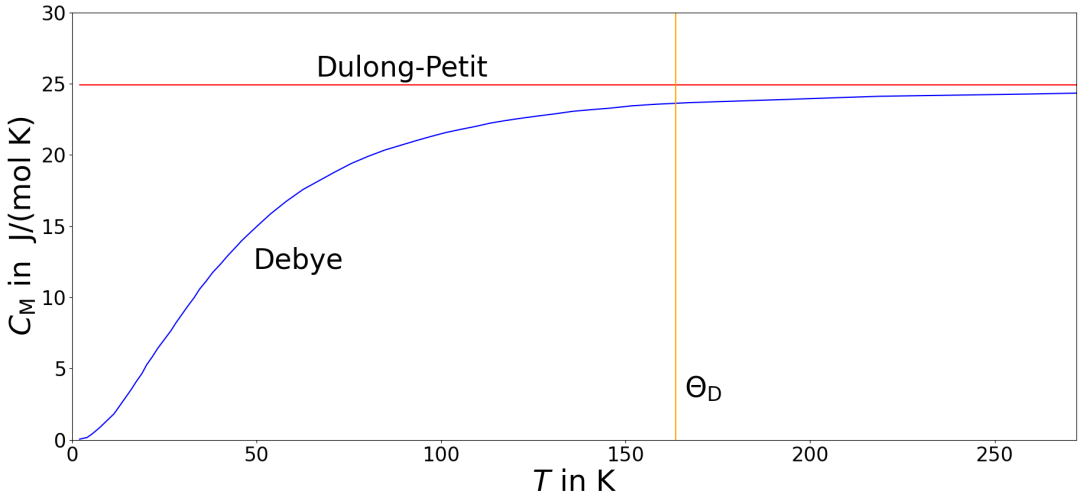

Abbildung 2.2: Molare Wärmekapazität des Phononensystems nach Debye und Dulong-Petit für das Materialbeispiel Gold. Θ_{D} stellt die Debye-Temperatur mit 165 K [19] dar.

Die Wärmekapazität C_{e} des Elektronensystems in Abhängigkeit von der Elektronentemperatur zeigt nach dem Sommerfeld-Modell des freien Elektronengases, welches in Abschnitt 8.1 betrachtet wird, einen linearen Verlauf. Für Gold und Aluminium befinden sich in [16] jedoch Daten für $C_{\mathrm{e}}(T_{\mathrm{e}})$, die auf Grundlage von DOS-Modellen, das bedeutet unter Berücksichtigung der energieabhängigen Zustandsdichte (DOS: engl. density of states) des Elektronengases [16] berechnet wurden. Die Datenpunkte geben die Wärmekapazität des Elektronensystems bis zu einer Elektronentemperatur von 50 kK wieder (siehe Abb. 2.3). Die Kurven $C_{\mathrm{e}}(T_{\mathrm{e}})$ werden für beide Materialien über polynomiale Approximationen aus diesen Datenpunkten gebildet und in das Zwei-Temperatur-Modell integriert.

Zuletzt wird noch die Wärmeleitfähigkeit des Elektronensystems bestimmt, die sich nach [15] über

$$\lambda_{\mathrm{e}} = K \frac{(\vartheta^2 + 0.16)^{5/4} (\vartheta^2 + 0.44)^2 \vartheta}{(\vartheta^2 + 0.092)^{1/2} (\vartheta^2 + b\vartheta_0)} \tag{13}$$

mit

$$\vartheta = \frac{k_{\mathrm{B}} T_{\mathrm{e}}}{\varepsilon_{\mathrm{f}}} \quad \text{sowie} \quad \vartheta_0 = \frac{k_{\mathrm{B}} T_0}{\varepsilon_{\mathrm{f}}}$$

berechnet, mit den in [15] dargestellten Materialkonstanten für Gold:

$$K = 353 \, \frac{\mathrm{W}}{\mathrm{mK}} \, , \quad b = 0,16 \quad \text{und} \quad u_{\mathrm{f}} = 1,39 \cdot 10^6 \, \frac{\mathrm{m}}{\mathrm{s}} \, .$$

Die Fermienergie ε_{f} setzt sich nach $\varepsilon_{\mathrm{f}} = \frac{1}{2} m_{\mathrm{e}} u_{\mathrm{f}}^2$ aus der Elektronenmasse und der

Fermigeschwindigkeit zusammen. Die elektronische Wärmeleitfähigkeit von Aluminium ist in [20] bis zu einer Temperatur von 50 kK dargestellt (vgl. Abb. 2.3). Bei höheren Elektronentemperaturen steigt λ_e nach Gl. 13 jedoch wieder an. Um diesen Verlauf für Al zu modellieren, werden die Parameter b und K aus Gl. 13 mit den Daten aus [20] approximiert (vgl. Abb. 2.3).

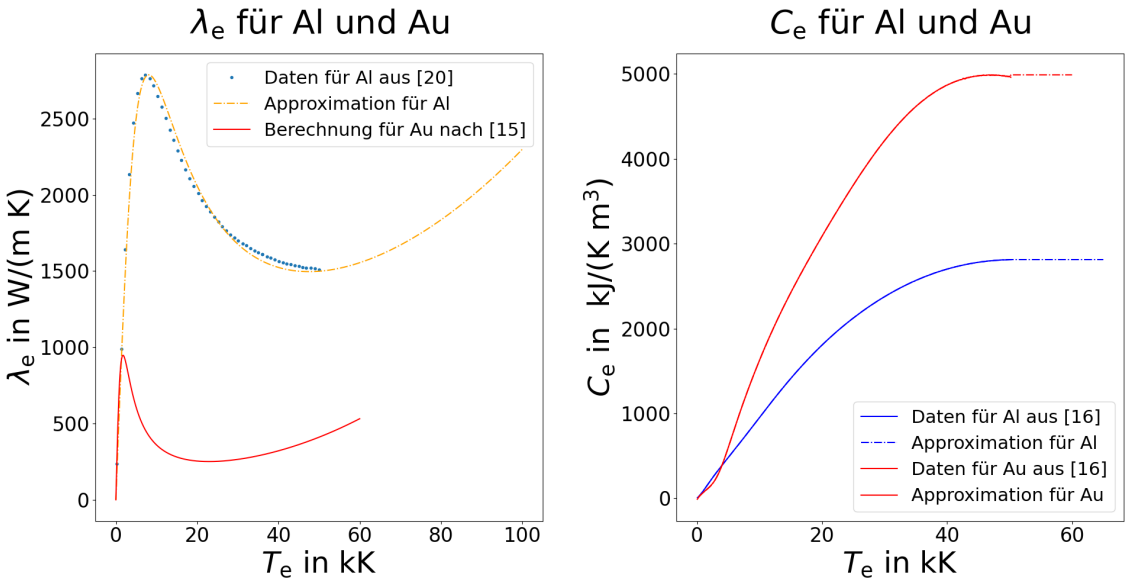

Abbildung 2.3: Wärmeleitkoeffizient und Wärmekapazität des Elektronensystems von Aluminium und Gold als Funktion der Elektronentemperatur.

2.3 Physikalische Betrachtungen in Euler- und Lagrange-Koordinaten

Bei der Eulerschen Betrachtungsweise wird eine statische, d. h. zeitinvariante Koordinatenbasis zugrundegelegt, auf welche sich die als Feldgleichungen bezeichneten Modellierungen mittels kontinuumsmechanischer Formulierungen, wie Masse-, Energie- und Impulserhaltung beziehen. Für die Lösung von Feldgleichungen mittels Finiter Differenzen wird ein Gitter aus Ortspunkten auf dem Definitionsbereich festgelegt, dessen Basis die zeitinvarianten Eulerkoordinaten bilden. Da die Basis zeitinvariant ist, gilt die Zeitinvarianz auch für das örtliche Diskretisierungsgitter.

Die Dynamik $\frac{\partial F_j}{\partial t}$ einer Zustandsgröße F ist durch die Feldgleichungen auf jedem der starren Gitterpunkte j gegeben. Damit kann der Wert F_j^n auf jedem Gitterpunkt j für jeden Zeitschritt n mittels numerischer Zeitintegrationsmethoden, wie dem expliziten Euler-Verfahren oder dem Runge-Kutta-Verfahren berechnet werden.

Dem gegenüber werden bei Betrachtungen in Lagrange-Koordinaten zunächst feste Punkte auf dem betrachteten Fluid bzw. Festkörper für den Anfangszeitpunkt t^0 defi-

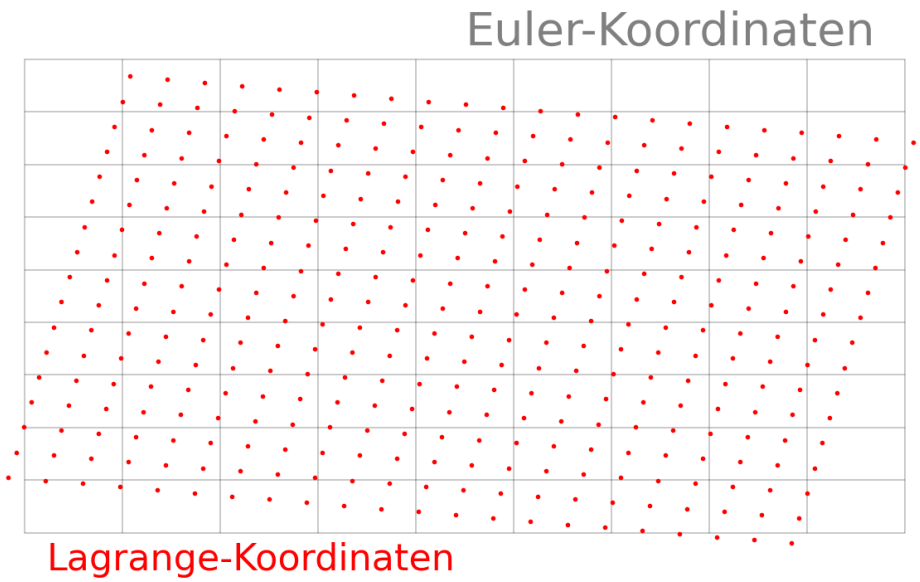

Abbildung 2.4: Beispielhafte Darstellung eines starren Gitters (Euler-Koordinaten, grau) zusammen mit einer beispielhaften Darstellung eines zweidimensionalen Festkörpers in Lagrange-Koordinaten (rot) für den Anfangszeitpunkt t^0. Bei Betrachtung späterer Zeitpunkte bleibt die Basis der Euler-Koordinaten konstant während die Lagrangepunkte bei Deformation des Festkörpers Translationen ausführen (vgl. Abb. 2.5).

niert, die im Folgenden Lagrange-Punkte genannt werden (vgl. Abb. 2.4). Diese werden auch als Materialpunkte bezeichnet, da sie auf dem Material „fixiert“ sind. Sie bilden die Koordinatenbasis physikalischer Formulierungen in der Lagrange'schen Betrachtungsweise. Findet infolge kontinuumsmechanischer Prozesse wie der Advektion eine Deformation des Festkörpers statt, resultiert diese in einer Translation der Lagrangepunkte bezüglich der stationären Euler-Koordinaten (vgl. Abb. 2.5). Grundlegend für Betrachtungen in Lagrange-Koordinaten ist demnach die Aufstellung von Bewegungsgleichungen für die Translation der Lagrangepunkte, deren Lösung in den Trajektorien $\vec{r}_j(t)$ aller Lagrangepunkte j resultiert.

Bei der Simulation physikalischer Prozesse wird schließlich die zeitliche Entwicklung der relevanten Zustandsgrößen F auf den instationären Lagrange-Punkten j formuliert. Für diese substanzielle Ableitung gilt

$$\frac{DF_j}{Dt} = \frac{\partial F}{\partial t} + \vec{u}\nabla F \tag{14}$$

wobei $\frac{\partial F_j}{\partial t}$ die lokale Ableitung darstellt, also die Zeitableitung von F an jenem Punkt bezüglich der Eulerkoordinaten, den der Lagrangepunkt zum Zeitpunkt t durchläuft. Physikalisch lässt sich dieser Ausdruck so interpretieren, dass advektive Terme in Feldgleichungen durch den Übergang von Euler- in Lagrangekoordinaten kompensiert werden. Damit enthält die Dynamik einer Zustandsgröße entlang der Lagrange-Punkte

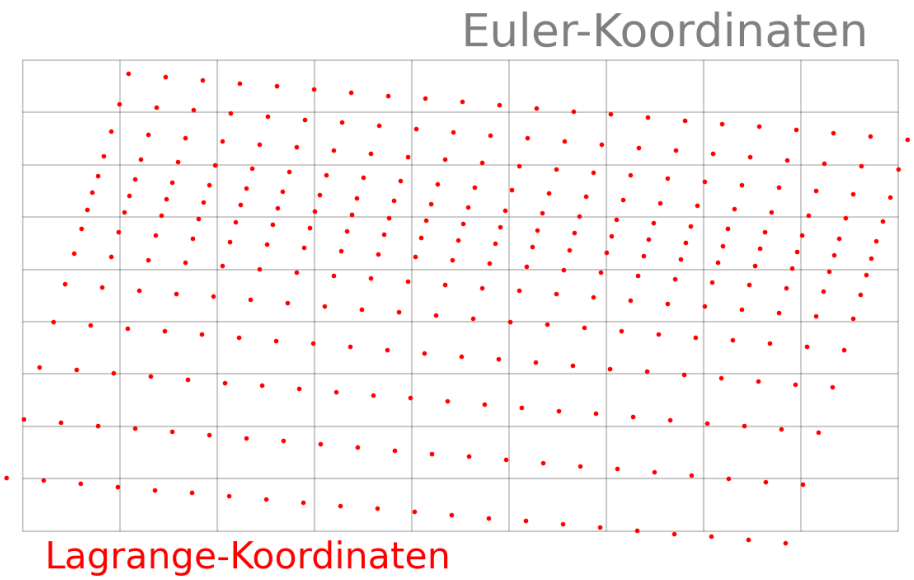

Abbildung 2.5: Darstellung der Euler-Koordinaten zusammen mit den Lagrangepunkten aus Abb. 2.4, jedoch für einen Zeitpunkt $t > t_0$. Der Festkörper zeigt eine beispielhafte Deformation in Gestalt einer Dichtewelle, die das Material infolge des Advektionsprozesses durchläuft. Diese bewirkt eine Translation der Lagrangepunkte. Verdichtungen und Verdünnungen des Materials sind anhand veränderter Abstände der Lagrange-Punkte zu erkennen.

keinen advektiven Anteil mehr, sondern wird nur noch von äußeren Einflüssen, wie Druckgradienten, oder im Falle der inneren Energie von der Diffusion beeinflusst. Im Rahmen der Entwicklung des Advektions-Diffusionsmodells in Lagrange-Koordinaten (Abschnitt 4) wird die Gültigkeit dieser allgemeinen Formulierung anhand einer anschaulicheren Betrachtung, der Methode der Charakteristiken bewiesen. Da ein erheblicher Anteil numerischer Herausforderungen mit der Lösung der in den Feldgleichungen enthaltenen Advektionsterme zusammenhängt, stellt der Übergang zu Lagrange-Koordinaten eine signifikante Vereinfachung numerischen Modelle sowie eine Stabilisierung numerischer Simulationen dar.

3 Darstellung des Advektions-Diffusionsmodells in Euler-Koordinaten

Die Simulation in Euler-Koordinaten erfolgt mittels Feldgleichungen, die auf Masse-, Energie- und Impulserhaltung basieren und den Advektions- und den Diffusionsprozess bezüglich einer zeitinvarianten Koordinatenbasis beschreiben. Das in diesem Abschnitt betrachtete Modell setzt sich zum einen aus den in Abschnitt 2 hergeleiteten Eulergleichungen (Gl. 3, 4, 5) und zum anderen aus dem Zwei-Temperatur-Modell zusammen. Für die numerische Lösung der Eulergleichungen wird in [11][12] auf einen Ansatz zurückgegriffen, der Operatorsplitting genannt wird. Die Eulergleichungen für Impuls und Energie (Gl. 4, 5) werden als inhomogene Differenzialgleichungen in jeweils zwei Gleichungen separiert, von denen die erste der homogenen DGL entspricht und damit eine Advektionsgleichung in expliziter Erhaltungsform darstellt. Diese beschreibt den Einfluss der reinen Advektion auf die Zeitentwicklung von Impuls bzw. Energie. Die zweite Gleichung formuliert dagegen den Einfluss der Quellterme von Gl. 4 und Gl. 5 auf die Zeitentwicklung der Zustandsgrößen. Numerisch zu lösen ist somit das Gleichungssystem:

reine Advektion der Dichte

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} = 0 \tag{15}$$

reine Advektion des Impulses

$$\frac{\partial \rho u}{\partial t} + \frac{\partial \rho u u}{\partial x} = 0 \tag{16}$$

Einfluss des Quellterms

$$\frac{\partial \rho u}{\partial t} = -\frac{\partial p}{\partial x} \tag{17}$$

reine Advektion der Energie

$$\frac{\partial \rho \varepsilon}{\partial t} + \frac{\partial \rho \varepsilon u}{\partial x} = 0 \tag{18}$$

Einfluss des Quellterms

$$\frac{\partial \rho \varepsilon}{\partial t} = -p\frac{\partial u}{\partial x} \tag{19}$$

Eine Zustandsgleichung $p(\rho, \varepsilon)$ schließt dabei das Gleichungsystem, wobei in diesem

Fall die Zustandsgleichung des idealen Gases

$$p = (\kappa - 1)\rho\varepsilon, \tag{20}$$

als einfachster Zusammenhang in das Modell integriert wird. κ ist in diesem Zusammenhang der Adiabatenexponent. Die Betrachtung der Energiediffusion erfordert die zusätzliche Lösung des Zwei-Temperatur-Modells. Die Energiegleichungen (Gl. 18, 19) werden dazu separat für die Elektronen- und die Phononenenergie gelöst und die beiden Energien über die spezifischen Wärmekapazitäten c_e und c_ph über $\varepsilon_\mathrm{ph} = c_\mathrm{e} T_\mathrm{e}$ bzw. $\varepsilon_\mathrm{ph} = c_\mathrm{ph} T_\mathrm{ph}$ in Temperaturen umgerechnet. Elektronen- und Phononentemperaturen werden schließlich in das Zwei-Temperatur-Modell

$$C_\mathrm{e}\frac{\partial T_\mathrm{e}}{\partial t} - \frac{\partial T_\mathrm{e}}{\partial x}\left(\lambda_\mathrm{e}\frac{\partial T_\mathrm{e}}{\partial x}\right) = -G(T_\mathrm{e} - T_\mathrm{ph}) + \dot{q}_V$$

$$C_\mathrm{ph}\frac{\partial T_\mathrm{ph}}{\partial t} = G\left(T_\mathrm{e} - T_\mathrm{ph}\right)$$

eingesetzt. Die Lösung dieses Advektions-Diffusionsmodells auf einem starren Gitter erfolgt mittels finiter Differenzen. Dies erfordert den Übergang der orts- und zeitabhängigen Größen in die diskrete Form:

$$\begin{array}{lll} \rho(x,t) \to \rho_j^n & u(x,t) \to u_j^n & \varepsilon(x,t) \to \varepsilon_j^n \\ p(x,t) \to p_j^n & T_\mathrm{e}(x,t) \to T_{\mathrm{e}j}^n & T_\mathrm{ph}(x,t) \to T_{\mathrm{ph}j}^n \end{array}$$

Detailierte Formulierungen der Diskretisierung sind in [11][12] beschrieben, wobei ersichtlich wird, dass der Hauptanteil des numerischen Aufwandes bei der Lösung bzw. Stabilisierung der Dichte-, Impuls- und Energieadvektionsgleichungen (Gl. 15, 16, 18) entsteht. Für diese sind komplexe Mittelungen auf einem Gitter mit versetzt angeordneten Größen notwendig, um numerische Fehler zu reduzieren und das unphysikalische Auftreten divergenter Oszillationen zu vermeiden [11][12]. Beim Übergang in Lagrange-Koordinaten im nächsten Abschnitt wird sich zeigen, dass durch die Wahl des beweglichen Koordinatensystems genau diese Advektionsgleichungen verschwinden und stattdessen die sehr viel einfachere numerische Lösung gewöhnlicher Differentialgleichungen genügt.

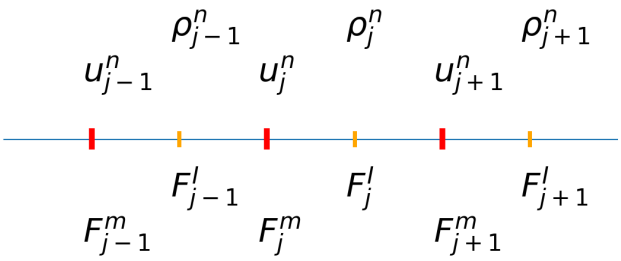

Abbildung 3.1: Starres, versetztes Diskretisierungsgitter nach [12].

Das Modell in Euler-Koordinaten liegt bereits als vollständige Implementierung in Python vor [11]. Die Ergebnisse dieser Implementierung (siehe Abb. 3.2, 3.3) werden bei der Bewertung des nachfolgend entwickelten Modells in Lagrange-Koordinaten als Referenz herangezogen.

Zunächst zeigt Abb. 3.2 die aus dieser Implementierung berechneten Elektronen- und Phononentemperaturen für eine Schichtdicke von 200 nm (Au) und eine Zeitdauer von 8 ps. Die Spitzenintensität des Laserpulses beträgt $10^{16}\ \mathrm{W/m^2}$ und die Pulsdauer 10 fs. Die elektronische Anregung ist durch den Anstieg der Temperatur auf einige kK in oberflächennahen Bereichen zu erkennen, während sich das Phononensystem bei Raumtemperatur befindet. Im Zeitbereich weniger ps macht sich schließlich die Elektronen-Phononen-Kopplung infolge der Streuprozesse durch den rapiden Abfall der Elektronen- und das ansteigen der Phononentemperatur bemerkbar. Desweiteren zeigt sich die Energiediffusion innerhalb des Elektronensystem durch einen verzögerten Anstieg der Elektronentemperatur in tieferen Schichtbereichen.

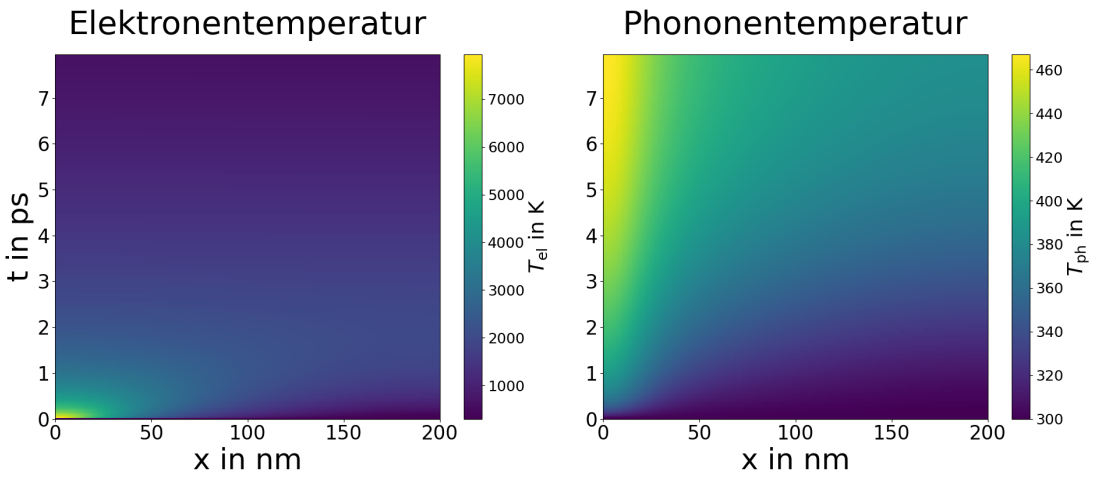

Abbildung 3.2: Lösungen des Advektions- Diffusionsmodells in Euler-Koordinaten für den Zeitbereich bis $t = 8$ ps für die Zustandsgrößen Elektronen- und Phononentemperatur.

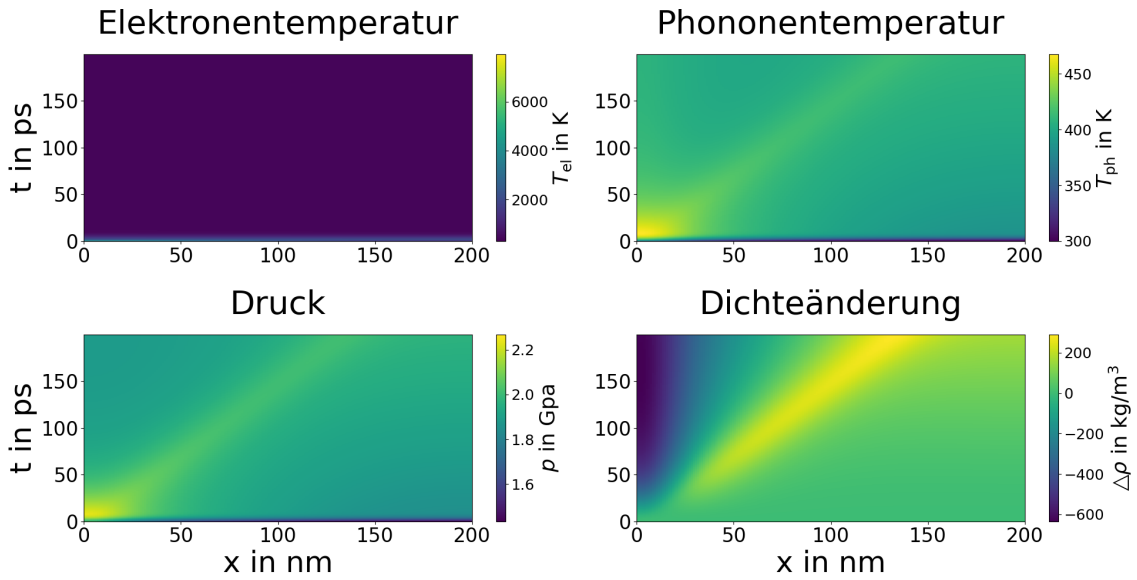

Abbildung 3.3: Lösungen des Advektions- Diffusionsmodells in Euler-Koordinaten für den Zeitbereich bis $t = 200$ ps für die Zustandsgrößen Elektronen- und Phononentemperatur, Druck und Dichte.

In Abb. 3.3 ist die Lösung des Modells für die Größen Elektronen, bzw. Phononentemperatur, Druck und Dichte in einem erweiterten Zeitbereich bis 200 ps abgebildet. Im Zeitbereich einiger ps bis wenige 10 ps bildet sich aufgrund der Erwärmung des Phononensystems ein steiler Druckgradient aus (links unten), der im weiteren Verlauf in einer Druckwelle resultiert, die den Festkörper durchläuft. Als Folge des durch den Druckgradienten angeregten Impulstransportes findet schließlich ein Massetransport statt, der sich in einer der Druckwelle folgenden Dichtewelle (Stoßwelle) manifestiert. Dargestellt ist hierbei (rechts unten) die Differenz $\triangle\rho$ zur Dichte $\rho = 19320\ \mathrm{kg/m^3}$ von Gold bei Normalbedingungen.

4 Entwicklung des Advektions-Diffusionsmodells in Lagrange-Koordinaten

Die Formulierung des Advektions-Diffusionsmodells in Lagrange-Koordinaten erfordert zunächst die Festlegung von Lagrange-Punkten auf dem in einer Dimension betrachteten Material. Analog zum Modell aus Abschnitt 3 bildet den Definitionsbereich die Senkrechte durch eine dünne Goldschicht. Entlang dieser Achse werden äquidistante Lagrange-Punkte definiert, so dass sich jeweils zwischen zwei Punkten ein Materialelement definierter Masse befindet. Die Masse m_j der einzelnen Elemente wird auch bei Translation der Lagrange-Punkte als konstant angenommen, so dass sich die Materialdichte aus den Abständen der Punkte zueinander ergibt.

Die allgemeine Bahnkurve $\vec{r}_j(t)$ eines Lagrange-Punktes j kann bei bekannter Advektionsgeschwindigkeit durch Lösen der Bewegungsgleichung

$$\frac{\partial \vec{r}_j(t)}{\partial t} = \vec{u}_j(\vec{r}_j(t),t) \tag{21}$$

bestimmt werden. Die Zeitintegration über den Hauptsatz der Differential- und Integralrechnung ergibt

$$\vec{r}_j(t_1) = \vec{r}_j(t_0) + \int_{t_0}^{t_1} \vec{u}_j(\vec{r}_j(t),t)dt \tag{22}$$

bzw. entlang eines eindimensionalen Definitionsbereiches

$$x_j(t_1) = x_j(t_0) + \int_{t_0}^{t_1} u_j(x_j(t),t)dt. \tag{23}$$

Da die Masse zwischen den Lagrange-Punkten stets konstant ist, bleibt bei der Berechnung der Trajektorien die Masseerhaltung erfüllt und die Lösung der Dichteadvektion (Gl. 3) entfällt. Die Dichte ρ_j der Festkörperelemente kann für jeden Zeitpunkt aus den Positionen bzw. Abständen der Lagrange-Punkte mit $m_j = \rho_j(x_{j+1} - x_j)$ berechnet werden. Trotz der Betrachtung eines eindimensionalen Definitionsbereichs, wird die Dichte

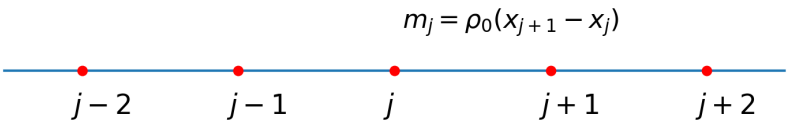

Abbildung 4.1: Schematische Darstellung der Lagrange-Punkte

als Masse pro Volumen mit $[\rho] = \mathrm{g/cm^3}$ definiert. Dies lässt sich durch die Auffassung des Definitionsbereiches als dünnen Stab mit infinitesimal kleiner Querschnittsfläche legitimieren. Die Geschwindigkeit $u_j(x_j(t),t)$ der Punkte sowie die in den Festkörperelementen enthaltenen inneren Energien können aus Impuls- und Energieerhaltung berechnet werden, die in Abschnitt 2 in Form der Eulergleichungen formuliert und in Abschnitt 3 in einen advektiven Anteil (Gl. 16, 18) und einen Anteil zur Beschreibung der äußeren Einwirkungen auf die Dynamik der Zustandsgrößen (Gl. 17, 19) separiert wurden. Während in Abschnitt 3 das vollständige Gleichungssystem auf einem stationären Gitter gelöst wurde, soll hier die Dynamik der Zustandsgrößen Impuls und Energie entlang der Trajektorien der Lagrange-Koordinaten berechnet werden. Bei diesem Koordinatenübergang verschwinden schließlich auch die Advektionsgleichungen von Impuls und Energie (Gl. 16, 18) und die Lösungen entlang der Trajektorien der Lagrange-Punkte genügen stattdessen gewöhnlichen Differenzialgleichungen. Dies soll im Folgenden am Beispiel der Energieadvektion bewiesen werden.

Zunächst wird das Lösungsverhalten der Energieadvektionsgleichung (Gl. 18) für den Spezialfall ortskonstanter Geschwindigkeiten betrachtet. Das Lösungsverhalten der entsprechenden Gleichung

$$\frac{\partial \rho\varepsilon(x,t)}{\partial t} + u(t)\frac{\partial \rho\varepsilon(x,t)}{\partial x} = 0 \tag{24}$$

kann durch Kurven im Orts-Zeitraum mit der Kurvenparametrisierung $x = \gamma(t)$ charakterisiert werden, entlang derer die Lösung $\rho\varepsilon(\gamma(t),t)$ konstant ist [13]. Sie werden Charakteristiken des Advektionsproblems genannt. Die Anfangswerte der Energie $\rho\varepsilon_0 = \rho\varepsilon(x,t=0)$ werden beim Advektionsprozess entlang dieser Kurven transportiert. Durch die Konstanz der Energie entlang der Charakteristiken folgt für deren Ableitung [13]

$$\frac{d}{dt}\left(\rho\varepsilon(\gamma(t),t)\right) = 0. \tag{25}$$

Durch Anwendung der mehrdimensionalen Kettenregel [21]

$$\frac{\partial}{\partial w_j} u(v_1(w_1,...,w_n),...,v_m(w_1,...,w_n)) = \frac{\partial u}{\partial v_k}\frac{\partial v_k}{\partial w_j} \tag{26}$$

führt die Ausführung der Zeitableitung von Gl. 25 zu

$$\frac{\partial}{\partial \gamma}\left(\rho\varepsilon(\gamma(t),t)\right)\frac{\partial}{\partial t}\gamma(t) + \frac{\partial}{\partial t}\left(\rho\varepsilon(\gamma(t),t)\right) = 0 \tag{27}$$

bzw.

$$\frac{\partial}{\partial x}\left(\rho\varepsilon(x,t)\right)\frac{\partial}{\partial t}\gamma(t) + \frac{\partial}{\partial t}\left(\rho\varepsilon(x,t)\right) = 0. \tag{28}$$

Durch Vergleich von Gl. 28 mit Gl. 24 ergibt sich

$$\frac{\partial}{\partial t}\gamma(t) = u(t) \tag{29}$$

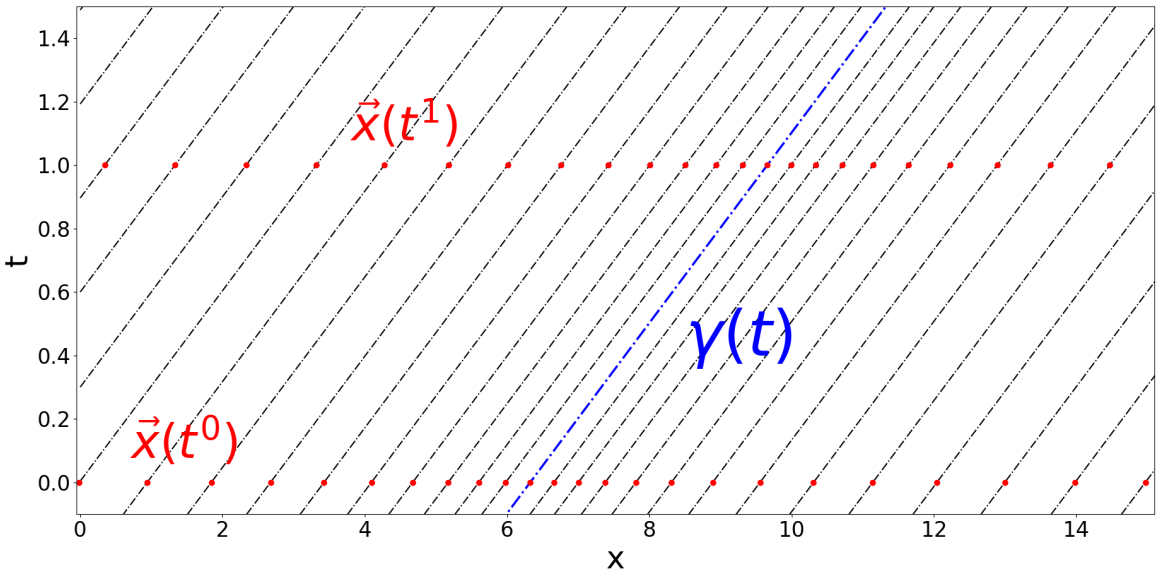

Abbildung 4.2: Beispielhafte Darstellung der Trajektorien von Lagrangepunkten nach Gl. 23 für den einfachsten Spezialfall konstanter Geschwindigkeit. Ebenfalls dargestellt sind die Charakteristiken $\gamma(t)$. Bei konstanter Geschwindigkeit u stellen diese nach Gl. 30 Geraden da, die den Lagrangepunkten folgen.

und durch Integration nach der Zeit

$$\gamma(t) = \gamma(t) + \int_{t_0}^{t_1} u(t)dt. \tag{30}$$

Der Vergleich von Gl. 30 mit Gl. 23 zeigt, dass die Charakteristiken der Energieadvektion den Trajektorien der Lagrange-Punkte für jede ortsunabhängige Geschwindigkeit $u(t)$ folgen. Da die spezifische Energie definitionsgemäß entlang der Charakteristiken konstant ist, ist sie somit auch entlang der Trajektorien der Lagrange-Punkte stets konstant, d. h. die zeitliche Änderung verschwindet, womit die Lösung der Impulsadvektionsgleichung (Gl. 25) entfällt.

Im allgemeineren Fall einer ortsabhängen Geschwindigkeit $u(x,t)$ wird die Energieadvektionsgleichung

$$\frac{\partial \rho\varepsilon(x,t)}{\partial t} + \frac{\partial \rho\varepsilon(x,t)u(x,t)}{\partial x} = 0 \tag{31}$$

durch Anwendung der eindimensionalen Kettenregel zu

$$\frac{\partial \rho\varepsilon(x,t)}{\partial t} + u(x,t)\frac{\partial \rho\varepsilon(x,t)}{\partial x} = -\rho\varepsilon(x,t)\frac{\partial u(x,t)}{\partial x}. \tag{32}$$

Gl. 32 unterscheidet sich von Gl. 24 also durch das Auftreten eines zusätzlichen Quellterms. In diesem Fall kann im Allgemeinen nicht mehr davon ausgegangen werden,

dass Kurven gefunden werden, entlang derer die Lösung konstant ist [13]. Dennoch können hier analog zu Gl. 29 Kurven mit der Parametrisierung $x = \gamma(t)$ definiert werden, die die gewöhnliche Differentialgleichung

$$\frac{d}{dt}\gamma(t) = u(t,\gamma(t)) \tag{33}$$

erfüllen [13]. Durch Integration

$$\gamma(t) = \gamma(t) + \int_{t_0}^{t_1} u(t,\gamma(t))dt \tag{34}$$

und Vergleich mit Gl. 23 ist ersichtlich, dass auch diese Kurven den Trajektorien der Lagrange-Punkte folgen. Sie werden hier deshalb ebenfalls als Charakteristiken bezeichnet. Die mehrdimensionale Kettenregel führt auch in diesem Fall wieder zu

$$\frac{d}{dt}\left(\rho\varepsilon(\gamma(t),t)\right) = \frac{\partial}{\partial\gamma}\left(\rho\varepsilon(\gamma(t),t)\right)\frac{\partial}{\partial t}\gamma(t) + \frac{\partial}{\partial t}\left(\rho\varepsilon(\gamma(t),t)\right). \tag{35}$$

Durch Einsetzen von Gl. 33 und Vergleich mit Gl. 32 ergibt sich schließlich

$$\frac{d}{dt}\left(\rho\varepsilon(\gamma(t),t)\right) = \frac{\partial}{\partial\gamma}\left(\rho\varepsilon(\gamma(t),t)\right)u(t,\gamma(t)) + \frac{\partial}{\partial t}\left(\rho\varepsilon(\gamma(t),t)\right) = -\rho\varepsilon\frac{\partial u}{\partial x}. \tag{36}$$

Während die Zeitänderung der Energiedichte entlang der Charakteristiken im obigen Fall ortskonstanter Geschwindigkeit verschwand, entspricht sie im Falle orts- und zeitvarianter Geschwindigkeiten also dem Quellterm mit

$$\frac{d}{dt}\left(\rho\varepsilon(\gamma(t),t)\right) = -\rho\varepsilon\frac{\partial u}{\partial x}. \tag{37}$$

Da die Charakteristiken $\gamma(t)$ definitionsgemäß den Trajektorien der Lagrange-Punkte $x(t)$ folgen, gilt für die Zeitentwicklung der Energiedichte entlang der Trajektorien der Lagrange-Punkte bezüglich der reinen Advektion die substantielle Ableitung

$$\frac{D}{Dt}\rho\varepsilon = -\rho\varepsilon\frac{\partial u}{\partial x}. \tag{38}$$

Dass dieser Ausdruck die Definition der substantiellen Ableitung nach Gl. 14 erfüllt, wird durch Einsetzen von Gl. 38 in den eindimensionalen Fall von Gl. 14 ersichtlich, was wieder Gl. 32 bzw. durch Anwendung der Kettenregel die Energieadvektionsgleichung (Gl. 31) ergibt.

Eine analoge Berechnung für die Advektion des spezifischen Impulses führt zur substanziellen Ableitung

$$\frac{D}{Dt}\rho u = -\rho u\frac{\partial u}{\partial x}. \tag{39}$$

Neben der reinen Advektion wirken zusätzlich noch der Druckgradient bzw. die Volumenarbeit auf die Zeitentwicklung von Impuls und Energie (Gl. 17, 19). Das vollständige Advektionsmodell lautet damit:

Trajektorien/Positionen der Lagrange-Punkte

$$x_j(t_1) = x_j(t_0) + \int_{t_0}^{t_1} u_j(x,t)dt \tag{40}$$

$$\text{mit} \quad m = \rho_j(x_{j+1} - x_j) = const.$$

Zeitentwicklung der Impulsdichte

$$\frac{D(\rho u)}{Dt} = -\rho u \frac{\partial u}{\partial x} - \frac{\partial p}{\partial x} \tag{41}$$

Zeitentwicklung der Energie

$$\frac{D(\rho \varepsilon)}{Dt} = -\rho \varepsilon \frac{\partial u}{\partial x} - p\frac{\partial u}{\partial x} \tag{42}$$

Die selben Ausdrücke ergeben sich durch Einsetzen der vollständigen Eulergleichungen für Impuls und Energie (Gl. 4, 5) in die Formulierung der substantiellen Ableitung (Gl. 23) für den eindimensionalen Fall. Im Gegensatz zu den Euler-Gleichungen, beschreibt dieses Modell die Zeitentwicklung der Zustandsgrößen nur noch über gewöhnliche Differenzialgleichungen (Gl. 41, 42), während numerisch aufwendige Advektionsterme nicht mehr enthalten sind. Das erwartete Resultat ist eine numerisch stabilere Lösung bei gleichzeitig geringerem Rechenaufwand.

5 Numerische Lösung des Advektionsproblems in Lagrange-Koordinaten

Das in Abschnitt 4 entwickelte Modell soll numerisch gelöst werden. Den ersten Schritt bildet die Berechnung der Trajektorien, d. h. die Positionen der Lagrange-Punkte für jeden der diskreten Zeitschritte t^n. Die Zeitintegration (Gl. 23) wird dazu mit dem expliziten Euler-Verfahren durchgeführt:

$$x(t^{n+1}) = x(t^n) + u(t^n) \triangle t \qquad \text{mit} \qquad \triangle t = t^{n+1} - t^n. \tag{43}$$

Die Position der Punkte für den neuen Zeitschritt $n+1$ lautet damit

$$x_j^{n+1} = x_j^n + u_j^n \triangle t. \tag{44}$$

Die Dichte eines Festkörperelements zum neuen Zeitschritt ergibt sich schließlich mit

$$\rho_j^{n+1} = \frac{m}{\triangle x_j^{n+1}} \tag{45}$$

$$\triangle x_j^{n+1} = x_{j+1}^{n+1} - x_j^{n+1} \tag{46}$$

aus der definierten Masse m eines Elementes sowie den Abständen der Lagrange-Punkte zueinander. Die Lagrange-Punkte zum neuen Zeitschritt fungieren nun als Diskretisierungsgitter zur Berechnung von Geschwindigkeit und Energie. Darum folgt der Übergang zu diskreten Größen u_j^n und ε_j^n. Die Geschwindigkeit zum neuen Zeitschritt $n+1$ wird mit Gl. 41 berechnet. Diese wird durch den Übergang

$$\frac{\partial u}{\partial x} \rightarrow \frac{u_{j+1} - u_{j-1}}{2 \triangle x_j}$$

$$\frac{\partial p}{\partial x} \rightarrow \frac{p_{j+1} - p_j}{\triangle x_j}$$

der Ortsableitung zu zentralen Differenzen mit

$$\frac{\partial \rho_j u_j}{\partial t} = -\rho_j u_j \frac{(u_{j+1} - u_{j-1})}{2 \triangle x_j} - \frac{(p_{j+1} - p_j)}{\triangle x_j} \tag{47}$$

ausgedrückt. Die Zentrierung des Differenzenquotienten des Druckes ergibt sich dadurch, dass dieser innerhalb der Elemente zwischen den einzelnen Lagrange-Punkten wirkt, die Geschwindigkeit bezieht sich jedoch auf den Lagrange-Punkt selbst. Damit stellt der Diffenenzenquotient eine Mittelung der Drücke rechts und links vom Lagrange-

Punkt j dar. Darauf folgt wieder die Zeitintegration

$$\rho_j^{n+1} u_j^{n+1} = \rho_j^n u_j^n - \int_{t^n}^{t^{n+1}} \left(\rho_j^n u_j^n \frac{(u_{j+1}^n - u_{j-1}^n)}{2 \triangle x_j} - \frac{(p_{j+1}^n - p_j^n)}{\triangle x_j} \right) dt \tag{48}$$

mit dem expliziten Eulerverfahren

$$u_j^{n+1} = \frac{\bar{\rho}_j^n u_j^n}{\bar{\rho}_j^{n+1}} - \triangle t \; \frac{\bar{\rho}_j^n u_j^n}{\bar{\rho}_j^{n+1}} \frac{(u_{j+1}^n - u_{j-1}^n)}{2 \triangle x_j} - \frac{(p_{j+1}^n - p_j^n)}{\triangle x_j}. \tag{49}$$

Da sich die Dichte ebenfalls auf die Elemente zwischen den Punkten bezieht, ist auch die Mittelung der Dichte $\bar{\rho}_j^n = (\rho_j^n + \rho_{j+1}^n)/2$ notwendig.

Zur Berechnung der Energie zum Zeitschritt $n+1$ wird Gl. 42 auf analoge Weise zu

$$\rho_j^{n+1} \varepsilon_j^{n+1} = \rho_j^n \varepsilon_j^n - \rho_j^n \varepsilon_j^n \triangle t \; \frac{(u_{j+1}^n - u_{j-1}^n)}{2 \triangle x_j} - \triangle t \; p_j^n \frac{(u_{j+1}^n - u_{j-1}^n)}{2 \triangle x_j}, \tag{50}$$

wobei sich der Druck aus der Zustandsgleichung des idealen Gases in diskretisierter Form nach

$$p_j^{n+1} = (\kappa - 1) \rho_j^{n+1} \varepsilon_j^{n+1} \tag{51}$$

ergibt. An den Grenzen sollen reflektierende Randbedingungen gelten, die auf den ersten bzw. letzten drei Zellen des Gitters mit

$$\begin{array}{ccc} u_2^n = 0 & \rho_1^n = \rho_2^n & \varepsilon_1^n = \varepsilon_2^n \\ u_1^n = -u_3^n & \rho_0^n = \rho_3^n & \varepsilon_0^n = \varepsilon_3^n \\ u_{N+2}^n = 0 & \rho_{N+2}^n = \rho_{N+1}^n & \varepsilon_{N+2}^n = \varepsilon_{N+1}^n \\ u_{N+3}^n = -u_{N+1}^n & u_{N+3}^n = -u_{N+1}^n & u_{N+3}^n = -u_{N+1}^n \end{array}$$

implementiert werden können. Bei Transportprozessen lautet das allgemeine Stabilitätskriterium $\sigma_{\mathrm{CFL}} = u \dfrac{\triangle t}{\triangle x} \leq 1$ [11], wobei σ_{CFL} als CFL-Zahl bezeichnet wird. Die Zeitschritte $\triangle t$ sind bei der Berechnung stets so zu wählen, dass $\sigma_{\mathrm{CFL}} \leq 1$ erfüllt ist.

6 Numerische Lösung des Zwei-Temperatur-Modells

Die Ortsdiskretisierung von Gl. 16 erfolgt ebenfalls durch den Übergang

$$\frac{\partial T}{\partial x} \rightarrow \frac{T_{j+1} - T_j}{\triangle x_j}$$

der Ortsableitungen zu Differenzenquotienten. Durch zweimaliges Einsetzen in die beiden Ortsableitungen und anschließende Zeitintegration mit dem expliziten Euler-Verfahren geht Gl. 7 in die diskrete Form

$$T_{\mathrm{e}\,j}^{n+1} = T_{\mathrm{e}\,j}^{n} + \frac{\triangle t}{C_\mathrm{e}} \frac{1}{2\triangle x_j^2}[(\lambda_{j+1}^n - \lambda_j^n)(T_{\mathrm{e}\,j+1}^{n} - T_{\mathrm{e}\,j}^{n}) - (\lambda_j^n - \lambda_{j-1}^n)(T_{\mathrm{e}\,j}^{n} - T_{\mathrm{e}\,j-1}^{n})]$$

$$+ \frac{\triangle t}{C_\mathrm{e}} \left(\dot{q}_j^n - G_j^n (T_{\mathrm{e}\,j}^{n} - T_{\mathrm{e}\,j}^{n})\right) \tag{52}$$

über. Die Gleichung des Phononensystems (Gl. 8) enthält nur eine Zeitableitung und kann auf die gleiche Weise mit

$$T_{\mathrm{ph}\,j}^{n+1} = T_{\mathrm{ph}\,j}^{n} + \frac{\triangle t}{C_\mathrm{ph}} G_j^n (T_{\mathrm{ph}\,j}^{n} - T_{\mathrm{ph}\,j}^{n}) \tag{53}$$

formuliert werden. Ebenfalls muss die Wärmequelle infolge der Laserstrahlung für diskrete Orts- und Zeitschritte mit

$$\dot{q}_j^n = I_0 \cdot e^{-4\log(2)\left(\frac{t^n - t_0}{\tau_\mathrm{H}}\right)^2} \alpha e^{-\alpha x_j} \tag{54}$$

beachtet werden. Das Stabilitätskriterium für die numerische Berechnung von Diffusionsprozessen bezüglich der Diskretisierung wird über die Fourier-Zahl

$$Fo = \min\left(a \frac{\triangle t}{\triangle x_j^2}\right) \leq 0{,}5 \tag{55}$$

mit der Temperaturleitfähigkeit $a = \frac{\lambda_\mathrm{th}}{C}$ bestimmt. An den Rändern des Definitionsbereiches sollen adiabatische Randbedingungen gelten, da die an der Oberfläche abgeführte Wärme bei den betrachteten Zeitspannen im Bereich bis einige 10 ps vernachlässigbar ist. Mathematisch bedeutet dies, dass die Ortsbleitungen der Temperatur an den Rändern verschwinden, so dass die Randbedingungen auf den ersten bzw. letzten zwei Zellen des Gitters mit $T_{\mathrm{e}\,0}^n = T_{\mathrm{e}\,1}^n, T_{\mathrm{e}\,N}^n = T_{\mathrm{e}\,N-1}^n, T_{\mathrm{ph}\,0}^n = T_{\mathrm{ph}\,1}^n, T_{\mathrm{ph}\,N}^n = T_{\mathrm{ph}\,N-1}^n$ implementiert werden können.

7 Advektions- Diffusionsmodell

7.1 Kopplung der numerischen Teilmodelle

Das numerische Advektionsmodell soll mit den Wärmeleitungsgleichungen eine geschlossene Formulierung bilden. In jedem Zeitschritt n werden dabei zunächst wieder die Trajektorien der Lagrange-Punkte bzw. die Diche aus deren Abständen berechnet. Die betrachtete Dichte bezieht sich dabei auf den gesamten Festkörper, ohne dass explizit zwischen Elektronen- und Phononensystem unterschieden wird. Anschließend erfolgt die Berechnung der Geschwindigkeit. Auch der Impuls und damit die Geschwindigkeit bezieht sich auf den gesamten Festkörper. Die Energiegleichung (Gl. 50) muss dagegen separat einmal für das Elektronensystem und einmal für das Phonensystem gelöst werden, da auch das Zwei-Temperatur-Modell zwischen den Temperaturen und damit den Energien beider Systeme unterscheidet. Zu beachten ist hier, dass die Energiedichte $\varepsilon\rho$ aus Gl. 50 der volumenbezogene Wärmeenergie q entspricht. Damit kann diese als

$$q_{\mathrm{e}\ j}^{n+1} = q_{\mathrm{e}\ j}^{n} - q_{\mathrm{e}\ j}^{n} \triangle t \ \frac{(u_{j+1}^{n} - u_{j-1}^{n})}{2 \triangle x_j} \ - \triangle t \ p_{\mathrm{e}\ j}^{n} \frac{(u_{j+1}^{n} - u_{j-1}^{n})}{2 \triangle x_j} \tag{56}$$

$$q_{\mathrm{ph}\ j}^{n+1} = q_{\mathrm{ph}\ j}^{n} - q_{\mathrm{ph}\ j}^{n} \triangle t \ \frac{(u_{j+1}^{n} - u_{j-1}^{n})}{2 \triangle x_j} \ - \triangle t \ p_{\mathrm{ph}\ j}^{n} \frac{(u_{j+1}^{n} - u_{j-1}^{n})}{2 \triangle x_j} \tag{57}$$

für das Elektronen- bzw. Phononensystem formuliert werden. Daraus leiten sich die Temperaturzwischenschritte über die Wärmekapazitäten mit

$$T_{\mathrm{e}\ j}^{1} = \frac{q_{\mathrm{e}\ j}^{n+1}}{C_{\mathrm{e}\ j}} \tag{58}$$

$$T_{\mathrm{ph}\ j}^{1} = \frac{q_{\mathrm{ph}\ j}^{n+1}}{C_{\mathrm{ph}}} \tag{59}$$

ab. Diese werden für die alten Zeitschritte der Temperaturen $T_{\mathrm{e}j}^{n}$ bzw. $T_{\mathrm{ph}j}^{n}$ in Gl. 52 und 53 eingesetzt, so dass sich für die Temperaturen zum neuen Zeitschritt

$$T_{\mathrm{e}\ j}^{n+1} = T_{\mathrm{e}\ j}^{1} + \frac{1}{C_{\mathrm{e}\ j}} \frac{\triangle t}{2 \triangle x^2} [(\lambda_{j+1}^{n} \ - \lambda_{j}^{n})(T_{\mathrm{e}\ j+1}^{1} \ - T_{\mathrm{e}\ j}^{1}) - (\lambda_{j}^{n} \ - \lambda_{j-1}^{n})(T_{\mathrm{e}\ j}^{1} \ - T_{\mathrm{e}\ j-1}^{1})]$$

$$+ \frac{\triangle t}{C_{\mathrm{e}\ j}} \left(\dot{q}_{j}^{n} - G_{j}^{n} (T_{\mathrm{e}\ j}^{1} - T_{\mathrm{ph}\ j}^{1}) \right) \tag{60}$$

sowie

$$T_{\mathrm{ph}\ j}^{n+1} = T_{\mathrm{ph}\ j}^{1} \ + \frac{\triangle t}{C_{\mathrm{ph}}} G_{j}^{n} (T_{\mathrm{e}\ j}^{1} - T_{\mathrm{ph}\ j}^{1}) \tag{61}$$

ergeben. Aus der separaten Berechnung von Elektronen- und Phononenenergie ergeben sich nach Gl. 51 auch zwei verschiedene Drücke. Der Gesamtdruck, der in den Quellterm der Impulsgleichung eingeht, setzt sich additiv aus dem Elektronen- und dem Phononendruck zusammen.

7.2 Darstellung der Ergebnisse

Die Advektions-Diffusionsmodelle in Euler- sowie in Lagrange-Koordinaten wurden zum Vergleich mit den selben Laser- und Materialparametern gelöst, wie zuvor in Abschnitt 3. Die Lösung erfolgte bis zu einer Maximalzeit von 20 ps. In Abb. 7.1 sind die für diesen Zeitpunkt berechneten Zustandsgrößen Elektronen- und Phononentemperatur, Druck und Dichteverteilung dargestellt. Der Vergleich zeigt insgesamt eine sehr gute Übereinstimmung der beiden Modelle. Die Elektronentemperatur wurde vom Lagrange-Modell um wenige Kelvin geringer berechnet, als vom Modell in Euler-Koordinaten. Diese Abweichung wirkt sich jedoch nur äußerst geringfügig auf die berechnete Phononentemperatur und damit auf die Druck- und Dichteverteilung aus.

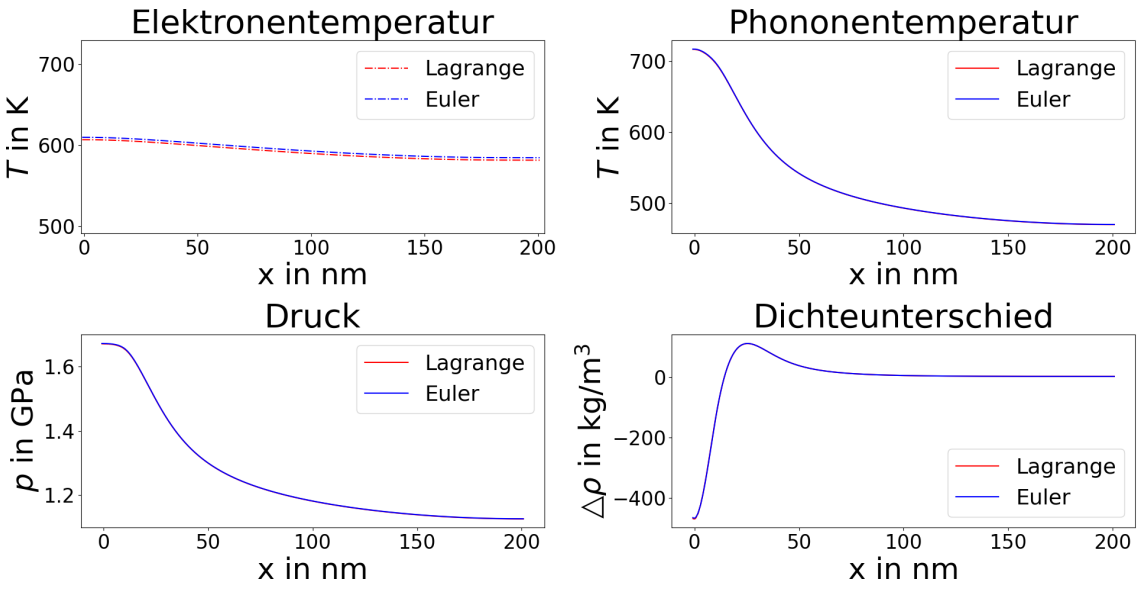

Abbildung 7.1: Vergleich der berechneten Zustandsgrößen Elektronentemperatur, Phononentemperatur, Druck und Dichte aus dem Advektions-Diffusionsmodell in Euler- und Lagrange-Koordinaten zum Zeitpunkt $t = 20$ ps. Die Simulation wurde mit einer Diskretisierung von 600 Ortsschritten Durchgeführt.

Eine genaue Quantifizierung der Einflüsse numerischer Fehler der Modelle auf das Ergebnis ist nicht möglich, da für das Advektions-Diffusions-Modell keine analytische Lösung als Vergleichsmöglichkeit vorliegt. Im Allgemeinen ist anzumerken, dass der Diskretisierungsfehler eines Finite-Differenzen-Schemas von der Anzahl der Ortsschritte abhängt und sich bei feiner gewählten Ortsdiskretisierungen deutlich verringert [11]. Aus diesem Grund wurde die Berechnung in Abb. 7.1 mit einer relativ feinen

Diskretisierung von 600 Ortsschritten durchgeführt. Eine weitere Verkleinerung der Ortsschritte lässt folglich auch eine weitere Verringerung numerischer Fehler erwarten, führt jedoch zu einem hohen Rechenaufwand. Bei der Lösung der Modelle ist demnach stets eine Abwägung zwischen Rechenaufwand und -Genauigkeit zu treffen. Um eine Abschätzung der Rechenzeiten und -Ergebnisse in Abhängigkeit der Zahl der örtlichen Diskretisierungsschritte zu erlangen, wurden jeweils vier Simulationen in Euler- und in Lagrange-Koordinaten bis zu einer Maximalzeit von 10 ps und mit 250, 300, 350 und 400 Ortschritten durchgeführt. In Abb. 7.2 (links) sind die Rechenzeiten der Simulationen über die Anzahl der verwendeten Ortsschritte aufgetragen. Wie erwartet fallen die Rechenzeiten unter Verwendung von Lagrange-Koordinaten geringer aus, als bei der Berechnung in Euler-Koordinaten.

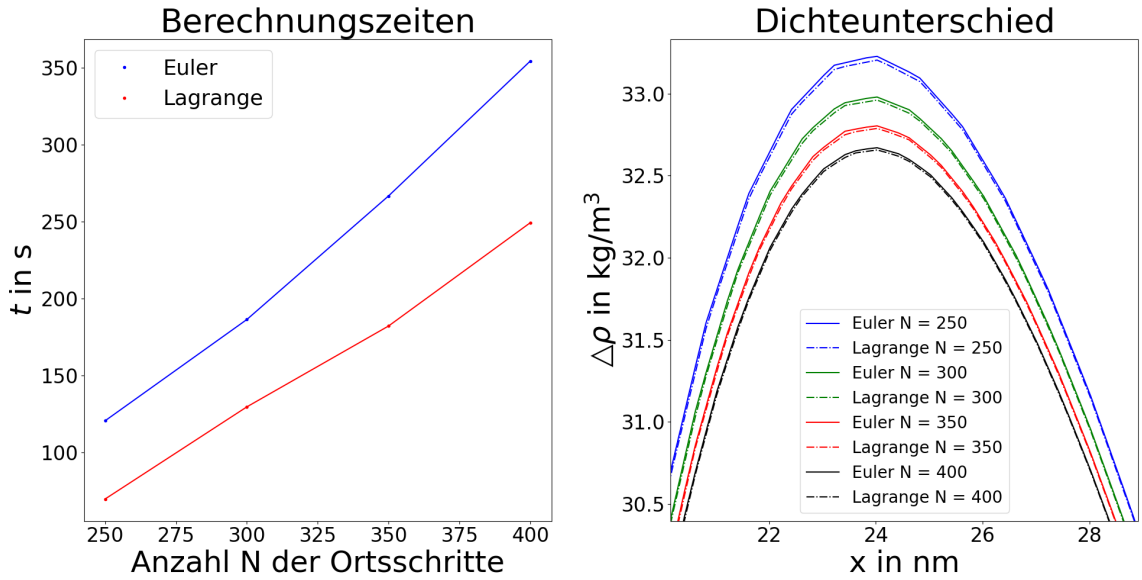

Abbildung 7.2: Vergleich der Berechnungszeiten des Advektions-Diffusions-Modells in Euler und in Lagrange-Koordinaten (links), sowie eine Darstellung der berechneten Dichtwellen zum Zeitpunkt $t = 10$ ps (rechts) für eine grobe Abschätzung des Diskretisierungsfehlers. Wie in den vorherigen Betrachtungen wird die Dichtewelle über die Abweichung $\triangle\rho$ zur Dichte von Gold bei Normalbedingungen dargestellt. Die Berechnungen wurden mit 250, 300, 350 und 400 Ortschritten durchgeführt.

Zur groben Abschätzung des Einflusses der Diskretisierungsfehler werden die aus den vier durchgeführten Simulationen berechneten Materialdichtewellen in einem kleinen Ausschnitt des Definitionsbereiches um das Dichtemaximum herum dargestellt (siehe Abb. 7.2 rechts). Mit zunehmender Zahl der Ortsschritte nimmt die Amplitude der Dichtewelle ab, wobei eine Konvergenz erkennbar ist. Dass das Lagrange-Modell bei gleicher Zahl der Ortsschritte jeweils die geringere Amplitude der Dichtewelle berechnet, lässt folglich darauf schließen, dass die Diskretisierungsfehler des Lagrange-Modells geringer ausfallen, als die des Euler-Modells. Die Differenz zwischen den Ergebnissen beider Modelle ist jedoch als sehr gering zu bewerten.

8 Modellierung der Zustandsgleichung

Wie in den vorherigen Abschnitten bereits ausgeführt wurde, ist zum Schließen des Advektions-Diffusions-Modells eine Zustandsgleichung notwendig, welche die Abhängigkeit des Druckes von der Dichte und der Temperatur bzw. der Energie im Modell wiedergibt. Mit dem Zusammenhang dieser drei Größen werden die thermodynamischen Materialeigenschaften in das Modell integriert. In den bisher betrachteten Simulationen wurde zum Schließen des Modells auf die Zustandsgleichung des idealen Gases zurückgegriffen, die als einfacher Zusammenhang zwar die Untersuchung des Advektions-Diffusions-Modells ermöglicht, dabei jedoch die thermodynamischen Eigenschaften von Aluminium und Gold nicht akkurat wiedergibt. Aus diesem Grund werden in diesem Abschnitt materialspezifische Zustandsgleichungen für Au und Al modelliert, und nachfolgend in das Advektions-Diffusions-Modell integriert. Als Grundlage wird auf die Arbeit [22] zurückgegriffen, in der Zustandsgleichungen für verschiedene Aggregatzustände von Aluminium beschrieben werden. Des Weiteren ist in [22] eine tabellarisch gelöste Zustandsgleichung für Aluminium dargestellt. Aus diesem Grund werden in diesem Abschnitt zunächst die Zustandsgleichungen für Aluminium betrachtet und in Python implementiert. Um die Zustandsgleichungen im Anschluss auch für Gold anwenden zu können, werden darin enthaltene Parameter aus Messdaten sowie theoretischen Zusammenhängen zunächst für Aluminium approximiert und die daraus berechneten Ergebnisse mit den Darstellungen aus [22] verglichen. Mit den selben Approximationsmethoden können anschließend die Parameter für Gold ermittelt werden. Die Approximationen sind erforderlich, da für eine Reihe von Parametern keine Literaturwerte vorhanden sind. Die approximierten Werte dieser Parameter sind den Tabellen 9.1 und 9.3 zu entnehmen.

Für einige der betrachteten Aggregatzustände finden sich empirisch oder auch analytisch modellierte Zustandsgleichungen. Sind für den Druck einzelner Phasengebiete keine exakten oder empirischen Ausdrücke bekannt, folgt die Berechnung des Druckes aus der freien Helmholtzenergie.

Die freie Helmholtzenergie

$$F = E - TS \tag{62}$$

mit der Entropie S und der inneren Energie E beschreibt als thermodynamisches Potential den Gleichgewichtszustand eines thermodynamischen Systems vollständig. So resultiert die Ableitung [22]

$$p = -\left(\frac{\partial F(v,T)}{\partial v}\right)_T \tag{63}$$

der freien Helmholtzenergie nach dem spezifischen Volumen $v = 1/\rho$ bei konstanter

Temperatur T im Druck p. Die Ableitung [22]

$$B = -v_0 \left(\frac{\partial p(v,T)}{\partial v} \right)_T = v_0 \left(\frac{\partial^2 F(v,T)}{\partial v^2} \right)_T \tag{64}$$

des Druckes nach dem spezifischen Volumen bei der Temperatur T führt wiederum zum Kompressionsmodul B, wobei v_0 das spezifische Volumen bei Normalbedingungen darstellt. Über

$$C_s = \sqrt{vB} \tag{65}$$

hängt die Schallgeschwindigkeit C_s mit dem Kompressionsmodul zusammen. Die Entropie S ergibt sich schließlich durch die Ableitung [22]

$$S = -\left(\frac{\partial F(v,T)}{\partial T} \right) \tag{66}$$

der freien Helmholtzenergie nach der Temperatur. Somit stellt die temperatur- und volumenabhängige Modellierung der freien Helmhotzenergie $F(v,T)$ eine vollständige Beschreibung des thermodynamischen Zustandes der simulierten Al und Au Schichten dar. Das bedeutet, die Implementierung der aus der freien Helmholtzenergie abgeleiteten Zustandsgleichung $p(\rho,T)$ repräsentiert die thermodynamischen Materialeigenschaften innerhalb der Simulation vollständig.

Die freie Helmholtzenergie setzt sich aus dem Bindungspotential $\phi(v)$ der Gitteratome, der freien Helmholtzenergie der Phononen $F_{\mathrm{ph}}(v,T)$ sowie der Energie $F_{\mathrm{e}}(v,T)$ des freien Elektronengases mit [22]

$$F(v,T) = \phi(v) + F_{\mathrm{ph}}(v,T) + F_{\mathrm{e}}(v,T) \tag{67}$$

zusammen, womit sich der Gesamtdruck aus den entsprechenden Druckkomponenten über

$$\begin{aligned} p(v,T) &= \left(\frac{\partial \phi(v)}{\partial v} \right)_T + \left(\frac{\partial F_{\mathrm{ph}}(v,T)}{\partial v} \right)_T + \left(\frac{\partial F_e(v,T))}{\partial v} \right)_T \\ &= p_{\mathrm{c}}(v) + p_{\mathrm{ph}}(v,T) + p_{\mathrm{e}}(v,T) \end{aligned} \tag{68}$$

berechnet.

8.1 Druck des freien Elektronengases

Bei konstantem Volumen beträgt die temperaturabhängige Wärmekapazität der freien Leitungselektronen nach dem Sommerfeld-Modell des freien Elektronengases

$$C_\mathrm{e} = \frac{\pi^2 k_\mathrm{B}^2}{3E_\mathrm{f}} T_\mathrm{e}, \tag{69}$$

mit der Boltzmann-Konstante k_B und der Fermi-Energie E_f. Der Proportionalitätsfaktor $\pi^2 k_\mathrm{B}^2/(3E_\mathrm{f})$ heißt theoretischer Sommerfeldparameter γ_th und wird in [16] für Au mit $\gamma_\mathrm{th} = 67{,}6\ \mathrm{Jm^{-3}K^{-2}}$ und für Al mit $\gamma_\mathrm{th} = 91{,}2\ \mathrm{Jm^{-3}K^{-2}}$ angegeben, womit das Modell einen linearen Zusammenhang für $C_\mathrm{e}(T_\mathrm{e})$ liefert. Ein Vergleich zwischen diesen linearen Verläufen und den in das Zwei-Temperatur-Modell integrierten, auf DOS-Berechnungen basierenden Wärmekapazitäten ist in Abbildung 8.1 dargestellt.

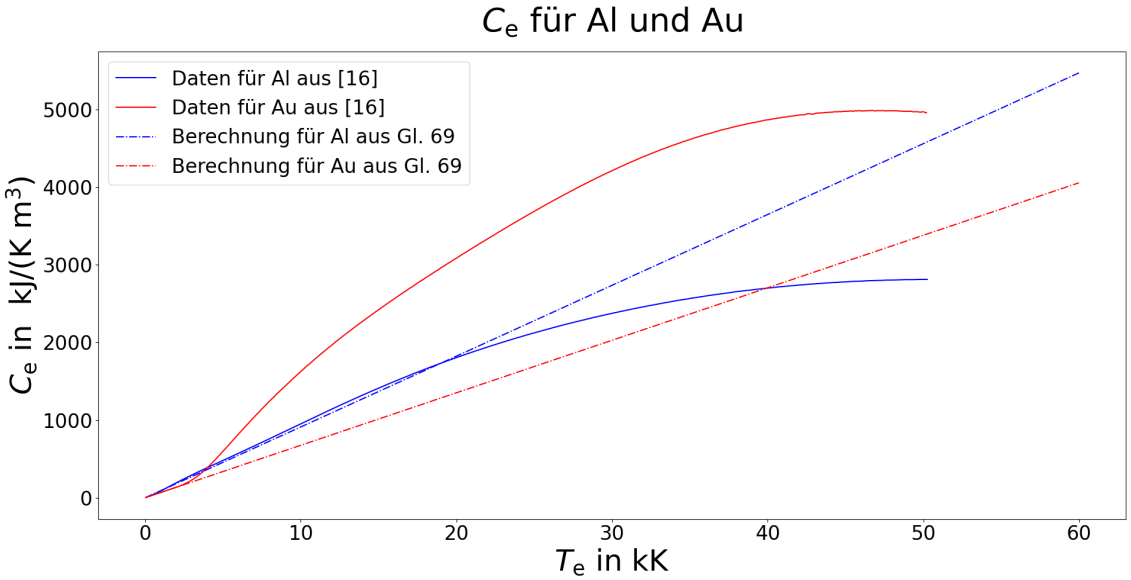

Abbildung 8.1: Darstellung der aus DOS-Berechnungen stammenden, in das Zwei-Temperatur-Modell integrierten elektronischen Wärmekapaziten nach [16]. Zum Vergleich dargestellt sind außerdem die aus dem Sommerfeldparameter γ_th berechneten, linear mit der Temperatur verlaufenden elektronischen Wärmekapazitäten.

Die Fermienergie E_f ist wiederum vom spezifischen Volumen über [18]

$$E_\mathrm{f} = \frac{\hbar^2}{2m_\mathrm{e}^*} (3\pi^2 n_\mathrm{e0})^{2/3} \sigma^{\gamma_\mathrm{e0}} \tag{70}$$

mit $\sigma = \frac{v_0}{v}$ abhängig, wobei n_e0 die Elektronendichte und γ_0e den Grüneisenparameter des freien Elektronengases darstellt [22], dessen Wert weiter unten für Al und Au approximiert wird. Damit gilt für die temperatur- und volumenabhängigen elektronische Wärmekapazität

$$C_\mathrm{e} = \beta_0 T_\mathrm{e} \sigma^{-\gamma_\mathrm{0e}} \tag{71}$$

mit dem Parameter [22]

$$\beta_0 = \left(\frac{\pi^2}{243}\right)^{1/3} \frac{k_B^2 m_e^*}{\rho_{e0}^{2/3}\hbar^2}. \tag{72}$$

Dieser ist in [22] mit $\beta_0 = 0,046$ angegeben. Der Zusammenhang zwischen der elektronischen Wärmekapazität und der freien Helmholtzenergie des Elektronengases ist über

$$C_e = \left(\frac{\partial F_e}{\partial T_e}\right)_V \tag{73}$$

gegeben. Die freie Helmholtzenergie berechnet sich demzufolge durch Integration von Gl. 71 nach der Temperatur mit

$$F_e = -\frac{1}{2}\beta_0 T_e^2 \sigma^{-\gamma_{0e}} \tag{74}$$

und die Ableitung nach dem spezifischen Volumen ergibt schlussendlich den Druck

$$p_e = \frac{\sigma_c^2}{v_0}\left(\frac{\partial F_e}{\partial \sigma}\right)_T = \frac{\beta_0 \gamma_{0e}}{2v_0}\sigma^{1-\gamma_{0e}} T_e^2 \tag{75}$$

der Elektronen. Die Approximation des elektronischen Grüneisenparameters γ_{0e} erfolgt mit Daten für den volumenabhängigen Elektronendruck der Isothermen $T_e = 10$ kK, $T_e = 20$ kK und $T_e = 55$ kK aus [23] (siehe Abb. 8.2) über die kleinsten quadratischen Abweichungen zu Gl. 75. Einsetzen des approximierten Wertes für γ_{0e} in Gl. 75 führt zu den in Abb. 8.3 abgebildeten dichte- und temperaturabhängigen Druckverteilungen.

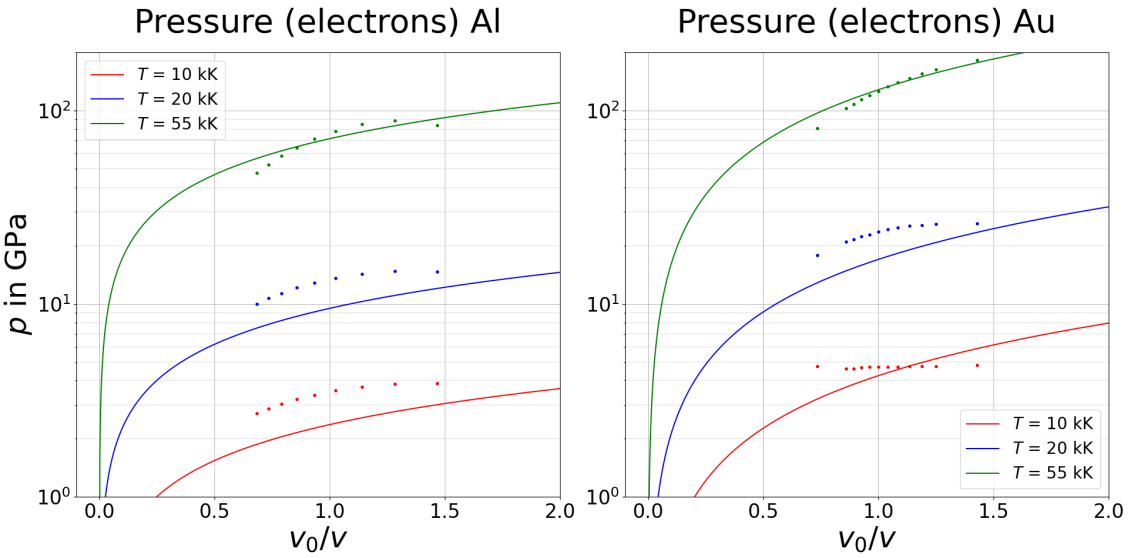

Abbildung 8.2: Datenpunkte aus [23] und Approximation des Elektronendruckes von Aluminium und Gold für die Isothermen 10 kK, 20 kK und 55 kK.

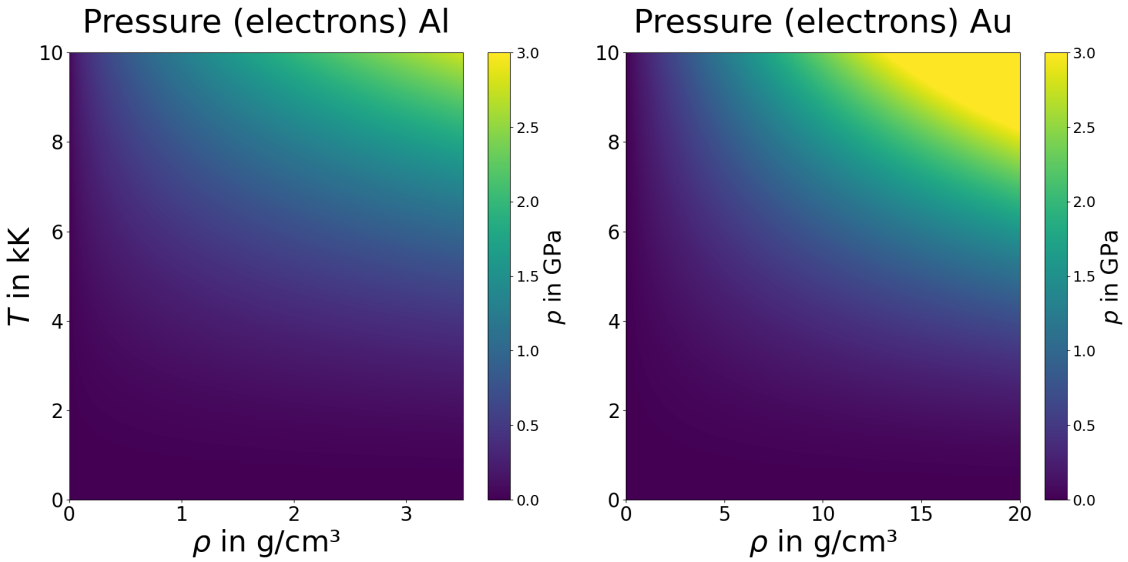

Abbildung 8.3: Berechneter Elektronendruck in Abhängigkeit von der Dichte und der Temperatur für Aluminium und Gold.

8.2 Stabile Festphase

8.2.1 Bindungspotential der kalten Kompression

Wird die Festkörpertemperatur mit $T = 0$ K angenommen, sind keine Phononen angeregt. Das bedeutet, die Gitteratome befinden sich in ihrer Ruhelage. Das spezifische Volumen v des Festkörpers wird in diesem Zustand mit v_{0c} bezeichnet. Findet eine Kompression des Festkörpers $v_{0c}/v > 1$ statt, verkleinern sich die Atomabstände und es wirken abstoßende Kräfte zwischen den Atomen, was in einer Druckkomponente p_c resultiert. Die Kompression bei $T = 0$ K wird auch als kalte Kompression $\sigma_c = v_{0c}/v$ bezeichnet [22] und die daraus resultierende Druckkomponente p_c als Druck der kalten Kompression [22]. Wird der Atomabstand infolge einer Materialausdehnung bei 0 K vergrößert, d. h. es gilt $\sigma_c = v_{0c}/v < 1$, wirken dagegen anziehende Kräfte zwischen den Atomen, was ebenfalls in einer Druckkomponete p_c resultiert. Das den Kraftverlauf verursachende Potential $\phi(v)$ ist volumenabhängig und heißt Bindungspotential. Die Ableitung des Bindungspotentials nach dem spezifischen Volumen v ergibt die Druckkomponente p_c [22]. Dies gilt sowohl für den Bereich der Kompression als auch für den Bereich der Ausdehnung.

Da das Bindungspotential jedoch einen asymmetrischen Verlauf bezüglich der Ruhelage zeigt [22], wird bei dessen Formulierung zwischen dem Bereich der Kompression $\sigma_c > 1$ und der Ausdehnung $\sigma_c < 1$ unterschieden. Das Bindungspotential kann im Bereich der Kompression nach [22] über die polynomiale Approximation

$$\phi(v) = 3v_{0c} \sum_{i=1}^{5} \frac{a_i}{i} (\sigma_c^{\frac{i}{3}} - 1) \tag{76}$$

formuliert werden, wobei die Entwicklung nach dem fünften Glied abgebrochen wird. Durch Ableiten nach dem spezifischen Volumen lässt sich für den Druck der kalten Kompression

$$p_{\mathrm{c}} = \sigma_{\mathrm{c}} \sum_{i=1}^{5} a_i \sigma_{\mathrm{c}}^{\frac{i}{3}} \tag{77}$$

schreiben. Die Approximation der Polynomparameter a_i erfolgt auf Grundlage theoretisch berechneter Werte für $p_{\mathrm{c}}(\sigma_{\mathrm{c}})$ aus [24] (KEOS7-Modell für Al und KEOS5-Modell für Au). Ohne Kompression bei $\sigma_{\mathrm{c}} = 1$ beträgt auch der Kompressionsdruck p_{c} Null, womit sich durch Einsetzen in Gl. 77 die Zwangsbedingung

$$\sum_i a_i = 0 \tag{78}$$

ergibt. Weitere Bedingungen ergeben sich durch Berechnung des Kompressionsmoduls, der mit dem Druck über

$$B = \sigma_{\mathrm{c}} \left(\frac{\partial p_{\mathrm{c}}}{\partial \sigma_{\mathrm{c}}} \right)_T \tag{79}$$

zusammenhängt, bzw. durch Einsetzen von Gl. 77 in Gl. 79 über die Approximation

$$B = \sigma_{\mathrm{c}} \sum_{i=1}^{5} a_i \left(1 + \frac{i}{3} \right) \sigma_{\mathrm{c}}^{1+\frac{i}{3}}. \tag{80}$$

Bei $p_{\mathrm{c}} = 0$ GPa und $T = 0$ K, das bedeutet bei $\sigma_{\mathrm{c}} = 1$ beträgt der Kompressionsmodul von Aluminium nach dem KEOS7-Modell [24] $B_{\mathrm{c}0} = 77,30$ GPa und nach dem KEOS5-Modell [24] für Gold $B_{\mathrm{c}0} = 185,7$ GPa. Durch Einsetzen dieser Bedingungen in Gl. 80 ergibt sich zusammen mit Gl. 78 für die Parameter a_i

$$B_{\mathrm{c}0} = \sum_{i=1}^{5} a_i \frac{i}{3}. \tag{81}$$

Eine dritte Zwangsbedingung folgt aus der Änderung des Kompressionsmoduls mit dem Druck bei 0 GPa und 0 K. Durch numerische Ableitung der Werte des Kompressionsmoduls aus dem KEOS7-Modell nach dem Kompressionsdruck an der Stelle $p_{\mathrm{c}} = 0$ ergibt sich $B_{\mathrm{c}p0} = 4,61$. Für Gold liefert das KEOS5 auf die gleiche Weise den Wert $B_{\mathrm{c}p0} = 6,37$. Unter Berücksichtigung von Gl. 78 gilt für die Polynomparameter

$$B_{\mathrm{c}P0} = \left(\frac{\partial B_{\mathrm{c}}}{\partial P_{\mathrm{c}}} \right)_T = 2 + \frac{1}{B_{\mathrm{c}0}} \sum_{i=1}^{5} a_i \left(\frac{i}{3} \right)^2. \tag{82}$$

Die Parameter a_i wurden schließlich über die kleinste quadratische Abweichung zwischen Gl. 77 und den Werten des Druckes aus dem KEOS7 Modell approximiert. Die Zwangsbedingungen (Gl. 78, 81, 82) sind dabei erforderlich, um die Stetigkeit des Druckverlaufes an der Grenze (bei $\sigma_{\mathrm{c}} = 1$) zum nachfolgend berechneten Bereich der Materialausdehnung zu gewährleisten. Ohne diese würde der Druck an dieser Grenze eine Sprungstelle zeigen und die thermische Gleichgewichtsbedingung wäre nicht

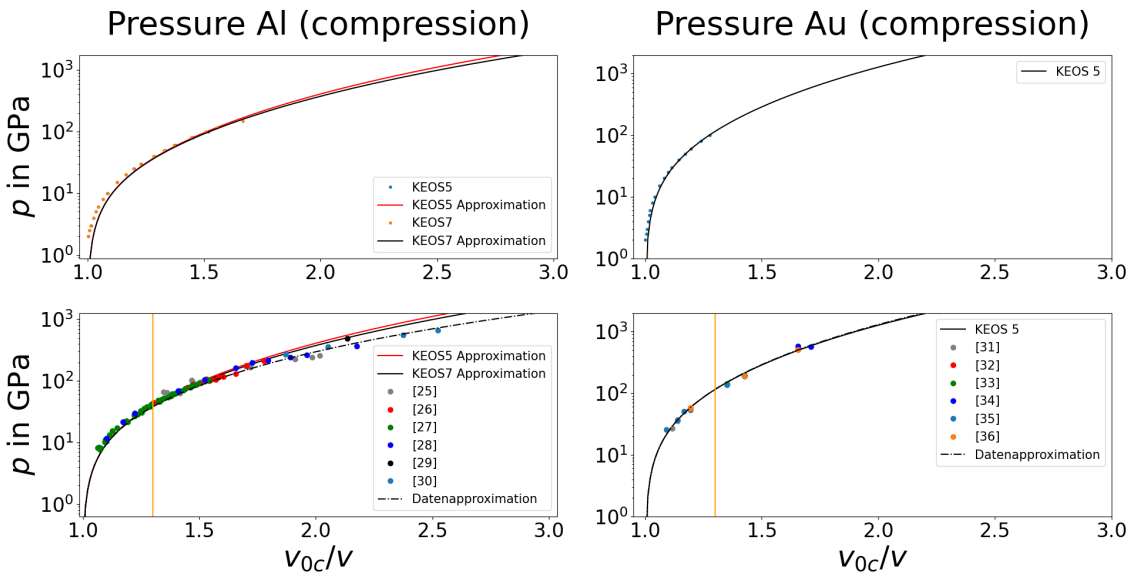

Abbildung 8.4: Approximation der Polynomparameter von Gl. 77 mit den KEOS7 und KEOS5 Modellen (oben) und Vergleich mit der Polynomapproximation aus Messdaten (unten) für die kalte Kompression. Innerhalb des in dieser Betrachtung relevanten Kompressionsbereich ($\sigma_c < 1.3$) stimmen alle drei Approximationen sehr gut überein. Abweichungen der Approximationen von den Daten in der Nähe von $\sigma_c = 1$ sind auf die notwendigen Zwangsbedingungen zurückzuführen.

erfüllt. Abb. 8.4 zeigt vergleichend die approximierten Drücke für das KEOS7- und das KEOS5- Modell, sowie eine Approximation aus experimentell gewonnener Daten. Innerhalb des relevanten Kompressionsbereiches $\sigma_c \leq 1.3$ des Festkörpers zeigen die drei Approximationen gut übereinstimmende Verläufe.

8.2.2 Bindungspotential im Bereich der Ausdehnung

Wie oben beschrieben, verhält sich das Bindungspotential im Bereich der Materialausdehnung, d. h. bei $\sigma_\mathrm{c} = v_{0\mathrm{c}}/v < 1$ asymmetrisch zum Kompressionsbereich, weshalb eine separate Approximation erforderlich ist. Da die Änderung des Bindungspotentials hier mit zunehmendem Atomabstand geringer ist, als im Bereich der Kompression, lässt sich das Bindungspotential nach [22] hinreichend genau als Polynom dritten Grades mit

$$\phi_0(v) = v_{\mathrm{c}0}\left(A_\mathrm{c}\frac{\sigma_\mathrm{c}^m}{m} + B_\mathrm{c}\frac{\sigma_\mathrm{c}^n}{n} + C_\mathrm{c}\frac{\sigma_\mathrm{c}^l}{l}\right) \tag{83}$$

beschreiben, wobei A_c, B_c, C_c, m, n und l die zu approximierenden Polynomparameter darstellen. Durch Ableiten nach dem spezifischen Volumen berechnet sich der Druck mit

$$p_c = A_\mathrm{c}\sigma_\mathrm{c}^{m+1} + B_\mathrm{c}\sigma_\mathrm{c}^{n+1} + C_\mathrm{c}\sigma_\mathrm{c}^{l+1} \tag{84}$$

und durch erneute Ableitung des Druckes nach dem spezifischen Volumen

$$B_\mathrm{c} = A_\mathrm{c}(m+1)\sigma_\mathrm{c}^{m+1} + B_\mathrm{c}(n+1)\sigma_\mathrm{c}^{n+1} + C_\mathrm{c}(l+1)\sigma_\mathrm{c}^{l+1} \tag{85}$$

der Kompressionsmodul. Gl. 84 führt mit $p_c(\sigma_c = 1)$ zu der Zwangsbedingung

$$A_\mathrm{c} + B_\mathrm{c} + C_\mathrm{c} = 0. \tag{86}$$

Auch für den Bereich der Materialausdehnung müssen die Zwangsbedingungen $B_{\mathrm{c}0}$ und $B_{\mathrm{c}p0}$ für den Grenzbereich gelten. Aus Gl. 85 ergeben sich durch Einsetzen von $\sigma_c = 1$ sowie Gl. 86 die Zusammenhänge

$$B_{\mathrm{c}0} = A_\mathrm{c}m + B_\mathrm{c}n + C_\mathrm{c}l. \tag{87}$$

und

$$B_{\mathrm{c}P0} = \frac{A_\mathrm{c}m^2 + B_\mathrm{c}n^2 + C_\mathrm{c}l^2}{B_{\mathrm{c}0}} + 2. \tag{88}$$

Für die Approximation der Parameter werden Werte aus dem KEOS-Modellen [24] (KEOS7 für Al und KEOS5 für Au, siehe Abb. 8.5) für die volumenabhängige Schallgeschwindigkeit $C_s(v)$ im Bereich $\sigma_\mathrm{c} < 1$ verwendet, wobei die Beziehung $C_s = \sqrt{vB_\mathrm{c}}$ genutzt wird, um die Parameter über die kleinste quadratische Abweichung zu Gl. 85 zu approximieren. Gl. 86, 87 und 88 bilden die Zwangsbedingungen dieser Approximation.

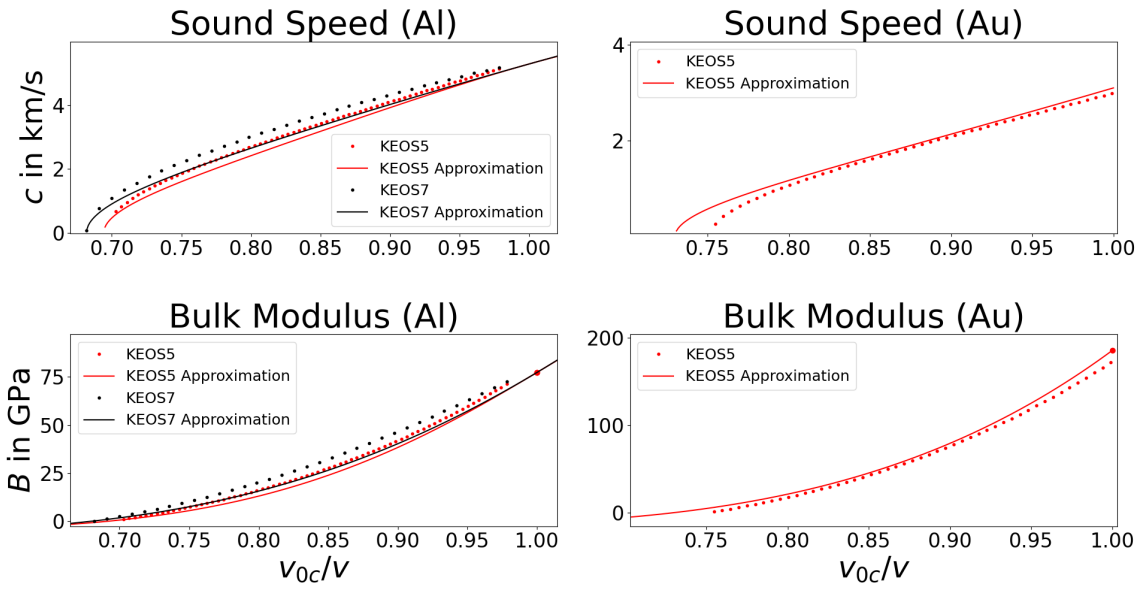

Abbildung 8.5: Approximation der Polynomparameter von Gl. 85 mit den KEOS7 und KEOS5 Modellen. Abweichungen der Approximationen von den Daten der KEOS-Modelle in der Nähe von $\sigma_c = 1$ sind auf die notwendigen Zwangsbedingungen zurückzuführen.

8.2.3 Phononendruck

Findet ausgehend vom absoluten Nullpunkt eine Temperaturerhöhung statt, führt dies zur Anregung von Phononen. Der Phononendruck lässt sich nach Gl. 68 als die Ableitung

$$p_{\mathrm{ph}} = \left(\frac{\partial F_{\mathrm{ph}}(v,T)}{\partial v} \right)_T \tag{89}$$

der freien Helmholtzenergie des Gitters nach dem spezifischen Volumen entlang jeder Isotherme T formulieren. Eine Hochtemperaturnäherung für die freie Helmholtzenergie $F_{\mathrm{ph}}^{(\mathrm{S})}$ des Phononensystems im Festkörper lässt sich aus dem in Abschnitt 2.2 beschriebenen Debye-Modell gewinnen, welches mit Gl. 12 einen linearen Zusammenhang zwischen Temperatur und innerer Energie $E_{\mathrm{ph}}^{(\mathrm{S})}$ der Phononen für den Grenzfall hoher Temperaturen liefert. Desweiteren gilt nach Gl. 62 der Zusammenhang $E_{\mathrm{ph}}^{(\mathrm{S})} = F_{\mathrm{ph}}^{(\mathrm{S})} + TS$.

Einsetzen der Enthropie S nach Gl. 66 und der inneren Energie $E_{\mathrm{ph}}^{(\mathrm{S})}$ nach Dulong-Petit (Gl. 12) ergibt die Differentialgleichung

$$F_{\mathrm{ph}}^{(\mathrm{S})}(T) - T\frac{\partial F_{\mathrm{ph}}^{(\mathrm{S})}(T)}{\partial T} = 3RT. \tag{90}$$

Eine Lösung dieser Gleichung führt zu dem Ausdruck [22]

$$F_{\mathrm{ph}}^{(\mathrm{S})}(T) = 3RT \log\left(\frac{\Theta}{T}\right). \tag{91}$$

mit der Debye-Temperatur $\Theta = \frac{\hbar\Omega_\mathrm{D}}{k_\mathrm{B}}$. Dabei geht das Debye-Modell von der Normdichte des Festkörpers aus. Bei Volumenänderungen des Festkörpers verschieben sich jedoch die besetzbaren Zustände innerhalb des Gitters und damit die Phononenfrequenzen [37]. Quantenmechanisch lässt sich dieses Verhalten durch die Veränderung der Atomabstände und demzufolge in der Steckung bzw. Stauchung des Parabelpotentials im Hamiltonoperator erklären. Somit verschieben sich die Eigenzustände bei der Kompression hin zu größeren und bei Ausdehnung hin zu geringeren Energien bzw. Frequenzen. Die daraus resultierende Volumenabhängigkeit der Debye-Frequenz Ω_D bzw. der Debye-Temperatur wird mithilfe des Grüneisen-Modells dargestellt [37][38]. Die relative Änderung der Debye-Frequenz mit dem spezifischen Volumen ist demnach das Differential [37]

$$\frac{\partial\Omega_\mathrm{D}}{\partial v} = -\gamma\frac{\Omega_\mathrm{D}}{v} \tag{92}$$

bzw. mit der Debye-Temperatur [38]

$$-\frac{\partial\log\Theta}{\partial\log v} = \gamma. \tag{93}$$

Der Proportionalitätsfaktor γ ist wiederum volumenabhängig und heißt Grüneisenfunktion. Für sie wird in [39] der ideale theoretische Zusammenhang

$$\gamma = \gamma_\infty + (\gamma_0 - \gamma_\infty)\sigma^{-\frac{\gamma_0}{\gamma_0-\gamma_\infty}} \tag{94}$$

beschrieben. γ_∞ hat den Wert 2/3 für alle Metalle, γ_0 ist materialabhängig und wird in [39] für Aluminium mit 2,14 und für Gold mit 3,06 angegeben. In [22] wird dieser Verlauf über

$$\gamma = \gamma_\infty + (\gamma_0 + \gamma_\infty)\left(\frac{(B_\mathrm{S}^2 + D_\mathrm{S}^2)}{B_\mathrm{S}^2 + (x + D_\mathrm{S})^2}\right) \tag{95}$$

mit den Approximationsparametern B_S und D_S sowie $x = \log(\sigma)$ modelliert.

Durch Integration von Gl. 93 nach dem spezifischen Volumen und Einsetzen der Grüneisenfunktion nach Gl. 95 ergibt sich für die volumenabhängige Debye-Temperatur des Festkörpers nach [22]

$$\Theta^{(\mathrm{S})} = \Theta_{0\mathrm{S}}\sigma^{\gamma_\infty}\exp\left(\frac{(\gamma_{0\mathrm{S}} - \gamma_\infty)({B_\mathrm{S}}^2 + {D_\mathrm{S}}^2)}{{B_\mathrm{S}}^2}\left(\arctan\left(\frac{xB_\mathrm{S}}{{B_\mathrm{S}}^2 + (x + {D_\mathrm{S}}^2)D_\mathrm{S}}\right)\right)\right). \tag{96}$$

$\Theta_{0\mathrm{S}}$ stellt die Debye-Temperatur des Festkörpers ohne Kompression dar. Da $\Theta_{0\mathrm{S}}$ jedoch bei der Berechnung des Druckes p_ph verschwindet, werden die konkreten Werte nicht benötigt. Die Materialparameter B_S und D_S können bei gegebenen Kurvenverläufen der Grüneisenfunktion über die Methode der kleinsten quadratischen Abweichung zu Gl. 95 approximiert werden. Die in der Literatur angegebenen Werte bzw. Verläufe für die Grüneisenfunktion zeigen teilweise erhebliche Abweichungen vom theoretischen funktionalen Zusammenhang nach G. 94. Abb. 8.6 und 8.7 zeigen vergleichend die

Verläufe bzw. Werte der Grüneisenfunktionen nach [39] und [24] (KEOS5) sowie den theoretischen Verlauf nach Gl. 94 für Aluminium und Gold.

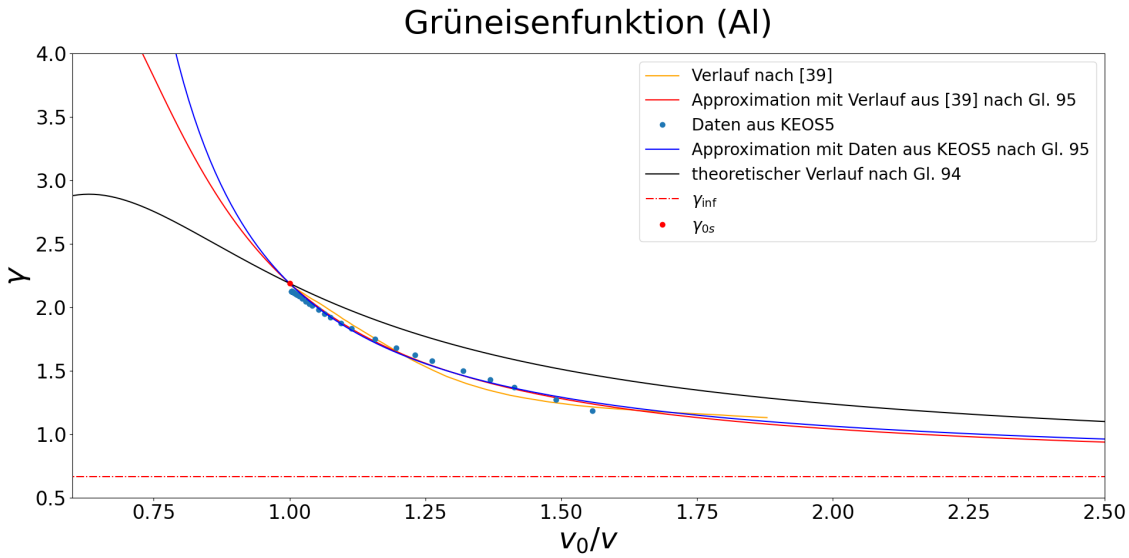

Abbildung 8.6: Volumenabhängiger Verlauf der Grüneisenfunktion von Aluminium aus verschiedenen Quellen und die daraus resultierenden Kurvenapproximationen.

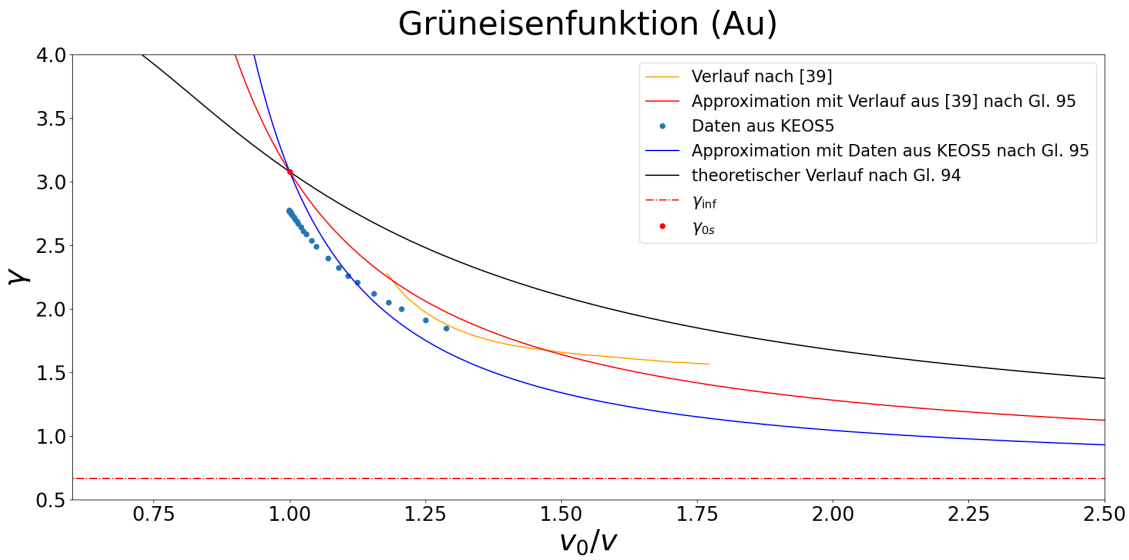

Abbildung 8.7: Volumenabhängiger Verlauf der Grüneisenfunktion von Gold aus verschiedenen Quellen und die daraus resultierenden Kurvenapproximationen.

Außerdem dargestellt sind in Abb. 8.6 und 8.7 die Kurven nach Gl. 95 mit den jeweils approximierten Parametern B_S und D_S. Auch diese unterscheiden sich stark vom idealen Verlauf. Durch Einsetzen von Gl. 96 in Gl. 91 und analytische Ableitung der freien Helmholtzenergie nach dem spezifischen Volumen ergibt sich schließlich der Phononendruck

$$p_{\mathrm{ph}} = -\frac{\partial F_{\mathrm{ph}}^{(\mathrm{S})}}{\partial v} = 3RT\frac{\sigma^2}{v_0\Theta^{(\mathrm{S})}}\frac{\partial\Theta^{(\mathrm{S})}}{\partial\sigma}. \tag{97}$$

Trotz der unterschiedlichen Verläufe der approximierten Grüneisenfunktionen variieren die Ergebnisse im Phasendiagramm jedoch nur wenig mit der Wahl des konkreten Kurvenverlaufes und den daraus resultierenden Parametern B_S und D_S. Dies zeigt die Konstruktion der Schmelzkurve in Abschnitt 8.3. Dort wurden die Parameter B_S und D_S vergleichend aus den drei betrachteten Verläufe der Grüneisenfunktion approximiert, jeweils in p_ph eingesetzt und anschließend die Schmelzkurve über die Gleichgewichtsbedingung des Festkörperdruckes $p_\mathrm{e} + p_\mathrm{c} + p_\mathrm{ph}$ und dem in Abschnitt 8.3 berechneten Schmelzdruck p_m konstruiert. Wie Abb. 8.9 zeigt, ist der berechnete Schmelzkurvenverlauf nahezu unabhängig von der Wahl der in p_ph integrierten Güneisenfunktion.

Begrenzt wird der Bereich der stabilen Festphase zudem von der isobaren thermischen Expansionskurve bei Atmosphärendruck (siehe Abb. 8.9, grüne Linie). Dabei handelt es sich um den v-T-Verlauf, entlang dessen der Festkörperdruck $p_\mathrm{e} + p_\mathrm{c} + p_\mathrm{ph}$ genau 10^{-4} GPa beträgt. Durch Einsetzen der Druckkomponenten (Gl. 75, 77, 84, 97) in

$$p_\mathrm{e}(v,T) + p_\mathrm{c}(v,T) + p_\mathrm{ph}(v,T) = 10^{-4}\ \mathrm{GPa} \tag{98}$$

kann für die Kurve $v(T)_{p=10^{-4}\mathrm{GPa}}$ ein analytischer Zusammenhang hergeleitet werden. Dieser wird für einen Vergleich mit experimentell gewonnenen Daten in die Ableitung

$$\alpha_\mathrm{th} = \frac{1}{v}\left(\frac{\partial v}{\partial T}\right) \tag{99}$$

eingesetzt, um den temperaturabhängigen thermischen Expansionskoeffizienten $\alpha_\mathrm{th}(T)$ bei Normaldruck zu berechnen (siehe Abb. 8.8). In [40] wird der thermische Expansionskoeffizient von Aluminium experimentell für einen Temperaturbereich bis 773 K bestimmt. Die aus Gl. 99 berechnete thermische Expansion (siehe Abb. 8.8) verläuft

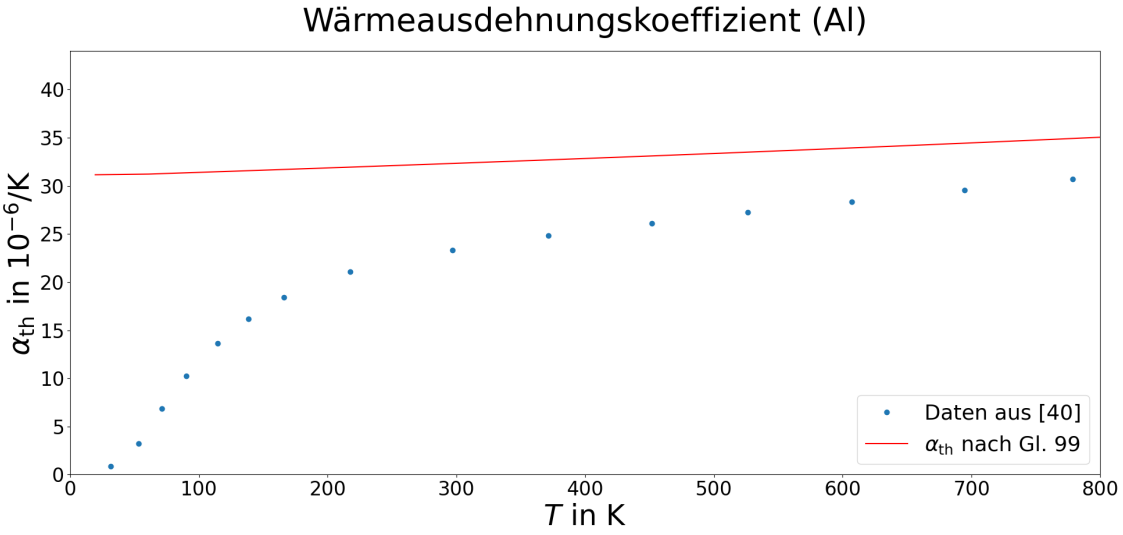

Abbildung 8.8: Berechneter Verlauf des temperaturabhängigen thermischen Expansionskoeffizienten von Aluminium bei Normaldruck und Vergleich mit Daten aus [40].

nahezu linear mit ansteigender Temperatur. Im Bereich unterhalb der Raumtemperatur zeigt dieser Verlauf eine große Diskrepanz zu den Messdaten aus [40]. In dem für die Betrachtungen dieser Arbeit relevanten Temperaturbereich oberhalb von 300 K nähern sich der berechnete Verlauf von $\alpha_{\mathrm{th}}(T)$ und die Datenpunkte jedoch stark an.

8.3 Schmelzzone

Wird die Temperatur des Festkörpers erhöht, erfolgt oberhalb einer dichteabhängigen Schmelztemperatur T_{m} der Übergang in die Schmelzzone, einem stabilen Zweiphasengebiet. Die Kurve $T_{\mathrm{m}}(\rho)$ bzw. $T_{\mathrm{m}}(v)$ heißt Schmelzkurve und trifft am Schmelzpunkt T_{m0} bei Atmosphärendruck mit der isobaren Expansionskurve zusammen (siehe Abb. 8.9). Einen einfachen, empirischen Zusammenhang zwischen Schmelzdruck und -temperatur liefert die Simon-Glatzel-Gleichung [22]

$$T_{\mathrm{m}} = T_{\mathrm{m0}} \left(\frac{p_{\mathrm{m}}}{A} + 1 \right)^{B}. \tag{100}$$

Abb. 8.10 zeigt den T_{m}-p_{m}-Verlauf und die Approximation der Materialparameter A und B für Aluminium (links) über Messwerte aus [41] und einen Vergleich mit Messwerten aus [42] für einen Bereich geringerer Schmelztemperaturen. Die Konstruktion der Schmelzkurve $T_{\mathrm{m}}(\rho)_{p_{\mathrm{m}}=p_{\mathrm{c}}+p_{\mathrm{ph}}+p_{\mathrm{e}}}$ erfolgte numerisch über die thermische Gleichgewichtsbedingung, d. h. den stetigen Übergang des Schmelzdruckes p_{m} zum Druck $p_{\mathrm{e}} + p_{\mathrm{c}} + p_{\mathrm{ph}}$ im Festphasenbereich. Abb. 8.9 zeigt einen Ausschnitt aus der Konstruktion des Phasendiagramms für das Beispiel Aluminium. Darin ist die Schmelzkurve zwischen dem Bereich der stabilen Festphase (S) und der Schmelzzone (S+L) eingetragen. Vergleichend wurden hier die drei verschiedenen Grüneisenfunktionen aus Abb. 8.6 verwendet, um jeweils zunächst den Festkörperdruck zu berechnen und daraus anschließend die Schmelzkurve zu konstruieren. Obwohl sich die Verläufe der drei betrachteten Grüneisenfunktionen unterscheiden, wird hier deutlich, dass die Auswirkung auf die berechnete Phononenenergie und damit auf die konstruierte Schmelzkurve gering ausfällt.

In [43] wird zudem ein semi-empirisches Modell zur Berechnung der Schmelzkurve hergeleitet, welches im Folgenden mit der empirischen Simon-Glatzel-Gleichung (Gl. 100) verglichen werden soll. Dieses basiert auf dem Grüneisen-Modell (Gl. 90), sowie auf dem Schmelzkriterium nach Lindemann. Letzteres geht davon aus, dass die Schmelztemperatur T_{m} dann erreicht ist, wenn das Verhältnis der Gitterabstände bezüglich der Ruhelagen der Atome zu deren maximaler Schwingungsauslenkung einen kritischen Wert erreicht und damit die Kristallordnung zusammenbricht [44].

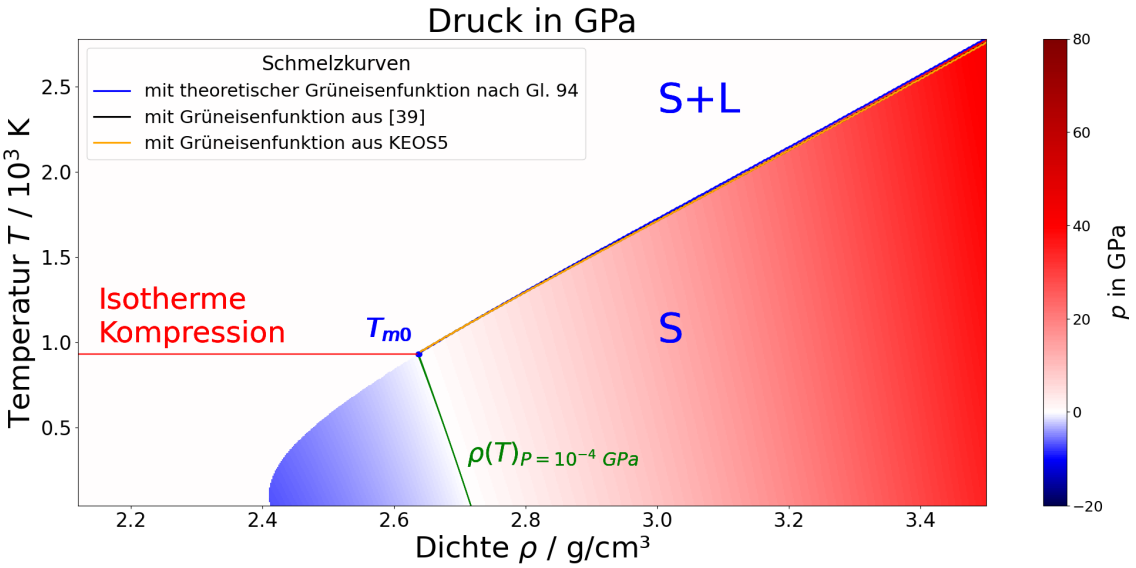

Abbildung 8.9: Ausschnitt aus der Konstruktion des Phasendiagramms. Dargestellt ist der Bereich der stabilen Festphase (S) und der Schmelzzone (S+L). Dazwischen befindet sich die Schmelzkurve, die jeweils aus den Verläufen der Grüneisenfunktionen verschiedener Quellen konstruiert wurde. Dass sich diese drei Kurven nur sehr gering voneinander unterscheiden, zeigt den geringen Einfluss der Wahl des konkreten Verlaufes der Grüneisenfunktion. Ebenfalls eingetragen sind die isobare Expansionskurve (grün), die isotherme Kompression (rot)

In [45] wird das Lindemann-Kriterium als Schmelztemperatur in Abhängigkeit des Kristallvolumens sowie der Debye-Temperatur mit

$$T_\mathrm{m} = const. \cdot v^{2/3}\Theta_D \tag{101}$$

modelliert. In [43] wird diese Modellierung durch logarithmische Ableitung von Gl. 101 und anschließendes Einsetzen von Gl. 93 mit dem Grüneisen-Modell kombiniert, was zu

$$\frac{\partial \log(T_\mathrm{m})}{\partial v} = \frac{2}{v}\left(\frac{1}{3} - \gamma_G\right) \tag{102}$$

führt. Einsetzen der Grüneisenfunktion in γ_G sowie anschließende Integration ergibt schließlich den Zusammenhang

$$T_\mathrm{m} = T_{\mathrm{m}0}\left(\frac{v}{v_0}\right)^{2/3} \exp\left(\frac{2\gamma_0}{q}\left(1 - \left(\frac{v}{v_0}\right)^q\right)\right) \tag{103}$$

zwischen Schmelztemperatur und Kristallvolumen mit dem Materialparameter q. An dieser Stelle ist noch anzumerken, dass für die Grüneisenfunktion in [43] der Ausdruck

$$\gamma_G = \gamma_0\left(\frac{v}{v_0}\right)^q \tag{104}$$

angenommen wird, der im relevanten Volumenbereich gut mit Gl. 94 übereinstimmt,

jedoch den Grenzwert $\gamma(v \to \infty) = \gamma_\infty$ nicht berücksichtigt. Als Zustandsgleichung wird in [43] der Zusammenhang [46]

$$p_\mathrm{m} = 3K_0 \left(\frac{v}{v_0}\right)^{-2/3} \left(1 - \frac{v}{v_0}\right)^{1/3} \exp\left(\frac{2}{3}(K_0' - 1)\left(1 - \left(\frac{v}{v_0}\right)^{1/3}\right)\right) \tag{105}$$

zwischen Schmelzdruck und Kristallvolumen mit den Materialparametern K_0 und K_0' implementiert. Die experimentellen Messdaten für den temperaturabhängigen Schmelzdruck aus [41] ermöglichen die numerische Approximation der Parameter q, K_0 und K_0' aus Gl. 103 und Gl. 105 für Aluminium (siehe Abb. 8.10 links). Für Gold (Abb. 8.10 rechts) sind diese Parameter bereits in [43] mit $q = 1,22$, $K_0 = 0,95$ und $K_0' = 126,37$ dargestellt. Sowohl für Gold, als auch für Aluminium zeigen die resultierenden Schmelzkurven gute Übereinstimmungen mit den Verläufen aus der Simon-Glatzel-Gleichung.

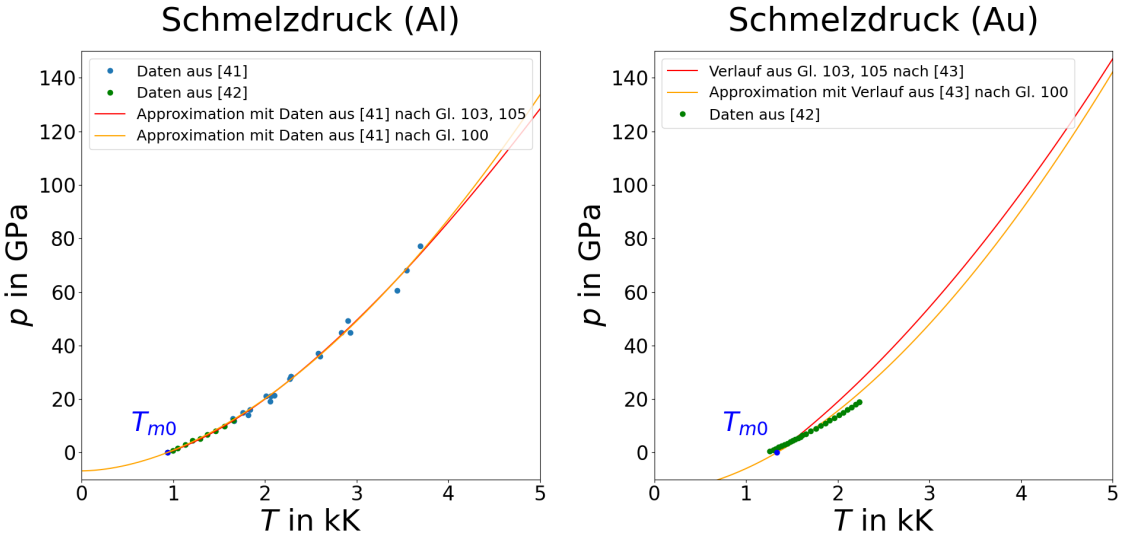

Abbildung 8.10: Vergleich der Schmelzkurven nach der empirischen Simon-Glatzel-Gleichung (Gl. 100) und dem semi-empirischen Modell nach [43] (Gl. 103, 105). Die Parameterapproximation für Aluminium erfolgte aus Messdaten (oben links). Die Parameterapproximation für Gold erfolgte bereits in [43]. Zum Vergleich sind zusätzlich Messwerte aus [42] dargestellt.

8.4 Stabile Gasphase

Im Temperaturbereich oberhalb der isothermen Kompressionskurve liegt bei kleineren Dichten die stabile Gasphase vor. Die reduzierte Van-der-Waals-Gleichung

$$p_r = \frac{8T_r}{3v_r - 1} - \frac{3}{v_r^2} \tag{106}$$

stellt einen empirischen Zusammenhang zwischen Temperatur, Dichte und Druck dar, welcher dem Verhalten eines realen Gases wesentlich näher kommt, als die Zustandsgleichung des ideales Gases. Dabei gilt die Normierung auf den kritischen Punkt mit $T_\mathrm{r} = \frac{T}{T_\mathrm{crit}}$, $v_\mathrm{r} = \frac{v}{v_\mathrm{crit}}$ bzw. $\rho_\mathrm{r} = \frac{\rho}{\rho_\mathrm{crit}}$ und $p_\mathrm{r} = \frac{p}{p_\mathrm{crit}}$. Die kritischen Werte für Aluminium sind in [22] mit $T_\mathrm{crit} = 7,158$ kK, $v_\mathrm{crit} = 1,23\ \mathrm{cm}^3/\mathrm{g}$ und $p_\mathrm{crit} = 1,41$ GPa angegeben. Die Werte für Gold sind beispielsweise den experimentellen Bestimmungen aus [47] mit $T_\mathrm{crit} = 7400 \pm 1100$ K, $\rho_\mathrm{crit} = 7,58 \pm 1,22\ \mathrm{g/cm}^3$ und $p_\mathrm{crit} = 0,35 \pm 0,02$ GPa zu entnehmen. An dieser Stelle muss jedoch darauf hingewiesen werden, dass sich die in der Literatur theoretisch oder experimentell ermittelten kritischen Werte teilweise stark voneinander unterscheiden [47].

Die Van-der-Waals-Gleichung modelliert neben dem Druck der stabilen Gasphase auch die verschiedenen, unterhalb der kritischen Temperatur auftretenden, stabilen und metastabilen Phasenbereiche. Wird von der stabilen Gasphase unterhalb von T_crit ausgegangen und die Dichte erhöht, geht das Gas zunächst in einen metastabilen Zustand über [48]. Die Dichte, an der dieser Übergang erfolgt, ist temperaturabhängig, die entsprechende Kurve $T_\mathrm{bin}(\rho)$ heißt Binodale. Die weitere Verdichtung bewirkt den Übergang in den stabilen Zweiphasenbereich [48], die abgrenzende Kurve heißt Spinodale $T_\mathrm{spin}(\rho)$. Beide Phasengebiete zeigen entgegen dem Verlauf der Van-der-Waals-Gleichung einen über die Dichte konstanten Druck [22]. Anschließend folgt bei entsprechender Verdichtung der Übergang in die metastabile und schließlich die stabile Flüssigphase. Abb. 8.11 zeigt den Verlauf des reduzierten Druckes p_r nach der Van-der-Waals-Gleichung für vier verschiedene Isothermen. Die durch die lokalen Extremstellen des reduzierten Druckes verlaufende Kurve $T_\mathrm{r\ spin}(v_\mathrm{r})$ definiert die Spinodale [22]. Sie kann durch die analytische Extremwertberechnung

$$\frac{\partial}{\partial v_\mathrm{r}} \left(\frac{8T_\mathrm{r}}{3v_\mathrm{r} - 1} - \frac{3}{v_\mathrm{r}^2} \right) = \frac{6}{v_\mathrm{r}^3} - \frac{24T_\mathrm{r}}{(3v_\mathrm{r} - 1)^2} = 0 \tag{107}$$

$$\Rightarrow T_\mathrm{r\ spin}(v_r) = \frac{6}{24v_\mathrm{r}^3} (3v_\mathrm{r} - 1)^2 \tag{108}$$

hergeleitet werden. Die Binodale, sowie die konstanten Drücke der metastabilen Gasphase und dem stabilen Zweiphasengebiet können aus den isothermen Druckverläufen der Van-der-Waals-Gleichung über eine Maxwell-Konstruktion berechnet werden. Dafür muss für jede Isotherme der konstante Druck p_maxwell so berechnet werden, dass die Bilanz der von p_maxwell und dem Druckverlauf aus der Van-der-Waals-Gleichung einge-

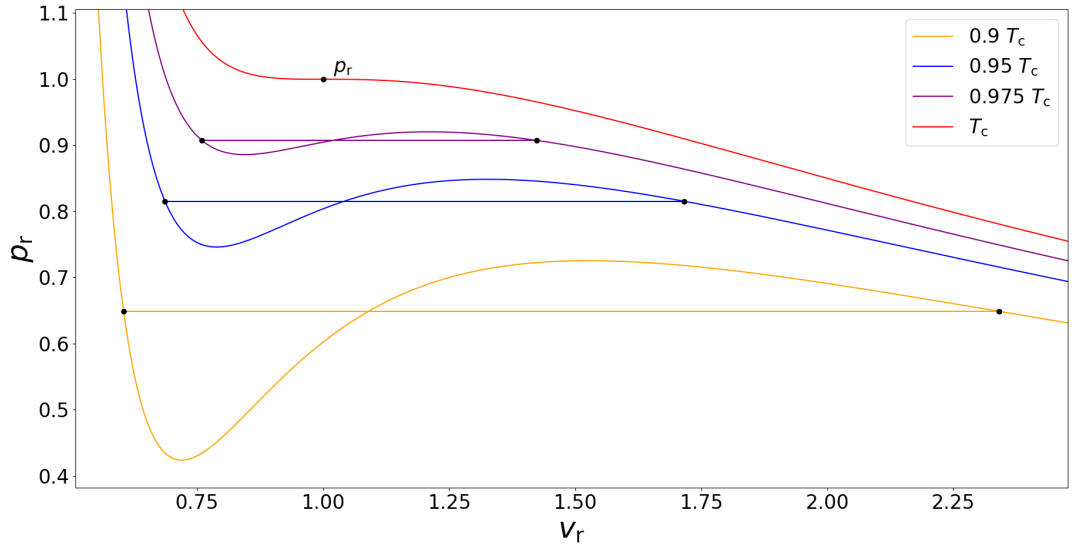

Abbildung 8.11: Verlauf des reduzierten Druckes p_r in Abhängigkeit vom reduzierten Volumen v_r nach der Van-der-Waals-Gleichung für vier verschiedene Isothermen. Da die reduzierten Größen auf den kritischen Punkt normiert sind, stellen sie dimensionslose Größen dar. Die waagrechten Linien kennzeichen die aus der Maxwell-Konstruktion resultierenden, konstanten Druckverläufe. Die Schnittpunkte (durch schwarze Punkte gekennzeichnet) bilden die Binodale.

schlossenen Fläche gleich null ist [22] (vgl. Abb. 8.11). Die Schnittpunkte bilden zugleich die Integrationsgrenzen. Die Maxwell-Konstruktion kann also mit dem Gleichungssystem

$$\int_{v_{\mathrm{r1}}(T_{\mathrm{r}})}^{v_{\mathrm{r2}}(T_r)} \left(\left(\frac{8T_{\mathrm{r}}}{3v_{\mathrm{r}} - 1} - \frac{3}{v_{\mathrm{r}}^2} \right) - p_{\mathrm{maxwell}} \right) dv_{\mathrm{r}} = 0 \tag{109}$$

$$\Rightarrow \frac{8T_{\mathrm{r}} \ln(3v_{\mathrm{r2}} - 1)}{3} - p_{\mathrm{r}} v_{\mathrm{r2}} + \frac{3}{v_{\mathrm{r2}}} - \frac{8T_{\mathrm{r}} \ln(3v_{\mathrm{r1}} - 1)}{3} + p_{\mathrm{r}} v_{\mathrm{r1}} - \frac{3}{v_{\mathrm{r1}}} = 0 \tag{110}$$

$$p_{\mathrm{r}}(v_{\mathrm{r1}}(T_{\mathrm{r}})) = \left(\frac{8T_{\mathrm{r}}}{3v_{\mathrm{r1}} - 1} - \frac{3}{v_{\mathrm{r1}}^2} \right) = p_{\mathrm{maxwell}} \tag{111}$$

$$p_{\mathrm{r}}(v_{\mathrm{r2}}(T_{\mathrm{r}})) = \left(\frac{8T_{\mathrm{r}}}{3v_{\mathrm{r2}} - 1} - \frac{3}{v_{\mathrm{r2}}^2} \right) = p_{\mathrm{maxwell}} \tag{112}$$

ausgedrückt werden, das für jede Isotherme T_{r} gelöst wird. Die Lösung erfolgte numerisch. Die Kurven $T(v_{\mathrm{r1}})$ und $T(v_{\mathrm{r2}})$ sind in dieser Formulierung die beiden Äste der Binodale für $v < v_{\mathrm{crit}}$ und $v > v_{\mathrm{crit}}$ bzw. $\rho > \rho_{\mathrm{crit}}$ und $\rho < \rho_{\mathrm{crit}}$. Der Gasdruck ergibt sich schlussendlich mit $p^{(\mathrm{G})} = p_{\mathrm{r}} p_{\mathrm{crit}}$ bzw. $p^{(\mathrm{G})} = p_{\mathrm{maxwell}} p_{\mathrm{crit}}$ für das Volumen $v = v_{\mathrm{r}} v_{\mathrm{crit}}$ und die Temperatur $T = T_{\mathrm{r}} T_{\mathrm{crit}}$.

8.5 Flüssigphase

Analog zum Festkörperdruck setzt sich der Druck der stabilen Flüssigphase aus dem Elektronendruck p_e, der aus dem Bindungspotential resultierenden Druckkomponente p_c, sowie einem Phononendruck $p_\mathrm{ph}^{(\mathrm{L})}$ zusammen.

$$p^{(\mathrm{L})} = p_\mathrm{e} + p_\mathrm{c} + p_\mathrm{ph}^{(\mathrm{L})} \tag{113}$$

Die Parameter für p_e und p_c entsprechen denen der stabilen Festphase. Der Flüssigdruck unterscheidet sich vom Festkörperdruck lediglich in der Komponente des Phononendrucks. Wie bereits bei der Modellierung der Schmelzkurve beschrieben, bricht bei Überschreiten der Schmelztemperatur die Kristallordnung zusammen. Infolge der vergrößerten Atomabstände treten mit steigender Temperatur zunehmend anharmonische Effekte im Phononensystem auf, die bei der Modellierung der freien Helmholtzenergie durch Korrekturterme im Debye-Modell berücksichtigt werden. Diese Korrekturterme ermöglichen einen stetigen Übergang des Flüssigdruckes an der Grenze zur Gasphase. In [22] wird die freie Helmholtzenergie des Phononensystems in der Flüssigphase mit

$$F_\mathrm{ph}^{(\mathrm{L})} = F_\mathrm{t} + F_\mathrm{m} \tag{114}$$

$$F_\mathrm{t} = \frac{3R}{2}\left(1 + \frac{\sigma T_T}{(\sigma+\sigma_T)(T+T_T)}\right) T\ \log\left(\frac{\Theta^{(\mathrm{L})}}{T}\right) \tag{115}$$

$$\Theta^{(\mathrm{L})}(v,T) = \sigma^{2/3} T_{sa}\left(\frac{\Theta_a + T}{T_{ca} + T}\right) \tag{116}$$

$$\Theta_a = \Theta_{0\mathrm{L}}\sigma^{\gamma_\infty}\exp\left(\frac{(\gamma_{0\mathrm{L}}-\gamma_\infty)({B_\mathrm{L}}^2+{D_\mathrm{L}}^2)}{{B_\mathrm{L}}^2}\left(\arctan\left(\frac{xB_L}{{B_\mathrm{L}}^2+(x+{D_\mathrm{L}}^2)D_\mathrm{L}}\right)\right)\right) \tag{117}$$

mit $\sigma_T = 0.09$, $T_T = 30$ kK, $T_{sa} = 4$ kK, $T_{ca} = 25$ kK, $\gamma_{0\mathrm{L}} = 1,55$, und $\Theta_{0\mathrm{L}} = 174$ kK modelliert. Die Parameter B_L und D_L können über das thermodynamische Gleichgewicht zur Gasphase und zum Festkörper approximiert werden. Das bedeutet, an der Stelle der kritischen Dichte ρ_crit muss der Druck

$$p_\mathrm{t} = -\left(\frac{\partial F_\mathrm{t}}{\partial v}\right) \tag{118}$$

für jede Temperatur $T \geq T_\mathrm{crit}$ dem Gasdruck entsprechen. Außerdem muss der stetige Übergang von p_t zum Festkörperdruck entlang der Schmelzkurve erfüllt sein. Insgesamt ist also das Gleichungssystem

$$p_\mathrm{t}(v_\mathrm{crit},\ T \geq T_\mathrm{crit},\ B_\mathrm{L},\ D_\mathrm{L}) = p^{(\mathrm{G})}(v_\mathrm{crit},\ T \geq T_\mathrm{crit}) \tag{119}$$

$$p_\mathrm{t}(v,\ T_\mathrm{m}(v)_{p_\mathrm{m}=p_\mathrm{c}+p_\mathrm{ph}+p_\mathrm{e}},\ B_\mathrm{L},\ D_\mathrm{L}) = p_\mathrm{ph}^{(\mathrm{S})}(v,\ T_\mathrm{m}(v)_{p_\mathrm{m}=p_\mathrm{c}+p_\mathrm{ph}+p_\mathrm{e}}) \tag{120}$$

numerisch nach den Parametern B_{L} und D_{L} aufzulösen. Trotz der Stetigkeit zum Festkörperdruck an der Schmelzkurve ist mit p_{t} die Stetigkeit des Druckes der Flüssigphase an der Grenze zur Schmelzzone nicht erfüllt. Der Grund besteht darin, dass der Schmelzdruck wesentlich steiler mit der Temperatur ansteigt, als der Festkörperdruck. In [22] wird deshalb der Ausdruck

$$F_{\mathrm{m}} = 3R\left(\frac{2(\frac{\sigma}{\sigma_{\mathrm{m0}}})^2 T_{\mathrm{m0}}}{1+(\frac{\sigma}{\sigma_{\mathrm{m0}}})^3}\left(C_{\mathrm{m}}+\frac{2A_{\mathrm{m}}}{5}\left(\left(\frac{\sigma}{\sigma_{\mathrm{m0}}}\right)^{5/3}-1\right)\right)+(B_{\mathrm{m}}-C_{\mathrm{m}})T\right) \tag{121}$$

mit $\sigma_{\mathrm{m0}} = 0,873$ als Korrekturterm für den stetigen Übergang zur Schmelzzone modelliert.

Bei Berechnung der dazugehörigen Druckkomponente über die analytische Volumenableitung

$$p_{\mathrm{m}}^{(\mathrm{L})} = -\left(\frac{\partial F_{\mathrm{m}}}{\partial v}\right) \tag{122}$$

entfällt der Parameter B_{m}, so dass im Folgenden die Parameter A_{m} und B_{m} approximiert werden.

Die erste Zwangsbedingung für den volumenabhängigen Verlauf von $p_{\mathrm{m}}^{(\mathrm{L})}$ besteht darin, dass der stetige Übergang zur Gasphase bereits über die Parameter in p_{t} konstruiert wurde, womit der Druck $p_{\mathrm{m}}^{(\mathrm{L})}$ an der Stelle ρ_{c} bzw. v_{c} verschwinden muss.

Desweiteren treffen die Binodale und die Übergangskurve zwischen Schmelzzone und Flüssigphase an der bereits konstruierten isothermen Kompressionskurve $T(\rho) = T_{\mathrm{m0}}$ zusammen (siehe Abb. 8.12). An diesem Punkt $p(v_{\mathrm{m\ bin}}, T_{\mathrm{m0}})$ liegt nach Gl. 100 Atmosphärendruck vor.

Numerisch gelöst wird somit das Gleichungssystem

$$p_{\mathrm{m}}^{(\mathrm{L})}(A_{\mathrm{m}}, C_{\mathrm{m}}, v = v_{\mathrm{c}}) = 0 \tag{123}$$

$$p_{\mathrm{m}}^{(\mathrm{L})}(A_{\mathrm{m}}, C_{\mathrm{m}}, v_{\mathrm{m\ bin}}) + p_{\mathrm{t}}(v_{\mathrm{m\ bin}}, T_{\mathrm{m0}}) + p_{\mathrm{e}}(v_{\mathrm{m\ bin}}, T_{\mathrm{m0}}) + p_{\mathrm{c}}(v_{\mathrm{m\ bin}}, T_{\mathrm{m0}}) - p_{\mathrm{m}}(v_{\mathrm{m\ bin}}, T_{\mathrm{m0}}) = 0. \tag{124}$$

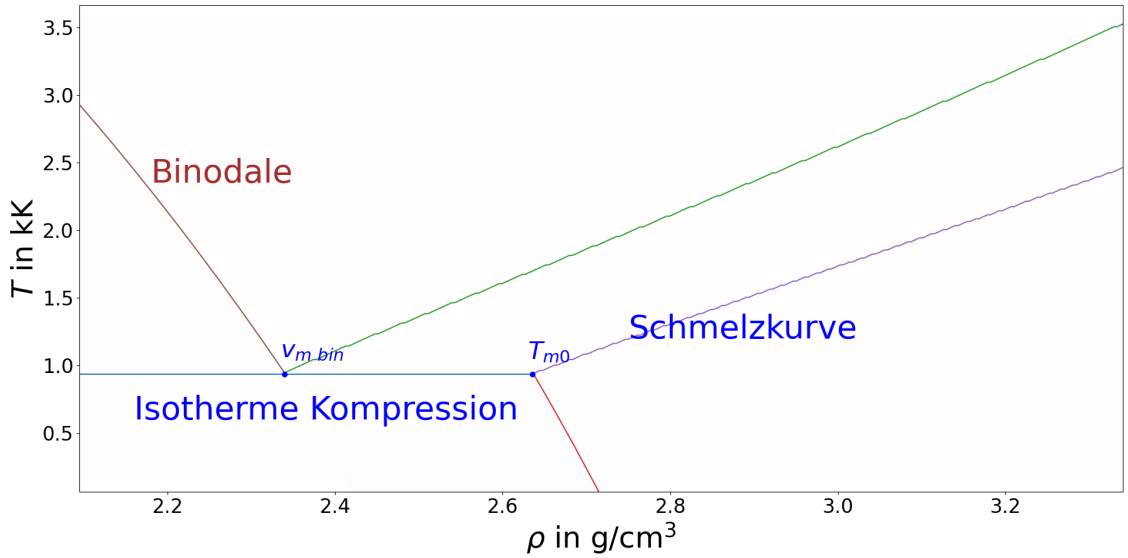

Abbildung 8.12: Ausschnitt aus der Konstruktion des Phasendiagramms für Aluminium. Dargestellt ist die Lage der isobaren Expansionskurve (rot), der Schmelzkurve (violett), der isothermen Kompression (blau), der Schnittkurve zwischen Schmelzzone und Flüssigphase (grün) sowie der Binodale (braun).

9 Vollständige Darstellung der Zustandsgleichungen

9.1 Zustandsgleichung für Aluminium

Auf Grundlage der modellierten Beziehungen zwischen Druck, Dichte und Temperatur sowie der Approximation der enthaltenen Parameter kann eine vollständige, in einem weiten Parameterbereich gültige Zustandsgleichung für die verschiedenen Aggregatzustände von Aluminium implementiert werden. Eine Übersicht der approximierten Parameter ist in Tabelle 9.1 abgebildet.

Gl. 77	a_1 -1402,15	a_2 2753,75	a_3 15,32	a_4 -2915,19	a_5 1548,27	
Gl. 84	A_c -54,55	B_c 62,57	C_c -8,01	m 1,03	n 2,84	l 5,54
Gl. 97, 75	B_s 0,54	D_s 0,69	γ_{e0} 0,38			
Gl. 100 Gl. 103, 105	A 6,83	B 0.56	q 0,121	$K_0^{'}$ 38,85	K_0 8,95	
Gl. 114-118 Gl. 121	B_L 1000	D_L 1000	A_m 12,5	C_m -2,5		

Tabelle 9.1: Übersicht über die approximierten Parameter der Zustandsgleichungen für Aluminium. Dargestellt sind die Parameter für (von oben nach unten) Bindungspotential des Kompressionsbereiches, Bindungspotential des Ausdehnungsbereiches, phononische und elektronische Grüneisenparameter, Schmelzzone und Flüssigphase.

Abb. 9.1 stellt den Druck zusammen mit den in den vorherigen Abschnitten ebenfalls konstruierten Phasengrenzlinien für eine Dichte bis $3,3\frac{\mathrm{g}}{\mathrm{cm}^3}$ und eine Temperatur bis 10 kK dar. Eingetragen sind die Bereiche der stabilen Festphase (S), der Schmelzzone (S+L), der stabilen Gasphase (G) und der stabilen Flüssigphase (L). Ebenfalls dargestellt sind die metastabilen Phasenbereiche (in eckigen Klammern stehend). Die metastabile Festphase [S], die metastabile Flüssigphase [L] und der metastabile Fest-Flüssig-Bereich [S+L] bilden jeweils die Verlängerung der Berechnung von Festkörper-, Flüssig- und Schmelzdruck in den negativen Druckbereich hinein. Negative Werte für den Druck bedeuten, dass die Kraftwirkung parallel und gleichgerichtet zur äußeren Oberflächennormale eines Volumenelementes verläuft, während die Kraftwirkung positiver Drücke antiparallel dazu, also in das Volumenelement hinein gerichtet ist.

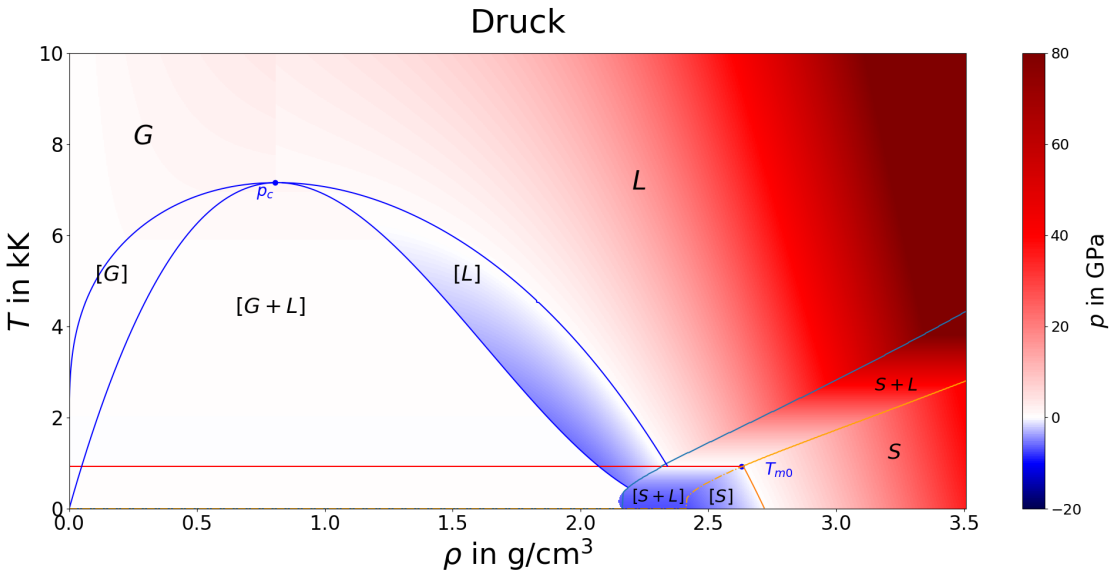

Abbildung 9.1: Vollständig implementierte Zustandsgleichung für Aluminium.

Die Arbeit [22], aus der Teile der Modellierungen für die Zustandsgleichung entnommen wurden, stellt als Ergebnis eine grob aufgelöste, tabellarische Lösung für den Druck in Abhängigkeit von Dichte und Temperatur innerhalb des Parameterbereiches bis 3,5 g/cm^3 und 10 kK dar (Tabelle 9.2).

ρ/T	0.3	0.99	1.7	4.2	3.1	3.8	4.5	5.2	5.8	6.5	7.2	7.9	8.6	9.3	10
10^{-3}	10^{-4}	$1.4\,10^{-4}$	0.0014	0.0057	0.0092	0.014	0.019	0.025	0.032	0.039	0.048	0.057	0.067	0.078	0.09
0.25	10^{-4}	0.00018	0.0017	0.0077	0.025	0.066	0.15	0.3	0.57	0.74	0.9	1.1	1.2	1.4	1.6
0.5	10^{-4}	0.00015	0.0018	0.0084	0.026	0.067	0.15	0.3	0.57	0.96	1.3	1.7	2.0	2.3	2.7
0.75	10^{-4}	0.00014	0.002	0.0084	0.027	0.069	0.15	0.3	0.57	0.96	1.5	2.0	2.5	3.0	3.6
1.0	10^{-4}	0.00014	0.0019	0.0084	0.027	0.068	0.15	0.3	0.57	0.96	1.5	2.2	3.0	3.7	4.5
1.3	10^{-4}	0.00014	0.0019	0.0085	0.026	0.068	0.15	0.3	0.57	0.96	1.9	2.9	3.8	4.8	5.8
1.5	10^{-4}	0.00014	0.0019	0.0079	0.026	0.067	0.15	0.3	0.72	1.9	3.1	4.3	5.6	6.8	8.0
1.8	10^{-4}	0.00014	0.0017	0.0078	0.025	0.065	0.15	1.3	2.7	4.2	5.7	7.1	8.6	10.0	11.0
2.0	10^{-4}	0.00014	0.0016	0.0074	0.024	1.1	2.9	4.7	6.4	8.1	9.8	12.0	13.0	15.0	17.0
2.3	10^{-4}	0.00014	0.0014	1.8	3.9	6.0	8.0	10.0	12.0	14.0	16.0	18.0	20.0	22.0	24.0
2.5	10^{-4}	3.4	5.9	8.3	11.0	13.0	15.0	17.0	20.0	22.0	24.0	26.0	28.0	31.0	33.0
2.8	1.1	4.9	14.0	17.0	19.0	22.0	25.0	27.0	29.0	32.0	34.0	37.0	39.0	42.0	44.0
3.0	9.3	13.0	17.0	28.0	31.0	33.0	36.0	39.0	42.0	44.0	47.0	49.0	52.0	55.0	57.0
3.3	20.0	24.0	28.0	32.0	44.0	47.0	50.0	53.0	56.0	59.0	62.0	64.0	67.0	70.0	73.0
3.5	33.0	37.0	41.0	45.0	60.0	63.0	66.0	69.0	72.0	76.0	79.0	82.0	85.0	88.0	91.0

Tabelle 9.2: Tabellarische Zustandsgleichung für den Druck in Abhängigkeit von der Dichte und der Temperatur aus der Arbeit [22]. Der Druck ist in GPa angegeben, die Dichte ρ in g/cm^3 und die Temperatur in kK.

Die Betrachtung der darin wiedergegebenen Ergebnisse zeigt zunächst das Fehlen der in den metastabilen Phasenbereichen auftretenden negativen Druckwerte. Ein Vergleich des in dieser Arbeit berechneten Druckverlaufes aus Abb. 9.1 mit den Werten an entsprechenden T-ρ-Punkten aus Tabelle 9.2 zeigt jedoch gute Übereinstimmungen in den stabilen Bereichen von Fest-, Flüssig- und Gasphase, sowie im Bereich der Schmelzzone.

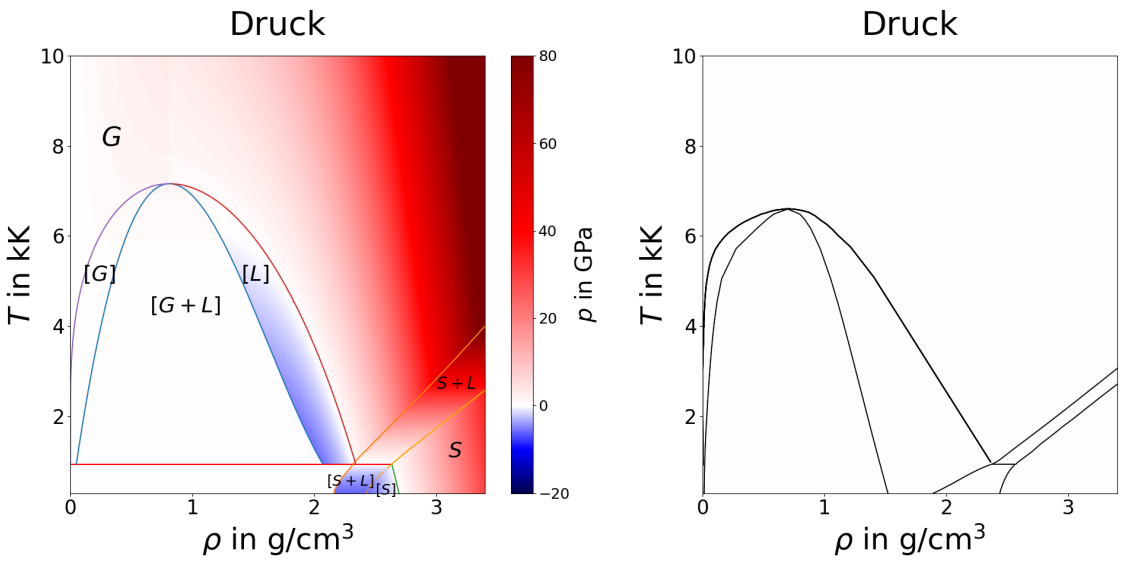

Abbildung 9.2: Vergleich der in dieser Arbeit konstruierten Phasengrenzlinien für Aluminium (links) mit dem in [48] dargestellten Phasendiagramm (rechts).

In der Arbeit [48] werden darüber hinaus die Phasenübergänge von Aluminium unter Einfluss ultrakurzer Laserpulse in einem Phasendiagramm dargestellt. Dieses ist in Abb. 9.2 (rechts) abgebildet und dem aus den Zustandsgleichungen dieser Arbeit resultierenden Phasendiagramm (links) gegenübergestellt. Ein Vergleich der Phasengrenzlinien zeigt zunächst eine sehr gute Übereinstimmung der Lage von stabiler bzw. metastabiler Festphase im ρ-T-Diagramm. Die Schmelzkurve zeigt in beiden Diagrammen einen sehr ähnlichen Verlauf, allerdings ist der Schmelzbereich in [48] deutlich schmaler berechnet. Desweiteren zeigen Binodale und Spinodale relativ ähnliche Verläufe, womit sich auch ähnliche Lagen der verschiedenen stabilen und metastabilen Flüssig- und Gasphasen ergeben. Allerdings endet die Binodale aus [48] aufgrund der schmaler berechneten Schmelzzone bei geringfügig höheren Dichten am Grenzpunkt zwischen Flüssig- und Schmelzbereich. Da die Spinodale dagegen einen deutlich steileren Abfall zeigt, als der in Abschnitt 8.4 hergeleitete analytische Verlauf (Gl. 108), ergibt sich insgesamt eine Verbreiterung der metastabilen Gasphase zwischen Spinodale und Binodale im Diagramm aus [48] im Vergleich zum Diagramm links.

9.2 Zustandsgleichung für Gold

Die in den vorherigen Abschnitten dargestellten Parameterapproximationen wurden anschließend auf das Materialbeispiel Gold angewendet. Die berechneten Werte sind in Tabelle 9.3 zusammengefasst.

Gl. 77	a_1 -2927,54	a_2 6346,41	a_3 -324,58	a_4 -7237,23	a_5 4142,95	
Gl. 84	A_c 262,87	B_c -131,43	C_c -131,43	m 2,54	n 1,83	l 1,84
Gl. 97, 75	B_s 5,30e-7	D_s 1,02	γ_{e0} 0,095			
Gl. 100 Gl. 103, 105	A 14,65	B 0,56	q 1,22	$K_0^{'}$ 126,37	K_0 0,59	
Gl. 114-118 Gl. 121	B_L 0,23	D_L 1,0e-6	A_{m} 0,5	C_{m} 0,75		

Tabelle 9.3: Übersicht über die approximierten Parameter der Zustandsgleichungen für Gold. Dargestellt sind die Parameter für (von oben nach unten) Bindungspotential des Kompressionsbereiches, Bindungspotential des Ausdehnungsbereiches, phononische und elektronische Grüneisenparameter, Schmelzzone und Flüssigphase.

Im Gegensatz zum Beispiel Aluminium sind bei der Zusammensetzung der Zustandsgleichung für Gold einige Korrekturen notwendig. Als erstes ist hier festzustellen, dass der Verlauf von Binodale und Spinodale nach der Maxwell-Konstruktion in Abschnitt 8.4 lediglich von den kritischen Werten p_{crit}, T_{crit}, ρ_{crit} des Materials abhängt. Wie dort bereits erwähnt, zeigen die in der Literatur dargestellten, bzw. ermittelten kritischen Werte zum Teil jedoch erhebliche Abweichungen. Trotz einer Reihe theoretischer und messtechnischer Ermittelungen des kritischen Punktes für Gold, führt keiner dieser Literaturwerte zu einem Verlauf der Binodale, der an der Schmelzzone endet. Um die in Abb. 8.12 dargestellte Bedingung zu erfüllen, nach der sich Binodale und isotherme Kompression an der Grenze zur Schmelzzone schneiden, wurde ein kritischer Punkt (T_{crit}, ρ_{crit}) so konstruiert, dass die daraus resultierende Binodale dieser Bedingung genügt. Abb. 9.3 zeigt die vollständige Implementierung der Zustandsgleichung für Gold zusammen mit den Phasengrenzlinien. Dargestellt sind weiterhin der konstruierte kritische Punkt (blau), sowie die Literaturwerte des kritischen Punktes aus [47], [49] und [50]. Der konstruierte kritische Punkt liegt dabei in der Nähe des Schwerpunktes der Literaturwerte.

Eine weitere Korrektur ist notwendig, da der Festkörperdruck deutlich zu hoch berechnet wird, so dass keine Schmelzkurve über die thermische Gleichgewichtsbedingung zur Schmelzzone konstruiert werden kann und die analytisch berechnete isobare Ex-

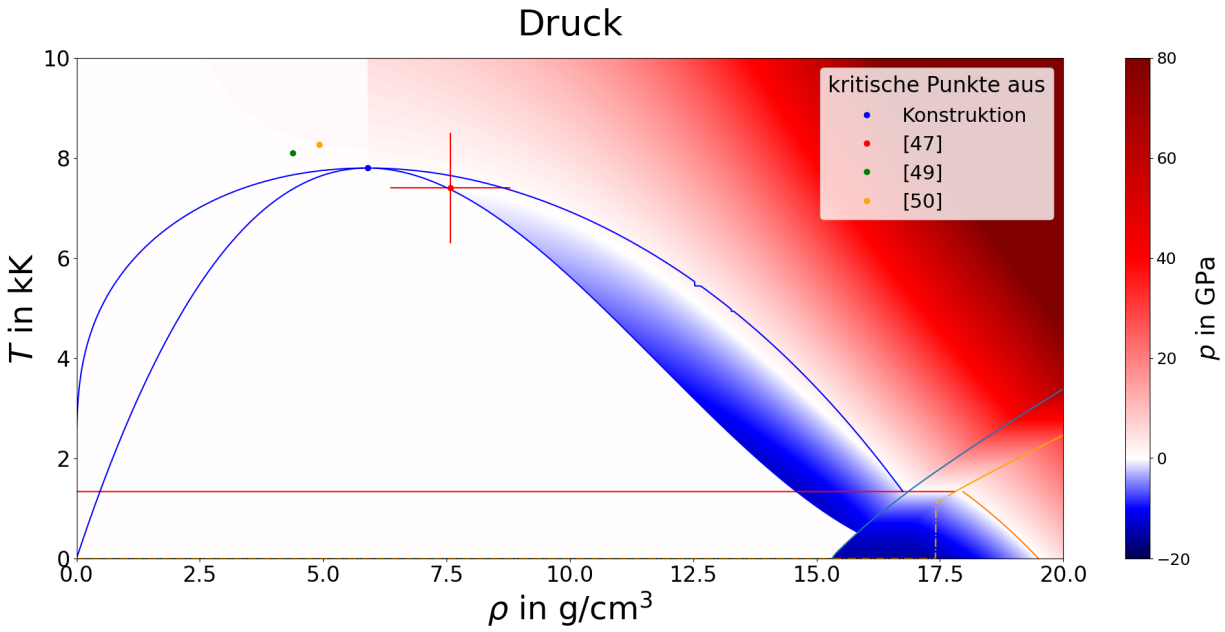

Abbildung 9.3: Vollständig implementierte Zustandsgleichung für Gold.

pansionskurve durch einen wesentlich höheren Druck, als den Normaldruck verläuft. Beide Kurven zeigen genau dann den Abb. 9.3 dargestellten, akkuraten Verlauf, wenn der aus dem Grüneisenmodell berechnete Phononendruck $p_{\mathrm{ph}}^{(S)}$ auf $0.4 p_{\mathrm{ph}}^{(S)}$ reduziert wird. In diesem Fall treffen Schmelz- und isobare Expansionskurve bei der Temperatur T_{m0} aufeinander.

Ein Mangel, für den keine Korrektur erreicht werden konnte, liegt an der Grenze zwischen der Flüssig- und Gasphase. Während dieser Übergang bei Aluminium annähernd stetig verläuft, zeigt er bei Gold eine deutlich ausgeprägtere Sprungstelle, die auch mit geringeren Toleranzen bei der Parameterapproximation nicht weiter reduziert werden kann.

Die Arbeit [51] zeigt im Zuge von Untersuchungen der Laser-Ablation ein Phasendiagramm für Gold, das wie bereits oben bei Aluminium vergleichend den Ergebnissen der Modellierungen dieser Arbeit gegenübergestellt wird. Abb. 9.4 zeigt links das aus den Zustandsgleichungen dieser Arbeit resultierende Phasendiagramm, sowie rechts die Phasengrenzlinien aus [51].

Zunächst ist wieder die gute Übereinstimmung bei den Lagen der stabilen und metastabilen Festphasengebiete zu erkennen, die durch die thermische Expansionskurve getrennt sind. Jedoch fällt bei den Modellierungen dieser Arbeit, genau wie am Beispiel Aluminium, die Breite der Schmelzzone geringer aus, als bei der Referenz. Demgegenüber zeigt sich auch hier ein schmaler berechneter Bereich der metastabilen Flüssiphase, während in [51] der steilere Abfall der Spinodale zu einem breiteren metastabilen Flüssigphasenbereich führt. Die Lagen der stabilen und metastabilen Gasphasen, sowie der stabilen Flüssigphase ist in beiden Diagrammen wiederum ähnlich.

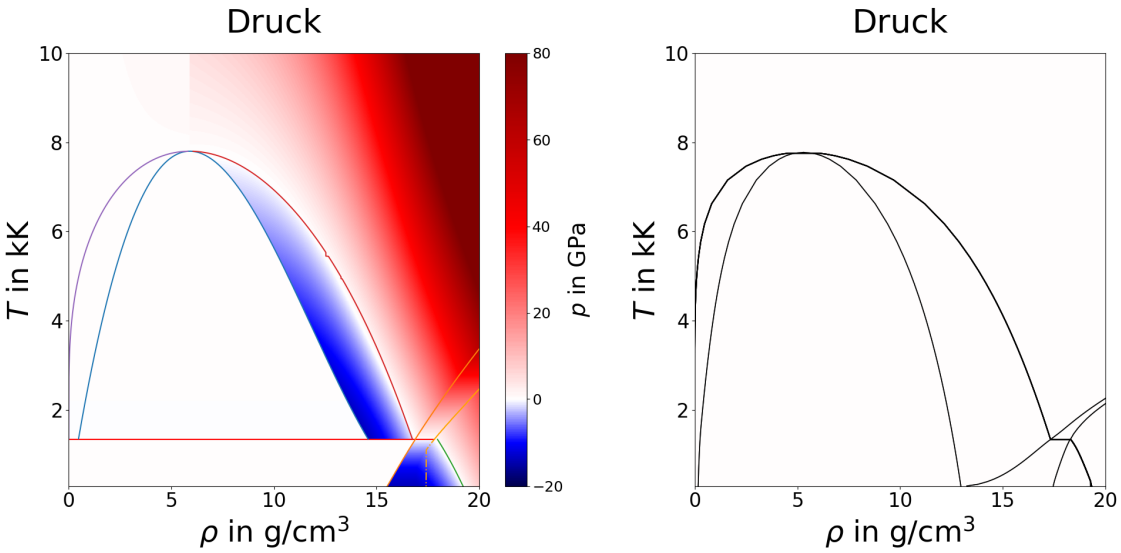

Abbildung 9.4: Vergleich der in dieser Arbeit konstruierten Phasengrenzlinien für Gold (links) mit dem in [51] dargestellten Phasendiagramm (rechts).

10 Lösung des Advektions-Diffusions-Modells mit implementierten Zustandsgleichungen

Zur Implementierung der Zustandsgleichungen in das Advektions-Diffusionsmodell ist die Separierung des Druckes in den Elektronen- und den Gitterdruck notwendig. Letzterer setzt sich aus der Druckkomponente der Phononen p_{ph} und der des Bindungspotentials p_{c} zusammen. Der Druck des Elektronensystems wurde bereits separat in Abschnitt 8.1 modelliert (siehe Abb. 8.3). Dieser wird vom Gesamtdruck (Abb. 9.1) abgezogen, um die Druckkomponente des Ionengitters für jeden der einzelnen Phasenbereiche zu erhalten.

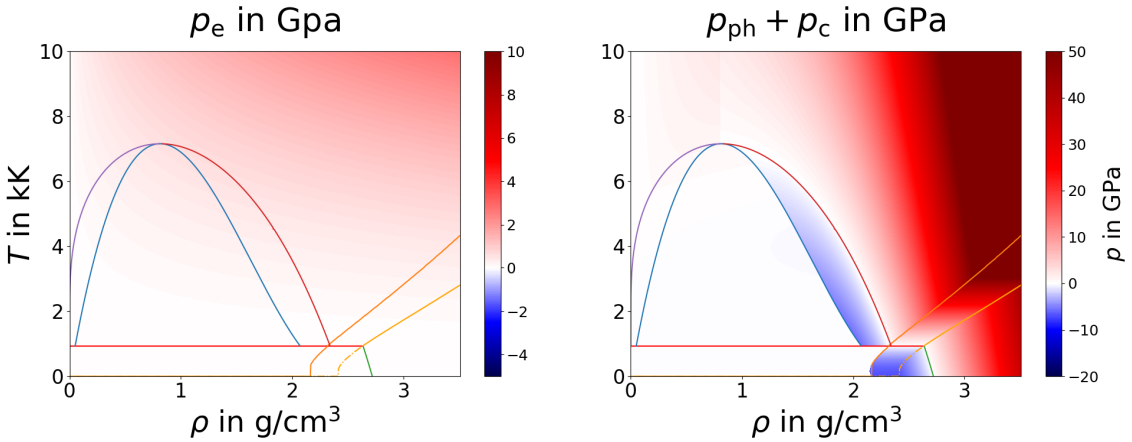

Abbildung 10.1: Darstellung der Druckkomponenten des Elektronensystems (links) und des Ionengitters (rechts) der verschiedenen Phasengebiete für Aluminium.

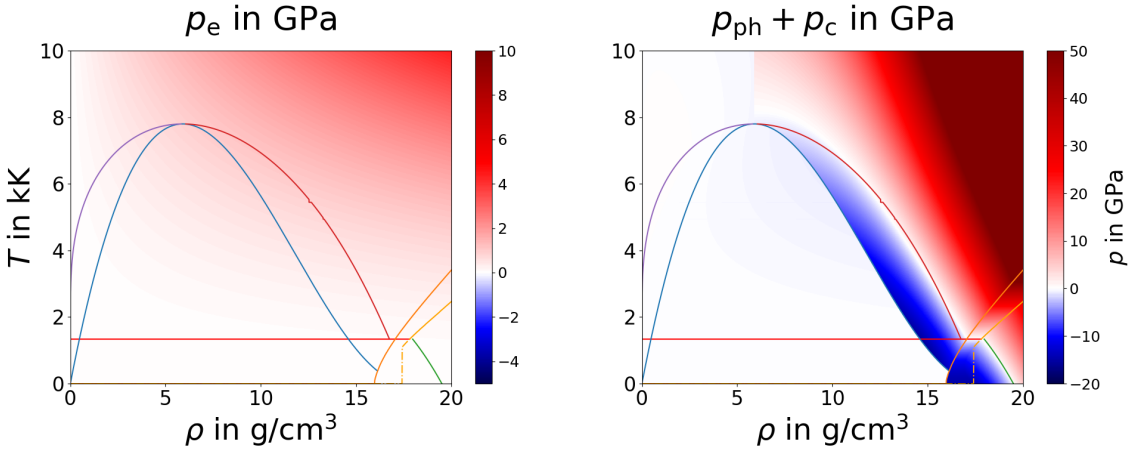

Abbildung 10.2: Darstellung der Druckkomponenten des Elektronensystems (links) und des Ionengitters (rechts) der verschiedenen Phasengebiete für Gold.

Abb. 10.1 und 10.2 zeigen die beiden Druckkomponenten p_e und $p_\mathrm{ph} + p_\mathrm{c}$ in den Phasendiagrammen für Aluminium und Gold. Elektronen- und Gitterdruck wurden schließlich anstelle der Zustandsgleichungen des idealen Gases in die Energiegleichungen des Elektronen- bzw. Phononensystems des Advektions-Diffusions-Modells in Lagrange-Koordinaten integriert.

10.1 Anregung von Aluminium

Die anschließende Berechnung der Zustandsgrößen wurde mit den selben Laserparametern durchgeführt, wie in Abschnitt 3, jedoch mit den thermischen Materialparametern von Aluminium. Da die Schallgeschwindigkeit von Aluminium etwa das doppelte der Schallgeschwindigkeit von Gold beträgt, wurde die Schichttiefe hier ebenfalls auf 400 nm verdoppelt, um eine vergleichbare Ausbreitung der Dichtewelle beobachten zu können. Die in Abb. 10.3 dargestellten Lösungen wurden wieder bis zu einer Maximalzeit von 200 ps berechnet und zeigen zunächst die energetische Kopplung zwischen Elektronen- und Phononensystem sowie eine durch den Temperaturgradienten induzierte Druckwelle, die mehrmals an den Schichtgrenzen reflektiert wird. Der aus dem Druckgradienten hervorgehende Impuls bewirkt auch in diesem Falle einen Massetransport und somit eine Dichtewelle, die dem Verlauf der Druckwelle folgt.

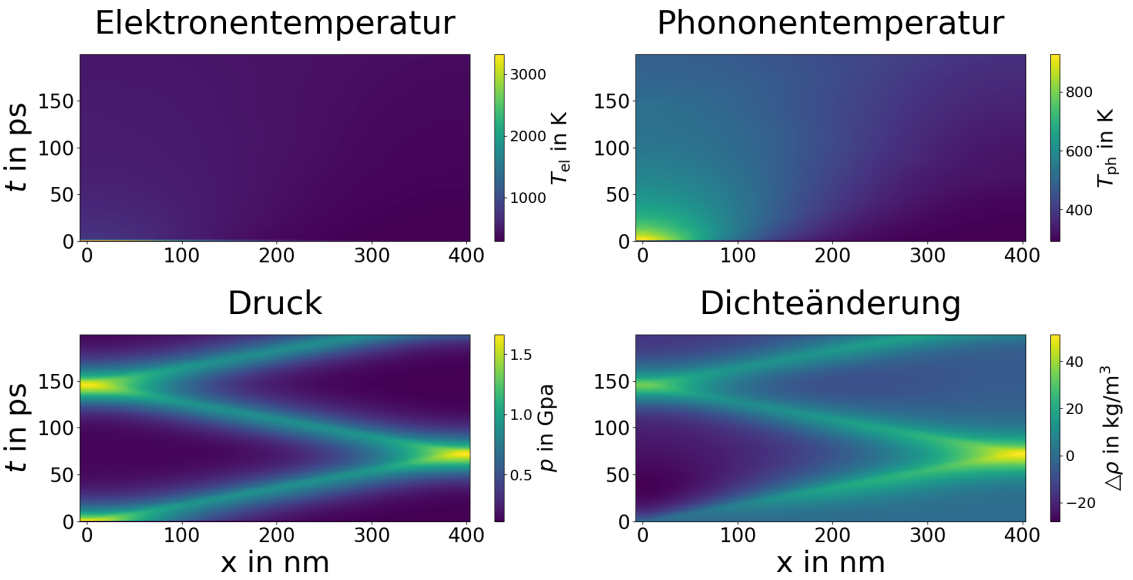

Abbildung 10.3: Lösungen des Advektions- Diffusionsmodells in Lagrange-Koordinaten mit Implementierung der Zustandsgleichung für Aluminium. Berechnet wurden die Zustandsgrößen Elektronen- und Phononentemperatur, Druck und Dichte für den Zeitbereich bis $t = 200$ ps.

Eine zweite Simulation wurde mit gleichen Laser- und Materialparametern, jedoch mit einer zentrierten Anregung in der Mitte eines spiegelsymmetrischen Definitionsbereiches von -400 nm bis 400 nm durchgeführt. Die Darstellung der berechneten Lösung in

Abb. 10.4 zeigt die erwarte, spiegelsymmetrische Verteilung der Zustandsgrößen. Das bedeutet insbesondere die symmetrische Ausbreitung der Phononentemperatur, sowie eine daraus resultierende Ausbreitung zweier gegenläufiger Druck bzw. Dichtewellen.

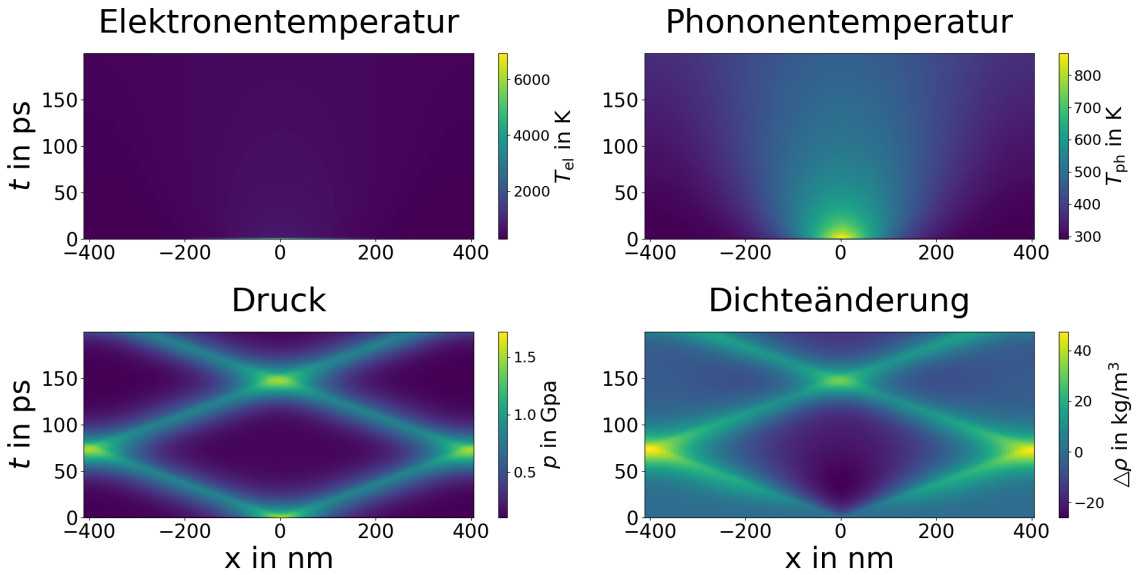

Abbildung 10.4: Lösungen des Advektions- Diffusionsmodells in Lagrange-Koordinaten mit Implementierung der Zustandsgleichung für Gold. Die Laserstrahlung trifft hier auf die Mitte des Definitionsbereiches.

10.2 Anregung von Gold

Zuletzt wird die Zustandsgleichung für Gold auf die gleiche Weise wie für Aluminium in das Lagrange-Modell integriert und die Zustandsgrößen bis zu einer Maximalzeit von 200 ps in einer Schicht der Tiefe 200 nm berechnet (siehe Abb. 10.5). Der Vergleich mit der in Abschnitt 3 durchgeführten Simulation zeigt hier eine deutlich schnellere Ausbreitung der Druck- bzw. Dichtewelle und eine mehrmalige Reflexion an den Schichtgrenzen. Ebenfalls auffällig sind die höheren Amplituden der Stoßwellen. Da der Druck einen skalierenden Faktor im Quellterm der Energieadvektion darstellt, macht sich die höhere Amplitude der Druckwelle wiederum in einer der Druckwelle folgenden Erhöhung der Phononentemperatur als Resultat der angeregten Energieadvektion bemerkbar.

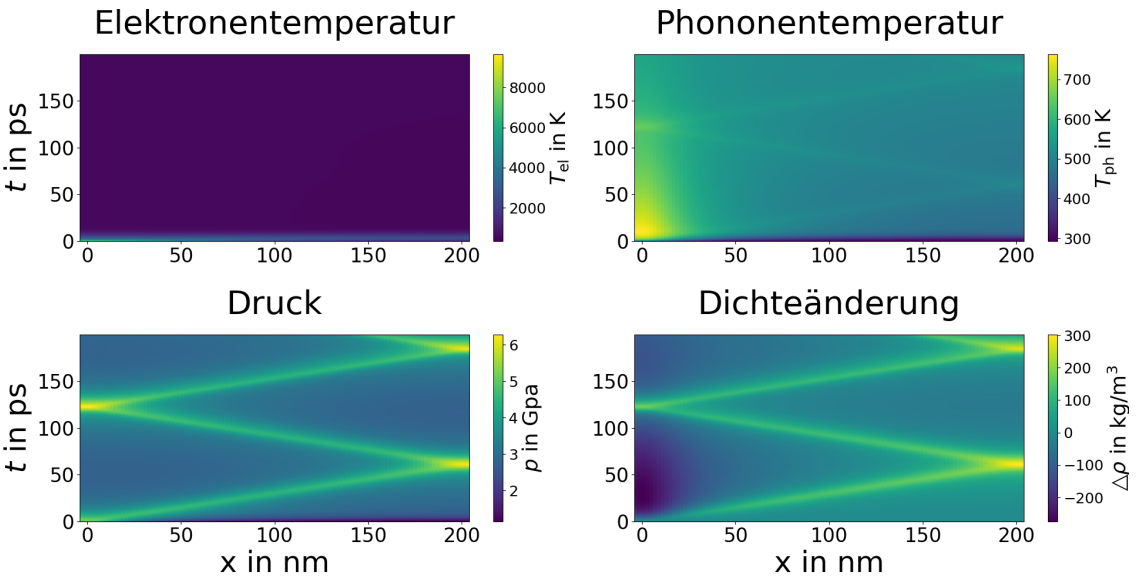

Abbildung 10.5: Lösungen des Advektions- Diffusionsmodells in Lagrange-Koordinaten mit Implementierung der Zustandsgleichung für Gold. Berechnet wurden die Größen Elektronen- und Phononentemperatur, Druck und Dichte für den Zeitbereich bis $t = 200$ ps.

Dass die Zustandsgleichung für Gold die thermodynamischen Eigenschaften des Metalls zumindest im Rahmen der Advektions-Diffusions-Simulation hinreichend genau wiedergibt, zeigt sich bei Betrachtung des Laufweges der Stoßwelle. Der Literaturwert der Schallgeschwindigkeit für Gold beträgt 3240 m/s. Die Laufstrecke der in Abb. 10.5 dargestellten Stoßwelle von etwa 650 nm innerhalb der Berechnungszeit von 200 ps ergibt eine Laufgeschwindigkeit von 3250 m/s, also eine sehr gute Übereinstimmung.

Analog wird auch für Gold eine Simulation mit zentrierter Anregung auf einem Definitionsbereich von -400 nm bis 400 nm unter Verwendung gleicher Parameter berechnet. In Abb. 10.6 ist die Zeitentwicklung der betrachteten Zustandsgrößen dargestellt. Wie erwartet, gleicht das Ergebnis dem aus Abb. 10.5, zeigt jedoch eine symmetrische Dynamik der Zustandsgrößen mit zwei entgegengesetzt verlaufenden Druck- bzw. Dichtewellen.

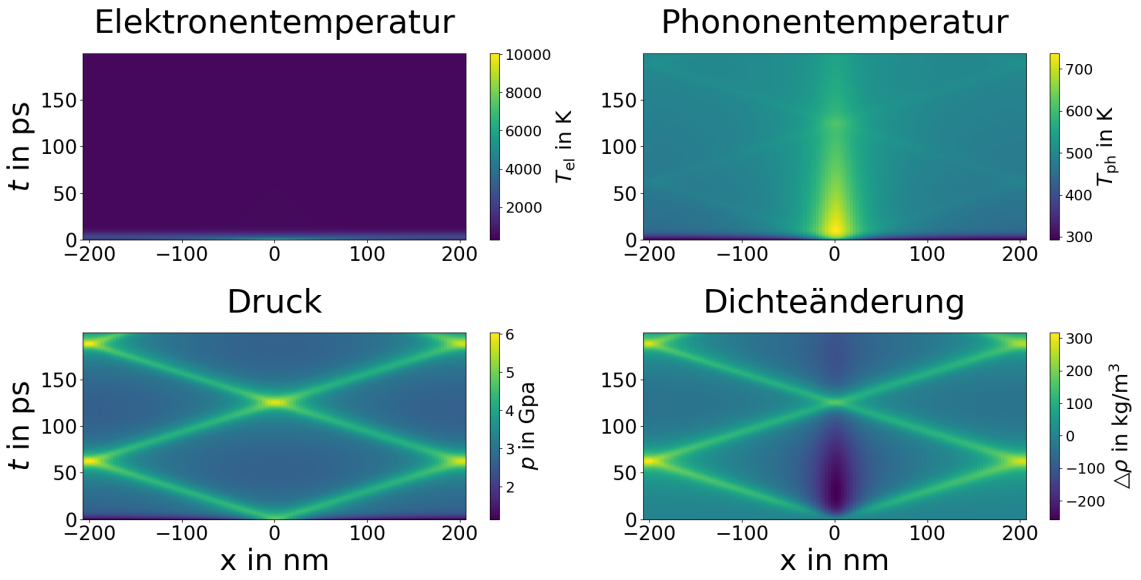

Abbildung 10.6: Lösungen des Advektions- Diffusionsmodells in Lagrange-Koordinaten mit Implementierung der Zustandsgleichung für Gold. Die Laserstrahlung trifft hier auf die Mitte des Definitionsbereiches.

11 Zusammenfassung

Ausgehend von der Analyse eines Advektions-Diffusionsmodells in Euler-Koordinaten wurde im ersten Teil dieser Arbeit der Übergang in Lagrange-Koordinaten dargestellt. Dieser basiert auf Trajektorien, die auf dem Festkörper fixierte Punkte bei laserinduzierten Advektionsprozessen zurücklegen, und die als bewegliches Koordinatengitter dienen. In dem die Trajektorien dieser Lagrange-Punkte mit den Charakteristiken von Transportprozessen in Verbindung gebracht wurde, konnte auf Grundlage der Theorie der Charakteristiken eine Lösung des Advektionsproblems bezüglich des beweglichen Koordinatensystems hergeleitet werden. Der entscheidende Unterschied liegt darin, dass die Dynamik der Zustandsgrößen, die auf dem starren Koordinatengitter die Lösung numerisch anspruchsvoller Advektionsgleichungen erfordert, auf den beweglichen Lagrange-Koordinaten einem System gewöhnlicher Differenzialgleichungen genügt.

Das Advektionsmodell in Lagrange-Koordinaten wurde schließlich mit einem Zwei-Temperatur-Modell gekoppelt, welches eine Unterscheidung zwischen Elektronen- und Phononenenergie vornimmt. Um die Simulation an den Materialbeispielen Aluminium und Gold durchzuführen, wurden dabei die thermischen Eigenschaften der Materialien, d. h. die elektronischen und phononischen Wärmekapazitäten, die elektronische Wärmeleitfähigkeit sowie der energetische Kopplungskoeffizient zwischen Elektronen- und Phononensystem ermittelt. Da das Modell in Euler-Koordinaten bereits als vollständige Implementierung für Gold vorliegt, wurde das entwickelte Lagrange-Modell für einen Vergleich zunächst ebenfalls für Gold implementiert. Der Vergleich beider Modellberechnungen bei gleichen Laserparametern zeigt insgesamt eine sehr gute Übereinstimmung der Lösungen, wobei hervorzuheben ist, dass mit dem Übergang in Lagrange-Koordinaten die Lösung mit deutlich geringeren Rechenzeiten erreicht werden konnte. Die Berechnungen zeigen zum einen die energetische Kopplung zwischen Elektronen- und Phononensystem, zum anderen eine durch den Druckgradienten induzierte Dichtewelle, die der Ausbeitung der Druckwelle durch den Festkörper folgt.

Während die Druckberechnungen dieser Lösungen auf der Zustandsgleichung des idealen Gases basieren, wurden im zweiten Teil der Arbeit Zustandsgleichungen für Aluminium und Gold auf Grundlage von Modellierungen in [22] konstruiert. Da diese Zustandsgleichungen für einen weiten Parameterbereich gültig sein müssen, wurden Abhängigkeiten des Druckes von Dichte und Temperatur für die verschiedenen Aggregatzustände separat betrachtet und durch Konstruktion der Phasengrenzlinien schließlich zusammengesetzt. Eine Reihe materialabhängiger Parameter wurde dabei aus theoretischen Modellierungen sowie experimentellen Messwerten approximiert. Während das Phasendiagramm von Aluminium gut mit der Darstellung aus [22] übereinstimmt, zeigt die Zustandsgleichung von Gold an einigen Stellen Abweichungen zu den erwarteten des Druckverläufen.

Die Zustandsgleichungen beider Materialien wurden schließlich in das Lagrange-Modell implementiert, womit die thermodynamischen Eigenschaften der Materialien in die Simulation der laserinduzierten Advektions-Diffusions-Prozesse integriert werden konnten. Die Ergebnisse der Berechnungen zeigen für beide Materialien die erwartete Dynamik der Zustandsgrößen. Dargestellt werden konnte auch hier die Elektronen- Phononenkopplung, sowie die Induzierung von Stoßwellen, die den Festkörper durchlaufen. Im Gegensatz zu den vorherigen Simulationen mit der Zustandsgleichung des idealen Gases wird dabei die Schallgeschwindigkeit in den Metallen, also die Laufgeschwindigkeit der Stoßwellen korrekt abgebildet.

Literaturverzeichnis

[1] J. Bliedtner, C.Schindler, M. Seiler, S. Wächter, M. Friedrich, J. Giesecke. Ultrashort Pulse Laser Material Processing. Laser Technik Journal, 05-2016

[2] E Steigera, G Stumpf. New micro and macro laser machining applications using fiber laser and ultra short pulse laser
systems in compact laser machines. (2016) Published by Bayerisches Laserzentrum GmbH

[3] Hügel, H., Graf, T. (2022). Abtragende Verfahren. In: Materialbearbeitung mit Laser. Springer Vieweg, Wiesbaden.
https://doi.org/10.1007/978-3-658-37252-1_12

[4] Sugioka, K., Cheng, Y. Ultrafast lasers—reliable tools for advanced materials processing. Light Sci Appl 3, e149 (2014). https://doi.org/10.1038/lsa.2014.30

[5] Thompson, H., Lammatao, J., Hecht, M. D., Yousif, A., Campbell, B. R., & Picard, Y. N. (2014). Ultrashort Pulsed Laser Induced Heat Affected Zones Characterized by Ion Channeling Contrast Imaging. Microscopy and Microanalysis, 20(S3), 1480- 1481.

[6] HOPPIUS, J., et al. On Line Evaluation of Femtosecond Laser Ablation Efficiency on Copper Structures. Laser in
Manufacturing 2015: Proceeding of the German Scientific Laser Society, 2015, S. 22-25.

[7] Sergei I Anisimov and B S Luk'yanchuk. Selected problems of laser ablation theory. 2002 Phys.-Usp. 45 293

[8] R. Davydov, V. Antonov and M. Angelina, "Computer Simulation of Metal Ablation by Single and Multiple Ultrashort Laser Pulses," 2018 IEEE International Conference on Electrical Engineering and Photonics (EExPolytech), 2018, pp. 236-239, doi: 10.1109/EExPolytech.2018.8564378

[9] Ionin, A. A., Kudryashov, S. I., & Samokhin, A. A. (2017). Material surface ablation produced by ultrashort laser pulses. Physics-Uspekhi, 60(2), 149.

[10] M. Olbrich, P. Lickschat, L. Schneider, J. Schille, U. Löschner, S. Weißmantel, A. Horn. Simulation der Temperaturverteilungen in Gold und Platin infolge der Bestrahlung mit ultrakurzen Laserpulsen

[11] Herrich-Schäffer, L. 2022. Erstellung und Implementierung eines gekoppelten Advektions- Diffusionsmodells

[12] Kley, W. (11.2020). Numerische Hydrodynamik: Stoßrohr. https://www.tat.physik.uni-tuebin- gen.de/~kley/lehre/bhydro/vsol1.lst (Abgerufen am 20.4.2023)

[13] W. Arendt, K. Urban. Partielle Differenzialgleichungen. Spektrum Akademischer Verlag 2010

[14] https://www.ita.uni-heidelberg.de/~dullemond/lectures/num_fluid_2011/Chapter_1.pdf (Abgerufen am 5.7.2022)

[15] Rethfeld, B., Ivanov, D. S., Garcia, M. E., & Anisimov, S. I. (2017). Modelling ultrafast laser ablation. Journal of Physics D: Applied Physics, 50(19), 193001.

[16] Lin, Z., Zhigilei, L. V., & Celli, V. (2008). Electron-phonon coupling and electron heat capacity of metals under conditions of strong electron-phonon nonequilibrium. Physical Review B, 77(7), 075133.

[17] refractiveindex.info (Abgerufen am 10.4.2023)

[18] Hunklinger, S. (2018). Festkörperphysik. Berlin, Boston: De Gruyter Oldenbourg. https://doi.org/10.1515/9783110567755

[19] Balerna, A., & Mobilio, S. (1986). Dynamic properties and Debye temperatures of bulk Au and Au clusters studied using extended x-ray-absorption fine-structure spectroscopy. Physical Review B, 34(4), 2293.

[20] Petrov, Y. V., Inogamov, N. A., Khokhlov, V. A., & Migdal, K. P. (2021, February). Electron thermal conductivity of nickel and aluminum in solid and liquid phases in two-temperature states. In Journal of Physics: Conference Series (Vol. 1787, No. 1, p. 012025). IOP Publishing.

[21] Erven, J. & Horák, J. (2018). 8. Mehrdimensionale Analysis. In Mathematik für angewandte Wissenschaften: Ein Taschenbuch für Ingenieure und Naturwissenschaftler (pp. 145-166). Berlin, Boston: De Gruyter. https://doi.org/10.1515/9783110537161-155

[22] Redka, D. S. (2018). Investigation of a Multi-Phase Equation of State for Metals: Aluminum as Prototype

[23] Petrov, Y. V., Migdal, K. P., Inogamov, N. A. & Zhakhovsky, V. V. Two-temperature equation of state for aluminum and gold with electrons excited by an ultrashort laser pulse. Applied Physics B 119, 401–411 (2015)

[24] http://www.ihed.ras.ru. (Abgerufen am 2.4.2023)

[25] Skidmore, J. S. (1962). E Morris. Thermodynamics of Nuclear Materials, Vienna.

[26] Isbell, W. M., Shipman, F. H., & Jones, A. H. (1968). Hugoniot equation of state measurements for eleven materials to five megabars. GENERAL MOTORS TECHNICAL CENTER WARREN MI MATERIALS AND STRUCTURES LAB.

[27] McQueen, R. G., Marsh, S. P., Taylor, J. W., Fritz, J. N., & Carter, W. J. (1970). The equation of state of solids from shock wave studies. High velocity impact phenomena, 293, 294-417.

[28] L. V. Al'tshuler and B. F. Chekin, "Metrology of pulsed pressures," in: Reports of the First All-Union Symposium on Pulsed Pressures [in Russian], VNIIFTRI, Moscow (1974).

[29] Kormer, S. B., Funtikov, A. I., Urlin, V. D., & Kolesnikova, A. N. (1962). Dynamic compression of porous metals and the equation of state with variable specific heat at high temperatures. Sov. Phys. JETP, 15(3), 477-488.

[30] Al'Tshuler, L. V., Kalitkin, N. N., Kuz'mina, L. V., & Chekin, B. S. (1977). Shock adiabats for ultrahigh pressures. Sov. Phys. JETP, 45(1), 167-171

[31] Walsh, J. M., Rice, M. H., McQueen, R. G., & Yarger, F. L. (1957). Shock-wave compressions of twenty-seven metals. Equations of state of metals. Physical Review, 108(2), 196.

[32] Al'Tshuler, L. V., Krupnikov, K. K., & Brazhnik, M. I. (1958). Dynamic compressibility of metals under pressures from 400,000 to 4,000,000 atmospheres. Sov. Phys. JETP, 7, 614-619.

[33] McQueen, R. G., & Marsh, S. P. (1960). Equation of state for nineteen metallic elements from shock-wave measurements to two megabars. Journal of Applied Physics, 31(7), 1253-1269.

[34] Jones, A., Isbell, W. M., & Maiden, C. J. (1966). Measurement of the very-high-pressure properties of materials using a light-gas gun. Journal of Applied Physics, 37(9), 3493-3499.

[35] Marsh, S. P. (Ed.). (1980). LASL shock Hugoniot data. University of California press.

[36] Al'Tshuler, L. V., Bakanova, A. A., Dudoladov, I. P., Dynin, E. A., Trunin, R. F., & Chekin, B. S. (1981). Shock adiabatic curves of metals: New data, statistical analysis, and general laws. Journal of Applied Mechanics and Technical Physics, 22(2), 145-169.

[37] Grüneisen, E. (1926). Zustand des festen Körpers. Thermische Eigenschaften der Stoffe, 1-59.

[38] Katsnelson, M. I. (2005). Lattice dynamics: Anharmonic effects. Encyclopedia of Condensed Matter Physics (Elsevier, Amsterdam etc., 2005), ed. by GF Bassani, GL Liedl, and P. Wyder, 77.

[39] Al'Tshuler, L. V., Brusnikin, S. E., & Kuz'Menkov, E. A. (1987). Isotherms and Grüneisen functions for 25 metals. J. Appl. Mech. Tech. Phys.(Engl. Transl.);(United States), 28(1).

[40] Sutton, P. M. (1953). The variation of the elastic constants of crystalline aluminum with temperature between 63 K and 773 K. Physical Review, 91(4), 816.

[41] Hänström, A., & Lazor, P. (2000). High pressure melting and equation of state of aluminium. Journal of alloys and compounds, 305(1-2), 209-215

[42] Errandonea, D. (2010). The melting curve of ten metals up to 12 GPa and 1600 K. Journal of Applied Physics, 108(3), 033517.

[43] Hieu, H. K., & Ha, N. N. (2013). High pressure melting curves of silver, gold and copper. AIP Advances, 3(11), 112125.

[44] Martin, C. J., & O'connor, D. A. (1977). An experimental test of Lindemann's melting law. Journal of Physics C: Solid State Physics, 10(18), 3521.

[45] Wang, Y., Ahuja, R., & Johansson, B. (2001). Melting of iron and other metals at earth's core conditions: A simplified computational approach. Physical Review B, 65(1), 014104.

[46] Vinet, P., Ferrante, J., Rose, J. H., & Smith, J. R. (1987). Compressibility of solids. Journal of Geophysical Research: Solid Earth, 92(B9), 9319-9325.

[47] Boboridis, K. (1996). Bestimmung der kritischen Daten von Gold.

[48] Povarnitsyn, M. E., Itina, T. E., Sentis, M., Khishchenko, K. V., & Levashov, P. R. (2007). Material decomposition mechanisms in femtosecond laser interactions with metals. Physical Review B, 75(23), 235414.

[49] Boboridis, K., Pottlacher, G., & Jäger, H. (1999). Determination of the critical point of gold. International journal of thermophysics, 20, 1289-1297.

[50] Young, D. A., & Alder, B. J. (1971). Critical point of metals from the van der Waals model. Physical Review A, 3(1), 364.

[51] Povarnitsyn, M. E., Itina, T. E., Levashov, P. R., & Khishchenko, K. V. (2013). Mechanisms of nanoparticle formation by ultra-short laser ablation of metals in liquid environment. Physical Chemistry Chemical Physics, 15(9), 3108-3114.

Anhang A: Code des Advektions-Diffusions-Modells in Lagrange-Koordinaten

```
# -*- coding: utf-8 -*-

# Material.py

Mat = "Au"  # Al oder Au
```

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import optimize
import matplotlib.animation as animation
from numpy.polynomial.polynomial import Polynomial
from numpy.polynomial.polynomial import polyval
from matplotlib.gridspec import GridSpec
import pathlib
import os
import shutil
from scipy import interpolate
from matplotlib import colors
import time as time_mes
from numba import njit

xsize = 24
ysize = 24
titlesize = 40
xlabelsize = 36
ylabelsize = 36
tick_font_size = 18

# set the colormap and centre the colorbar
class MidpointNormalize(colors.Normalize):
    """Normalise the colorbar."""
    def __init__(self, vmin=None, vmax=None, midpoint=None, clip=False):
        self.midpoint = midpoint
        colors.Normalize.__init__(self, vmin, vmax, clip)

    def __call__(self, value, clip=None):
        x, y = [self.vmin, self.midpoint, self.vmax], [0, 0.5, 1]
        return np.ma.masked_array(np.interp(value, x, y) , np.isnan(value))

# Materialeigenschaften

from Material import Mat

if Mat == "Al":
```

```
    rho0 = 2.6989 # g/cm^3 # Al
if Mat == "Au":
    rho0 = 19.32 # # g/cm^3 # Au

# Interpolanten Zustandsgleichungen einlesen
p_ph_inter=np.load("p_ph_inter" + Mat + ".npy", allow_pickle=True).item()
p_e_inter=np.load("p_e_inter" + Mat + ".npy", allow_pickle=True).item()
#B_inter=np.load("B_inter" + Mat + ".npy", allow_pickle=True).item()


def p_PD(T, rho):
    p = [p_ph_inter(rho[i]/1e3, T[i]/1e3)[0] for i in range(len(T))] # GPa
    return np.array(p)*1e9 # Pa   # 3.5*1e7

def p_PD_e(T, rho):
    p = [p_e_inter(rho[i]/1e3, T[i]/1e3)[0] for i in range(len(T))] # GPa
    return np.array(p)*1e9 # Pa


@njit
def lambda_e_T(T):
    if Mat == "Au":
        K = 353      # W/m/K
        b = 0.16
        u_f = 1.39e6         # m/s
    if Mat == "Al":
        K = 2150    # W/m/K
        b = 1.5
        u_f = 2.03e6         # m/s
    k_B = 1.380649e-23  # J/K
    m_e = 9.109383e-31  # kg
    e_f = 1/2*m_e*u_f**2  # J
    T_i = T_e_0    # K
    theta = k_B*T/e_f
    theta_i = k_B*T_i/e_f

    return K*(theta**2+0.16)**(5/4)*(theta**2+0.44)* \
           theta/((theta**2+0.092)**0.5*(theta**2+b*theta_i))



Ce_data = np.loadtxt("Ce_" + Mat + ".dat") # Datenpunkte einlesen
T_ce = Ce_data[:,0]*1e4
Ce = Ce_data[:,1]*1e5

coeff = np.polyfit(T_ce, Ce, 60)

def cp_e_T(T_test):
    cp_e_fit = np.polyval(coeff, T_test)
    cp_e = np.where(T_test < np.max(T_ce), cp_e_fit, np.max(Ce)+1.65e2)
    return cp_e
```

```
fig, ax_th = plt.subplots()

T_test = np.linspace(0,60000,1000)
ax_th.plot(T_test*1e-3, cp_e_T(T_test)*1e-3, color = "red")
ax_th.plot(T_ce*1e-3, Ce*1e-3, color = "orange", label = "Daten␣Al")
ax_th.plot(1, 1, color = "red", label = "Approximation␣Al")

T_interp = np.linspace(0,10000000,10000000)
cp_e_fit = np.polyval(coeff, T_interp)
cp_e_ = np.where(T_interp < np.max(T_ce), cp_e_fit, np.max(Ce)+1.65e2)
epsilon_e = T_interp*cp_e_
cp_e_epsilon = interpolate.interp1d(epsilon_e, cp_e_)


# Kopplungsfaktor
data = np.loadtxt("G_" + Mat + ".dat")
T_dat = data[:,:-1]*1e4
G_dat = data[:,1:]*1e17
T_dat = T_dat[:,0]
G_dat = G_dat[:,0]

coeff_G = np.polyfit(T_dat,G_dat,27)

def G_values(T_e):
    g = np.where(T_e < np.max(T_dat), polyval(T_e, coeff_G[::-1]), G_dat[-1])
    return g


# Laserpulsparameter
if Mat == "Au":
    kappa = 4.9077
    R = 0
if Mat == "Al":
    kappa = 1.89 # 500 nm
    kappa = 8.3543# 800 nm
    R = 0

wavelenth = 800*1e-9
alpha = 4*kappa*np.pi/wavelenth # Absorptionskoeffizient


@njit
def pulse(t):
    t_pulse = 10e-15
    I_0 = 1e16
    t_0 = 2*t_pulse
    I = I_0*np.exp(-4*np.log(2)*(t-t_0)**2/(t_pulse)**2)
    return I


# Parameter Simulation
small_number = 1e-16
```

```
gamma = 1.4
sigma_max = 0.1
Fo_max = 0.5
Nx=140

if Mat == "Au":
    xmax=2e-7
if Mat == "Al":
    xmax =4e-7
xmin=-xmax
dx=(xmax-xmin)/Nx
x = np.arange(xmin-2*dx,xmax+2*dx,dx)
tmax=200e-12


# Anfangsbedingungen
u = np.zeros_like(x)
rho = np.ones_like(x)*rho0*1e3

# Temperaturen
T_e_0 = 293*np.ones_like(x) # K
T_ph_0 = 293*np.ones_like(x) # K
T_e = 293*np.ones_like(x) # K
T_ph = 293*np.ones_like(x) # K

# Druck
p_1_ph = p_PD(T_ph_0, rho)
p_1_e = p_PD_e(T_e_0, rho)
p_1 = p_1_ph + p_1_e


# Energien
if Mat == "Au":
    cp_ph = 128*np.ones_like(x)*rho0*1e3
    epsilon_e = T_e*cp_e_T(T_e)
    epsilon_ph = T_ph*128*rho0*1e3
if Mat == "Al":
    cp_ph = 897*np.ones_like(x)*rho0*1e3
    epsilon_e = T_e*cp_e_T(T_e)
    epsilon_ph = T_ph*897**rho0*1e3


# Arrays neue Zeitschritte
u_n2 = np.zeros_like(x)
rho_1 = np.zeros_like(x)
T_e_new = np.zeros_like(x)
T_ph_new = np.zeros_like(x)

m = rho[0]*(x[1:] - x[:-1])

# Arrays neue Zeitschritte
rho_new = np.zeros_like(x)
```

```
u_n2 = np.zeros_like(x)
u_old = np.zeros_like(x)
epsilon_e_n2 = np.zeros_like(x)
epsilon_ph_n2 = np.zeros_like(x)
epsilon_e_2 = np.zeros_like(x)
epsilon_ph_2 = np.zeros_like(x)
p_new = np.zeros_like(x)
x_new = x.copy()


def f(rho, rho_new, u_old, u, u_n2, T_e, T_e_new, T_ph, T_ph_new, x):

    cp_e = cp_e_T(T_e)
    lam_e = lambda_e_T(T_e)
    a_e_t=lam_e/cp_e
    a_e_t_max = np.max(a_e_t)
    dt_new_temp_e = Fo_max*np.max(x[1:] - x[:-1])**2/a_e_t_max

    I = u*rho
    if t == 0:
        dt = 0.2e-16
    else:
        dt_new_euler = sigma_max*np.min(x[1:] - x[:-1])/np.max(u)
        dt = (min([dt_new_temp_e, dt_new_euler]))/1.1


    # Lagrange-Punkte
    x_new = x + u*dt
    dx_new = x_new[1:] - x_new[:-1]
    dt_dx=dt/dx_new
    rho_new[:-1] = m/dx_new


    # reflektierende Randbedingungen
    rho_new[0] = rho_new[3]
    rho_new[1] = rho_new[2]
    rho_new[-2] = rho_new[-3]
    rho_new[-1] = rho_new[-4]

    epsilon_e = T_e*cp_e
    epsilon_ph = T_ph*cp_ph

    p_1_ph = p_PD(T_ph, rho)
    p_1_e = p_PD_e(T_e, rho)
    p_1 = p_1_ph + p_1_e

    rho_average = np.zeros_like(x)
    rho_average[1:-1] = 0.5*(rho[:-2] + rho[1:-1])
    rho_average_new = np.zeros_like(x)
    rho_average_new[1:-1] = 0.5*(rho_new[:-2] + rho_new[1:-1])
```

```
u_n2[2:-1] = u[2:-1]*rho_average[2:-1]/rho_average_new[2:-1] \
- dt_dx[2:]* u[2:-1]*rho_average[2:-1]/rho_average_new[2:-1]* \
(u[3:]-u[2:-1]) - dt_dx[2:]*(p_1[2:-1] - p_1[1:-2])/ \
(rho_average_new[2:-1] + small_number) \

# reflektierende Randbedingungen
#u_n2[2] = 0
#u_n2[1] = -u_n2[3]
#u_n2[-2] = 0
#u_n2[-1] = -u_n2[-3]

#offene Randbedingungen
u_n2[0] = u_n2[1]
u_n2[-1] = u_n2[-2]

# Quelle
source = pulse(t)*(1-R)*np.exp(-alpha*x)*alpha

# Quelle
source = pulse(t)*(1-R)*np.exp(-alpha*np.abs(x))*alpha

G = G_values(T_e)


# Zwischenschritt Energie
epsilon_e_1 = np.zeros_like(x)
epsilon_e_1[1:-1] = epsilon_e[1:-1]
-   dt_dx[1:] * epsilon_e[1:-1] * ((u[2:]-u[:-2])/2)
epsilon_e_1[1] = epsilon_e_1[2]
epsilon_e_1[0] = epsilon_e_1[3]
epsilon_e_1[-2] = epsilon_e_1[-3]
epsilon_e_1[-1] = epsilon_e_1[-4]


cp_e = cp_e_epsilon(epsilon_e_1)
T_e = epsilon_e_1/cp_e
lam_e = lambda_e_T(T_e)

T_e_new[1:-1] = (epsilon_e_1[1:-1] - dt_dx[1:] * p_1_e[1:-1] \
*((u[2:]-u[:-2])/2)  )/cp_e[1:-1] \
+ dt/cp_e[1:-1]*((lam_e[2:]+lam_e[1:-1])*(T_e[2:] - T_e[1:-1]) \
- (lam_e[1:-1]+lam_e[:-2])*(T_e[1:-1] - T_e[:-2]))/(2*dx_new[1:]**2) \
+     dt/cp_e[1:-1]*(source[1:-1] - G[1:-1]*(T_e[1:-1] - T_ph[1:-1])) \

# Adiabatische Randbedingungen
T_e_new[0] = T_e_new[1]
T_e_new[-1] = T_e_new[-2]


# Zwischenschritt Energie
epsilon_ph_1 = np.zeros_like(x)
epsilon_ph_1[1:-1] = epsilon_ph[1:-1]
```

```
    - dt_dx[1:] * epsilon_ph[1:-1] * ((u[2:]-u[:-2])/2)
    epsilon_ph_1[1] = epsilon_ph_1[2]
    epsilon_ph_1[0] = epsilon_ph_1[3]
    epsilon_ph_1[-2] = epsilon_ph_1[-3]
    epsilon_ph_1[-1] = epsilon_ph_1[-4]


    T_ph_new[1:-1] = (epsilon_ph_1[1:-1] - dt_dx[1:] * p_1_ph[1:-1] \
    *((u[2:]-u[:-2])/2))  /cp_ph[1:-1] + dt/cp_ph[1:-1] \
    *(G[1:-1]*(T_e[1:-1] - T_ph[1:-1])) \

    # Adiabatische Randbedingungen
    T_ph_new[0] = T_ph_new[1]
    T_ph_new[-1] = T_ph_new[-2]

    return dt, x_new, p_1_e, p_1_ph, p_1



# Liste Zeitschritte
rho_t = [rho.copy()]
u_t = [u.copy()]
epsilon_e_t = [epsilon_e.copy()]
epsilon_ph_t = [epsilon_ph.copy()]
p_t = [p_1.copy()]
p_e_t = [p_1_e.copy()]
p_ph_t = [p_1_ph.copy()]
x_t = [x.copy()]
T_max_e = []
T_min_e = []
T_max_ph = []
T_min_ph = []
T_e_t = [T_e]
T_ph_t = [T_ph]
x_t = [x.copy()]
rho_t = [rho.copy()]
a = 0
t = 0
t_plot = 0
time = [t]
count = 0
count2 = 0

while t < tmax:

    [dt,x_new,p_1_e,p_1_ph,p_1]=f(rho,rho_new,u_old, \
    u,u_n2,T_e,T_e_new,T_ph,T_ph_new,x)

    T_e_help=T_e
    T_e=T_e_new
    T_e_new=T_e_help
```

```
    T_ph_help=T_ph
    T_ph=T_ph_new
    T_e_new=T_e_help

    x_help=x
    x=x_new
    x_new=x_help

    rho_help=rho
    rho=rho_new
    rho_new=rho_help

    u_help=u_old
    u_old=u
    u=u_n2
    u_n2=u_help

    if count2 == 100:
        time.append(t)
        x_t.append(x.copy())
        p_t.append(p_1.copy())
        p_e_t.append(p_1_e.copy())
        p_ph_t.append(p_1_ph.copy())
        rho_t.append(rho.copy()),
        T_ph_t.append(T_ph_new.copy())
        T_e_t.append(T_e_new.copy())
        T_max_e.append(np.max(T_e_new))
        T_max_ph.append(np.max(T_ph_new))
        count2 = 0
    count2 += 1

    t += dt


fig, ax_r = plt.subplots(1,3)

ax_r[0].plot(x_t[-1]*1e9,rho_t[-1]-rho0*1e3,label ="Lagrange",color ="red")
ax_r[1].plot(x_t[-1]*1e9,T_e_t[-1], label = "Elektronen", color = "blue")
ax_r[1].plot(x_t[-1]*1e9,T_ph_t[-1], label = "Phononen", color = "orange")
ax_r[2].plot(x_t[-1]*1e9, p_e_t[-1], label = "Elektronen", color = "blue")
ax_r[2].plot(x_t[-1]*1e9, p_ph_t[-1], label = "Phononen", color = "orange")

ax_r[0].set_ylabel(r"$\bigtriangleup \rho$ in $\dfrac{kg}{m^{3}}$",fontsize=30)
ax_r[0].set_xlabel("x in nm",  fontsize = 30)
ax_r[1].set_ylabel("T in K", fontsize = 30)
ax_r[1].set_xlabel("x in nm",  fontsize = 30)

ax_r[0].tick_params(axis="x", labelsize=24)
ax_r[0].tick_params(axis="y", labelsize=24)

ax_r[1].tick_params(axis="x", labelsize=24)
ax_r[1].tick_params(axis="y", labelsize=24)
```

```
ax_r[0].set_title("Dichtedifferenz␣bei␣10␣ps", fontsize=34)
ax_r[0].set_ylim([np.min(rho_t)-rho0*1e3 -10, np.max(rho_t)-rho0*1e3 +2])
ax_r[1].set_title("Temperatur␣bei␣10␣ps", fontsize=34)
ax_r[1].legend(fontsize=26)
plt.subplots_adjust(left=0.095,
                    bottom=0.09,
                    right=0.99,
                    top=0.95, wspace = 0.25, hspace = 0.33)

xx, tt = np.meshgrid(x, time)
fig, ax = plt.subplots(2,2)

plt.subplots_adjust(left=0.06,
                    bottom=0.095,
                    right=0.995,
                    top=0.92, wspace = 0.14, hspace = 0.5)

# Elektronentemperatur
c = ax[0][0].pcolormesh(xx*1e9, tt*1e12, T_e_t)
cbar = fig.colorbar(c, ax = ax[0][0])

cbar.ax.tick_params(labelsize=tick_font_size)
cbar.set_label(label="$T_\mathrm{el}$" + "␣in␣" + "K",fontsize =26)

ax[0][0].set_title("Elektronentemperatur", fontsize=titlesize , pad =25)
#ax[0][0].set_xlabel("x in nm", fontsize = xlabelsize)
ax[0][0].set_ylabel("$t$␣in␣ps",  fontsize = ylabelsize)
ax[0][0].tick_params(axis="x", labelsize=xsize)
ax[0][0].tick_params(axis="y", labelsize=ysize)


# Phononentemperatur
c = ax[0][1].pcolormesh(xx*1e9, tt*1e12, T_ph_t)
cbar = fig.colorbar(c, ax = ax[0][1])

cbar.ax.tick_params(labelsize=tick_font_size)
cbar.set_label(label="$T_\mathrm{ph}$" + "␣in␣" + "K",fontsize =26)

ax[0][1].set_title("Phononentemperatur", fontsize=titlesize , pad =25)
#ax[0][1].set_xlabel("$x$ in nm", fontsize = xlabelsize)
#ax[1][1].set_ylabel("t in ps",  fontsize = ylabelsize)

ax[0][1].tick_params(axis="x", labelsize=xsize)
ax[0][1].tick_params(axis="y", labelsize=ysize)
#ax[0][1].set_xlabel("x in nm", fontsize = xlabelsize)


# Druck
c = ax[1][0].pcolormesh(xx*1e9, tt*1e12, np.array(p_t)*1e-9)
cbar = fig.colorbar(c, ax = ax[1][0])

cbar.ax.tick_params(labelsize=tick_font_size)
```

```
cbar.set_label(label="$p$" + "␣in␣" + "Gpa",fontsize = 26)

ax[1][0].set_title("Druck", fontsize=titlesize , pad = 25)
ax[1][0].set_xlabel("x␣in␣nm", fontsize = xlabelsize)
ax[1][0].set_ylabel("$t$␣in␣ps",  fontsize = ylabelsize)
ax[1][0].tick_params(axis="x", labelsize=xsize)
ax[1][0].tick_params(axis="y", labelsize=ysize)


# Dichte
c = ax[1][1].pcolormesh(xx*1e9, tt*1e12, np.array(rho_t)-rho0*1e3 )
cbar = fig.colorbar(c, ax = ax[1][1])

cbar.ax.tick_params(labelsize=tick_font_size)
cbar.set_label(label=r"$\bigtriangleup␣\rho$" + "␣in␣"
               + r"$\mathrm{kg}␣/␣\mathrm{m}^3$", fontsize = 26)

ax[1][1].set_title("Dichtedifferenz", fontsize=titlesize , pad = 25)
ax[1][1].set_xlabel("x␣in␣nm", fontsize = xlabelsize)
#ax[1][1].set_ylabel("t in ps",  fontsize = ylabelsize)
ax[1][1].tick_params(axis="x", labelsize=xsize)
ax[1][1].tick_params(axis="y", labelsize=ysize)
```

Anhang B: Code zur Berechnung des Elektronendruckes

```
# -*- coding: utf-8 -*-
"""
Created on Fri Sep 23 14:36:30 2022

@author: linus
"""

import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from scipy.optimize import *
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

from Material import Mat

df = pd.read_table("p_el_"+ Mat +"_55kK.txt", delim_whitespace=True)
p_55 = df["p"].tolist()

df = pd.read_table("p_el_"+ Mat +"_20kK.txt", delim_whitespace=True)
p_20 = df["p"].tolist()

df = pd.read_table("p_el_"+ Mat +"_10kK.txt", delim_whitespace=True)
p_10 = df["p"].tolist()

sig = df["sigma"].tolist()
sig = np.array(sig)
t = np.array([10, 20, 55])


if Mat == "Al":
    v0 = 0.37
    beta0 = 0.046

if Mat == "Au":
    v0 = 0.05175983436853002
    beta0 = 0.046

# Approximation Elektronendruck
def Pe(xy, gamma0e):
    T = xy[1]
    sigma = xy[0]
    return beta0*gamma0e/2/v0 * sigma**(1-gamma0e) * T**2

sigma, T = np.meshgrid(sig, t)
```

```
p = np.array([np.array(p_10), np.array(p_20), np.array(p_55)])

#Approximation
popt,Pcov=curve_fit(Pe,[sigma.ravel(),T.ravel()],p.ravel())
gamma0e = popt[0]

print('gamma0e     ', gamma0e)


# Graphische Darstellung
fig, ax = plt.subplots()
#ax = ax_plot[0]

plt.subplots_adjust(left=0.065,
                    bottom=0.115,
                    right=0.98,
                    top=0.9)
plt.subplots_adjust(hspace = 0.525, wspace = 0.135)

xsize = 24
ysize = 24
titlesize = 40
xlabelsize = 36
ylabelsize = 36

ax.plot(sig, p_10, "r.")
sig_plot = np.linspace(0, 2, 1000)
ax.plot(sig_plot, beta0*gamma0e/2/v0 * sig_plot**(1-gamma0e) * \
        10**2, "r", label = "$T$ = 10 kK")

ax.plot(sig, p_20, "b.")
ax.plot(sig_plot, beta0*gamma0e/2/v0 * sig_plot**(1-gamma0e) * \
        20**2, "b", label = "$T$ = 20 kK")

ax.plot(sig, p_55, "g.")
ax.plot(sig_plot, beta0*gamma0e/2/v0 * sig_plot**(1-gamma0e) * \
        55**2, "g", label = "$T$ = 55 kK")

ax.set_yscale('log')

ax.set_xlim(-0.1,2)
ax.set_ylim(1,200)
ax.grid(visible=True, which='major', linewidth="0.7", axis="y", linestyle="-")
ax.grid(visible=True, which='minor', linewidth="0.3", axis="y", linestyle="-")
ax.grid(visible=True, which='major', linewidth= "0.7", axis="x", linestyle="-")

ax.set_title("Pressure (electrons) " + Mat, fontsize = titlesize, pad = 25)
ax.set_xlabel("$v_0 / v$", fontsize = xlabelsize)
ax.set_ylabel("$p$ in GPa", fontsize = ylabelsize)
ax.legend(fontsize = 20)
ax.tick_params(axis="x", labelsize=xsize)
ax.tick_params(axis="y", labelsize=ysize)
```

```
T_0 =0.293
beta0 = 0.046 #kJ /g/ kK ^2
R = 0.31  # kJ/g/kK
gamma_inf = 2/3


from Material import Mat

if Mat == "Al":
    rho0 = 2.6989 # g/cm^3 # Al
    N = 1000
    t = np.linspace(0,10, N) + 1e-15  # kK
    r = np.linspace(0,rho0*1.3,N) + 1e-15 # g/cm^3
    rho, T= np.meshgrid(r, t)


if Mat == "Au":
    rho0 = 19.32 # Au
    N = 1000
    t = np.linspace(0,10, N) + 1e-15  # kK
    r = np.linspace(0,20,N) + 1e-15 # g/cm^3
    rho, T= np.meshgrid(r, t)


# Elektronendruck
def Pe(T, rho):
    sigma = v0/(1/rho)
    return beta0*gamma0e/2/v0 * sigma**(1-gamma0e) * T**2

# Graphische darstellung
fig, ax_PD = plt.subplots()
#ax_PD = ax_plot_PD[0]

c = ax_PD.pcolormesh(rho, T, Pe(T, rho), vmin = 0, vmax = 3)
cbar = fig.colorbar(c, ax = ax_PD)
tick_font_size = 18
cbar.ax.tick_params(labelsize=tick_font_size)
cbar.set_label(label="$p$ in GPa ",fontsize = 28)
ax_PD.set_xlabel(r'$\rho$ in g/cm^3', fontsize = xlabelsize)
ax_PD.set_ylabel(r'$T$ in kK',  fontsize = ylabelsize)
ax_PD.set_title("Pressure (electrons) " + Mat, fontsize=titlesize, pad=25)
ax_PD.tick_params(axis="x", labelsize=xsize)
ax_PD.tick_params(axis="y", labelsize=ysize)
```

Anhang C: Code zur Berechnung des Druckes aus dem Bindungspotential

```
# -*- coding: utf-8 -*-


import numpy as np
import matplotlib.pylab as plt
from scipy.optimize import minimize
import pandas as pd



def p_rarefraction(T,rho):
    p_r = Ac*(v0_c*rho)**(m+1)+Bc*(v0_c*rho)**(n+1)+Cc*(v0_c*rho)**(l+1)
    return p_r

def sound_speed(param,sigma,v):
    Ac = param[0]
    Bc = param[1]
    Cc = param[2]
    m = param[3]
    n = param[4]
    l = param[5]
    B=Ac*(m+1)*sigma**(m+1) + Bc*(n+1)*sigma**(n+1) + Cc*(l+1)*sigma**(l+1)
    A=B>0
    return (v*B*A)**0.5

def fminr(param,s,c,v):
    cs=sound_speed(param,s,v)
    return np.sum((cs/c-1)**2)

def con1(param):
    Ac = param[0]
    Bc = param[1]
    Cc = param[2]
    m = param[3]
    n = param[4]
    l = param[5]
    return Ac + Bc + Cc

def con2(param):
    Ac = param[0]
    Bc = param[1]
    Cc = param[2]
    m = param[3]
    n = param[4]
    l = param[5]
    return Bc0 - Ac*m -Bc*n - Cc*l
```

```
def con3(param):
    Ac = param[0]
    Bc = param[1]
    Cc = param[2]
    m = param[3]
    n = param[4]
    l = param[5]
    return (BcP0-2)*Bc0 - Ac*m**2 - Bc*n**2 - Cc*l**2



con1 = {'type': 'eq', 'fun': con1}
con2 = {'type': 'eq', 'fun': con2}
con3 = {'type': 'eq', 'fun': con3}
cons = ([con1,con2,con3])

def curve_fit(args):
    solution = minimize(fminr,x0,args=args,method='SLSQP', \
    constraints=cons,tol=1e-15, bounds= bounds)
    x=solution.x
    Ac = x[0]
    Bc = x[1]
    Cc = x[2]
    m = x[3]
    n = x[4]
    l = x[5]
    return Ac, Bc, Cc, m, n, l


colors =["red","black","green","green","green","green","green","green"]
labels =["KEOS5", "KEOS7","Quelle", "Quelle", "Quelle", "Quelle", "Quelle"]


from Material import Mat

fig, ax_B = plt.subplots(1,2)


plt.subplots_adjust(left=0.06, bottom=0.1, right=0.98, top=0.94)
plt.subplots_adjust(hspace = 0.525, wspace = 0.15)
xsize = 24
ysize = 24
titlesize = 40
xlabelsize = 36
ylabelsize = 36


if Mat == "Al":
    rho0c = 2.72
    v0_c = 1/rho0c
    df = pd.read_table('B_cP0_Al.txt', delim_whitespace=True)
```

```
    df1 = pd.read_table('KEOS5_coldcurve_Al.txt', delim_whitespace=True)
    df2 = pd.read_table('KEOS7_coldcurve_Al.txt', delim_whitespace=True)
    x0=[-10,10,-10,10,10,1]
    bounds = ([-60,0], [0,65],[-100,0],[0,1.03],[0,10],[0,10])
    x0=[-57.38,64.18,-6.8,1.03,2.8,5.92]
    px = np.array(df["px"].tolist())
    cx = np.array(df["cx"].tolist())
    v = np.array(df["v"].tolist())
    Bx = cx **2 / v

    grad = np.gradient(Bx,px)
    coeff = np.polyfit(px,Bx,100)
    Bc0 = np.polyval(coeff, 0)
    coeff = np.polyfit(px,grad,100)
    BcP0 = np.polyval(coeff, 0)

    Data = [df1, df2]

if Mat == "Au":
    rho0c = 19.5
    v0_c = 1/rho0c
    df = pd.read_table('B_cP0_Au.txt', delim_whitespace=True)
    df1 = pd.read_table('KEOS5_coldcurve_Au.txt',delim_whitespace=True)

    x0=[-20,10,-10,5,5,2]
    x0=[-10,-10,-10,-10,-1,-1]
    bounds = ([-1000,1000],[-1000,1000],[-1000,1000],[0,5],[-5,5],[-5,5])
    px = np.array(df["px"].tolist())
    cx = np.array(df["cx"].tolist())
    v = np.array(df["v"].tolist())
    Bx = cx **2 / v

    grad = np.gradient(Bx,px)
    coeff = np.polyfit(px,Bx,200)
    Bc0 = np.polyval(coeff, 0)
    coeff = np.polyfit(px,grad,200)
    BcP0 = np.polyval(coeff, 0)

    Data = [df1]



for i in range(len(Data)):

    df = Data[i]
    v = df["v"].tolist()
    v = np.array(v)
    sigma_c = v0_c/v
    c = df["cx"].tolist()
    c = np.array(c)
    s = sigma_c
    #y = cs **2 /v
```

```
args=(s,c,v)

if i > 1:
    x2 = np.append(x2, x[:])
    y2 = np.append(y2, y[:])

if i < 2:
    a = 0
    Ac, Bc, Cc, m, n ,l = curve_fit(args)

    ax_B[0].plot(s, c, ".", color = colors[i], label = labels[i])

    ax_B[1].plot(1, Bc0, "o", color = "red")
    sigma_c_plot = np.linspace(0,10,10000)
    ax_B[0].plot(sigma_c_plot, np.sqrt(v0_c/sigma_c_plot*(sigma_c_plot* \
    (Ac*(m+1)*sigma_c_plot**(m+1) + Bc*(n+1)*sigma_c_plot**(n+1) + Cc* \
    (l+1)*sigma_c_plot**(l+1)))), color=colors[i], label=labels[i] \
    + "␣Approximation")

    ax_B[1].plot(s, c**2/v, ".", color = colors[i], label = labels[i])

    ax_B[1].plot(sigma_c_plot, sigma_c_plot*(Ac*(m+1)*sigma_c_plot**(m+1) \
    + Bc*(n+1)*sigma_c_plot**(n+1) + Cc*(l+1)*sigma_c_plot**(l+1)), \
    color = colors[i], label = labels[i] + "␣Approximation")


elif i < len(Data)-1:
    a = 1

else:
    Ac, Bc, Cc, m, n ,l = curve_fit(x, y)

    ax_B[a].plot(sigma_c, np.sqrt(v0_c/sigma_c*(Ac*(m+1)*sigma_c**(m+1) + \
    Bc*(n+1)*sigma_c**(n+1) + Cc*(l+1)*sigma_c**(l+1))), \
    color = colors[i], label = labels[i])


ax_B[a].set_xlim(0.65,1)
ax_B[a].set_ylim(0,6)
ax_B[a].set_title("Sound␣Speed␣(Al)", fontsize = titlesize)
#ax_B[a].set_xlabel("$v_{0c}/v", fontsize = xlabelsize)
ax_B[a].set_ylabel("$c$␣in␣km/s", fontsize = ylabelsize)
ax_B[a].legend(fontsize = 18)
ax_B[a].tick_params(axis="x", labelsize=xsize)
ax_B[a].tick_params(axis="y", labelsize=ysize)
ax_B[a+1].legend(fontsize = 18)

ax_B[a+1].set_xlim(0.65,1)
ax_B[a+1].set_ylim(0,100)
ax_B[a+1].set_title("Bulk␣Modulus␣(Al)", fontsize = titlesize)
#ax_B[a+1].set_xlabel("$v_{0c}/v$", fontsize = xlabelsize)
#ax_B[a+1].set_ylabel("c", fontsize = ylabelsize)
```

```
    ax_B[a+1].legend(fontsize = 18)
    ax_B[a+1].tick_params(axis="x", labelsize=xsize)
    ax_B[a+1].tick_params(axis="y", labelsize=ysize)
    ax_B[a+1].set_ylabel("$B$ in GPa", fontsize = ylabelsize)
    ax_B[a+1].set_xlabel("$v_{0c}/v$", fontsize = xlabelsize)



fig, ax = plt.subplots(1,2)

plt.subplots_adjust(left=0.065,
                    bottom=0.115,
                    right=0.98,
                    top=0.94)
plt.subplots_adjust(hspace = 0.525, wspace = 0.135)
```

```
from lmfit import *
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import scipy
import matplotlib.pyplot as plt


from Material import Mat

if Mat == "Al":
    v0_c = 0.361 # Aluminium
    rho0c = 2.72
    v0_c = 1/rho0c

    df_B0 = pd.read_table('B_cP0_Al.txt', \
    delim_whitespace=True)
    df1 = pd.read_table('KEOS5_Shock_Hugoniot_Al.txt', \
    delim_whitespace=True)
    df2 = pd.read_table('KEOS7_Shock_Hugoniot_Al.txt', \
    delim_whitespace=True)
    df3 = pd.read_table('Morris_Schock_Hugoniots_Al.txt', \
    delim_whitespace=True)

    df4 = pd.read_table('Isbell_Schock_Hugoniots_Al.txt', \
    delim_whitespace=True)
    df5 = pd.read_table('McQueen_Schock_Hugoniots_Al.txt', \
    delim_whitespace=True)
    df6 = pd.read_table('Chekin_Schock_Hugoniots_Al.txt', \
    delim_whitespace=True)
```

```
    df7 = pd.read_table('Bakanova_Schock_Hugoniots_Al.txt', \
    delim_whitespace=True)
    df8 = pd.read_table('Altshuler_Schock_Hugoniots_Al.txt', \
    delim_whitespace=True)
    Data = [df1, df2, df3, df4, df5, df6, df8]
    colors = ["red","black","grey","red","green","blue","black","orange"]
    labels = ["KEOS5", "KEOS7","1)", "2)", "3)", "4)", "5)", "6)"]

if Mat == "Au":
    v0_c = 0.05175983436853002 # Gold
    rho0c = 19.5
    v0_c = 1/rho0c
    df_B0 = pd.read_table('B_cP0_Au.txt', delim_whitespace=True)
    df1 = pd.read_table('KEOS5_Shock_Hugoniot_Au.txt', \
    delim_whitespace=True)
    df2 = pd.read_table('Walsh_Shock_Hugoniot_Au.txt', \
    delim_whitespace=True)
    df3 = pd.read_table('Altshuler_1958_Schock_Hugoniots_Au.txt',\
    delim_whitespace=True)
    df4 = pd.read_table('McQueen_Schock_Hugoniots_Au.txt', \
    delim_whitespace=True)
    df5 = pd.read_table('Jones_Schock_Hugoniots_Au.txt', \
    delim_whitespace=True)
    df6 = pd.read_table('Marsh_Schock_Hugoniots_Au.txt', \
    delim_whitespace=True)
    df7 = pd.read_table('Altshuler_1981_Schock_Hugoniots_Au.txt',\
    delim_whitespace=True)

    Data = [df1, df2, df3, df4, df5, df6, df7]
    colors = ["red","black","grey","red","green","blue","black","orange"]
    labels = ["KEOS5", "KEOS7","1)", "2)", "3)", "4)", "5)", "6)"]

px = np.array(df_B0["px"].tolist())
cx = np.array(df_B0["cx"].tolist())
v = np.array(df_B0["v"].tolist())
Bx = cx**2 / v


grad = np.gradient(Bx,px)


coeff = np.polyfit(px,Bx,100)
B_c0 = np.polyval(coeff, 0)
coeff = np.polyfit(px,grad,100)
B_cP0 = np.polyval(coeff, 0)

print("B_c0␣␣␣", B_c0)
print("B_cP0␣␣␣", B_cP0)
```

```
def p_compression(sigma_c, params):
    a1 = params[0]
    a2 = params[1]
    a3 = params[2]
    a4 = params[3]
    a5 = params[4]
    return sigma_c *(a1*sigma_c**(1/3) + a2*sigma_c**(2/3) + \
           a3*sigma_c**(3/3) + a4*sigma_c**(4/3) + a5*sigma_c**(5/3))

def least_squares(params,x,y):
    y_pred = p_compression(x,params)
    return np.sum((y_pred - y) ** 2) # least squares



def con1(params):
    a1 = params[0]
    a2 = params[1]
    a3 = params[2]
    a4 = params[3]
    a5 = params[4]
    return a1*(1/3) + a2*(2/3) + a3*(3/3) + a4*(4/3) + a5*(5/3) - B_c0


def con2(params):
    a1 = params[0]
    a2 = params[1]
    a3 = params[2]
    a4 = params[3]
    a5 = params[4]
    return a1 + a2 + a3 + a4 + a5


def con3(params):
    a1 = params[0]
    a2 = params[1]
    a3 = params[2]
    a4 = params[3]
    a5 = params[4]
    return  (a1*(1/3)**2 + a2*(2/3)**2 + a3*(3/3)**2 + \
            a4*(4/3)**2 + a5*(5/3)**2)/B_c0 - B_cP0 + 2


cons = [{'type':'eq', 'fun': con1},
        {'type':'eq', 'fun': con2},
        {'type':'eq', 'fun': con3}]


def curve_fit(sigma_c, p):

    x0 =[469.49,-1338.45,1038.59,-180.78,11.15]
    x0 = [0,0,0,0,0]
```

```
    x = sigma_c
    new = scipy.optimize.minimize(least_squares, x0, \
    args=(sigma_c, p), method='SLSQP', constraints=cons,tol=1e-15)
    popt = new.x
    y_fit = p_compression(x,popt)

    a1 = popt[0]
    a2 = popt[1]
    a3 = popt[2]
    a4 = popt[3]
    a5 = popt[4]

    print("Cold Compression")
    print("a1    ",a1)
    print("a2    ",a2)
    print("a3    ",a3)
    print("a4    ",a4)
    print("a5    ",a5)

    return a1, a2, a3, a4, a5




xsize = 24
ysize = 24
titlesize = 40
xlabelsize = 36
ylabelsize = 36



sigma_c2 = []
p2 = []

for i in range(len(Data)):

    df = Data[i]

    if Mat == "Au":
        i = i+1

    p = np.array(df["p"].tolist())
    v = np.array(df["v"].tolist())
    p = p[(v<v0_c)]
    v = v[(v<v0_c)]
    sigma_c = v0_c/v

    if i > 1:
        sigma_c2 = np.append(sigma_c2, sigma_c[:])
```

```
        p2 = np.append(p2, p[:])
    #print(len(sigma_c2))
    #print(len(p2))
    if i < 2:
        a = 0
        a1, a2, a3, a4, a5 = curve_fit(sigma_c, p)
        ax[a].plot(sigma_c, p, ".", label = labels[i])
        sigma_c_plot = np.linspace(0,10, 1000)
        ax[a].plot(sigma_c_plot, sigma_c_plot *(a1*sigma_c_plot**(1/3)\
        + a2*sigma_c_plot**(2/3) + a3*sigma_c_plot**(3/3) + a4* \
        sigma_c_plot**(4/3) + a5*sigma_c_plot**(5/3)), color = colors[i], \
        label = labels[i] + " Approximation")
        ax[a+1].plot(sigma_c_plot, sigma_c_plot *(a1*sigma_c_plot**(1/3) + \
        a2*sigma_c_plot**(2/3) + a3*sigma_c_plot**(3/3) + a4* \
        sigma_c_plot**(4/3) + a5*sigma_c_plot**(5/3)), \
        color = colors[i], label = labels[i] + " Approximation")

    elif i < len(Data)-1:
        a = 1
        ax[a].plot(sigma_c, p, "o", color = colors[i], label = labels[i])
    else:
        ax[a].plot(sigma_c, p, "o", label = labels[i])
        a1, a2, a3, a4, a5 = curve_fit(sigma_c2, p2)

    if i == len(Data)-1:
        ax[a].plot(sigma_c_plot, sigma_c_plot *(a1*sigma_c_plot**(1/3) \
        + a2*sigma_c_plot**(2/3) + a3*sigma_c_plot**(3/3) + a4* \
        sigma_c_plot**(4/3) + a5*sigma_c_plot**(5/3)),"-.", \
        color = "black", label = "Datenapproximation")

    ax[a].set_yscale('log')
    ax[a].set_xlim(0.95, 3)
    ax[a].set_ylim(1, 2000)



    ax[0].set_title("Pressure " + Mat +" (compression)", \
    fontsize = titlesize , pad = 25)
    ax[0].set_xlabel("$v_{0 c}/v$", fontsize = xlabelsize)
    ax[0].set_ylabel("$p$ in GPa", fontsize = ylabelsize)
    ax[1].set_xlabel("$v_{0 c}/v$", fontsize = xlabelsize)
    ax[1].set_ylabel("$p$ in GPa", fontsize = ylabelsize)

ax[0].legend(fontsize = 15)
ax[1].legend(fontsize = 15)

ax[0].tick_params(axis="x", labelsize=xsize)
ax[0].tick_params(axis="y", labelsize=ysize)

ax[1].tick_params(axis="x", labelsize=xsize)
ax[1].tick_params(axis="y", labelsize=ysize)
```

```
plt.subplots_adjust(left=0.065,
                    bottom=0.1,
                    right=0.98,
                    top=0.915)
plt.subplots_adjust(hspace = 0.3, wspace = 0.16)

ax[1].plot([1.3, 1.3], [0, 10e4], color = "orange")

def Pc(T,rho):
    A_pc=(1/rho)>v0_c
    return p_compression((v0_c*rho), [a1,a2,a3,a4,a5]) * \
           (1-A_pc)+A_pc * p_rarefraction(T,rho)
```

Anhang D: Code zur Berechnung von Grüneisenparameter und Phononendruck

```
# -*- coding: utf-8 -*-

import scipy
import numpy as np
import matplotlib.pyplot as plt
from lmfit import *
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from numpy import log2 as log
from numpy import exp as exp
from numpy import arctan

gamma_inf= 2/3
T_0 =0.293
R = 0.31  # kJ/g/kK


from Material import Mat

df=pd.read_table("Gruneisen_" + Mat + ".txt", delim_whitespace=True)
df2=pd.read_table("Gruneisen_KEOS5_" + Mat + ".txt", delim_whitespace=True)

#if Mat == "Au":
beta0 = 0.046 #kJ /g/ kK ^2
v_0 = 0.37  # cc/g

gamma_0s = 2.19
rho_plot = np.linspace(2.68, 3.5, 1000)
Tm0 =  0.9334 # kK
Tdeb0s = 0.428

T_0 =0.293
beta0 = 0.046 #kJ /g/ kK ^2
R = Rs = 0.31  # kJ/g/kK
R = 0.31  # kJ/g/kK
gamma_inf = 2/3


from Material import Mat

if Mat == "Al":
    rho0 = 2.6989 # g/cm^3 # Al
    rho0c = 2.72
```

```
    v_0 = 1/rho0
    v0_c = 1/rho0c

    N = 1000
    t = np.linspace(0,10, N) + 1e-15  # kK
    r = np.linspace(0,rho0*1.3,N) + 1e-15 # g/cm^3
    rho, T= np.meshgrid(r, t)
    gamma_0s = 2.19

if Mat == "Au":
    rho0 = 19.32 # Au
    v_0 = 1/rho0
    N = 1000
    t = np.linspace(0,10, N) + 1e-15  # kK
    r = np.linspace(0,20,N) + 1e-15 # g/cm^3
    rho, T= np.meshgrid(r, t)
    gamma_0s = 3.08


sigma_plot = np.linspace(0.3,10,1000)

x_redka = sigma_plot
y_redka = gamma_inf + (gamma_0s - gamma_inf) * (0.6**2 + 0.36**2) \
/ (0.6**2 + (log(sigma_plot) + 0.36)**2)


def FuncNew(x,params):
    #gamma_inf = params[0]

    Bs = params[0]
    Ds = params[1]
    return gamma_inf + (gamma_0s - gamma_inf) * (Bs**2 + Ds**2) / \
           (Bs**2 + (log(x) + Ds)**2)


def Objective(params,x,y):
    y_pred = FuncNew(x,params)
    return np.sum((y_pred - y) ** 2) # least squares


x0 = [2/3, 2.1, 0.3, 0.6]
x0 = [0.6,0.36]

bounds = ([0, 10],[0, 10])

sigma = df["sigma"]
gamma = df["gamma"]
v2 = df2["v"]
gamma2 = df2["ga"]
sigma2 = v_0/v2
x = sigma
y = gamma
```

```
gamma_th = gamma_inf+(gamma_0s-gamma_inf)* \
    sigma**(-(gamma_0s/(gamma_0s-gamma_inf)))

y3 = gamma_th

Y =[gamma,gamma2,gamma_th]
X =[sigma,sigma2,sigma]

fig, ax = plt.subplots()

c = ["red", "blue","black"]
l = ["Approximation mit Verlauf aus [26] nach Gl. 95",\
     "Approximation mit Daten aus KEOS5 nach Gl. 95", \
         "theoretischer Verlauf nach Gl. 94"]
params = []
for i in range(3):

    x = X[i]
    y = Y[i]

    new = scipy.optimize.minimize(Objective, x0, args=(x,y), \
          method='SLSQP', tol=1e-12)
    popt3 = new.x
    y_fit3 = FuncNew(x,popt3)


    Bs = popt3[0]
    Ds = popt3[1]

    print(gamma_inf)
    print(gamma_0s)
    print(Bs)
    print(Ds)
    print("-----------------")
    params.append([Bs,Ds])

    if i == 1:
        ax.plot(sigma2, gamma2, "o", label = "Daten aus KEOS5")
    if i == 0:
        ax.plot(sigma, gamma, "r", label="Verlauf nach [26]", color="orange")


    ax.plot(sigma_plot, gamma_inf + (gamma_0s - gamma_inf) * \
    (Bs**2 + Ds**2) / (Bs**2 + (log(sigma_plot) + Ds)**2), \
    color = c[i], label = l[i])


xsize = 24
ysize = 24
titlesize = 40
xlabelsize = 36
```

```
ylabelsize = 36


ax.set_xlim(0.6,2.5)
ax.set_ylim(0.5,4)

sigma_plot = np.linspace(0.75, 10, 1000)

ax.plot(np.linspace(0,10,10), \
gamma_inf*np.ones_like(np.linspace(0,10,10)), \
"r-.", label= "$\gamma_{\inf}$")

ax.plot(1, gamma_0s,"ro", label= "$\gamma_{0s}$")
ax.legend(fontsize = 20)
ax.tick_params(axis="x", labelsize=xsize)
ax.tick_params(axis="y", labelsize=ysize)

ax.set_xlabel(r'$v_0/v$', fontsize = xlabelsize)
ax.set_ylabel(r'$\gamma$',  fontsize = ylabelsize)
ax.set_title("Grüneisenfunktion␣(" + Mat + ")", \
fontsize = titlesize, pad = 25)

plt.subplots_adjust(left=0.065,
                    bottom=0.115,
                    right=0.98,
                    top=0.9)
plt.subplots_adjust(hspace = 0.525, wspace = 0.135)


Bs_Ds_params = params
Bs = Bs_Ds_params[2][0]
Ds = Bs_Ds_params[2][1]
```

Anhang E: Code zur Berechnung der Schmelzkurve

```
# -*- coding: utf-8 -*-
"""
Created on Thu Nov 17 21:18:02 2022

@author: linus
"""

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy
from scipy.optimize import curve_fit
from scipy.optimize import fsolve

from Material import Mat

if Mat == "Al":
    df = pd.read_table('Melting_curve_Al_2.txt', delim_whitespace=True)
    df2 = pd.read_table('Melting_curve_Al.txt', delim_whitespace=True)
    T_m0 = 0.93345

if Mat == "Au":
    df = pd.read_table('Melting_curve_Au_3.txt', delim_whitespace=True)
    df2 = pd.read_table('Melting_curve_Au_2.txt', delim_whitespace=True)
    T_m0 =  1.33733


T = df["T"].tolist()
T = (np.array(T))*1e-3

p = df["p"].tolist()
p = np.array(p)

T2 = df2["T"].tolist()
T2 = (np.array(T2))*1e-3

p2 = df2["p"].tolist()
p2 = np.array(p2)

# Simon-Glatzel-Gleichung
def P_melt(T,A_param,B_param):
    return A_param*((T/T_m0)**(1/B_param) - 1)


# Approximation
popt,Pcov=curve_fit(P_melt,T,p)
```

```
A_param = popt[0]
B_param = popt[1]

print(A_param)
print(B_param)

t = np.linspace(0,10, 1000) + 1e-15  # kK
r = np.linspace(0,3.5,1000) + 1e-15 # g/cm^3


# Graphische Darstellung
fig, ax = plt.subplots()


xsize = 24
ysize = 24
titlesize = 40
xlabelsize = 36
ylabelsize = 36
ax.set_title("Schmelzdruck␣(" + Mat +")", fontsize=titlesize, pad=25)
ax.plot(T,p,"o", label = "Daten␣aus␣[41]")
ax.set_xlim(0,5)
ax.set_ylim(-10,150)

ax.text(T_m0-0.4, 8, r'$T_{m0}$', color = "blue", fontsize=32)
ax.tick_params(axis="x", labelsize=xsize)
ax.tick_params(axis="y", labelsize=ysize)
ax.set_ylabel(r'$p$␣in␣GPa',  fontsize = ylabelsize)
ax.plot(T2,p2,"o", label = "Daten␣aus␣[42]", color = "green")


ax.tick_params(axis="x", labelsize=xsize)
ax.tick_params(axis="y", labelsize=ysize)
ax.set_xlabel(r'$T$␣in␣kK', fontsize = xlabelsize)
ax.set_xlabel(r'$T$␣in␣kK', fontsize = xlabelsize)
ax.set_ylabel(r'$p$␣in␣GPa',  fontsize = ylabelsize)

ax.plot(T_m0,0, "ob", color = "blue")


plt.subplots_adjust(left=0.07,
                    bottom=0.15,
                    right=0.975,
                    top=0.9, wspace = 0.25,  hspace = 0.2)




# Modell aus Hieu, H. K., & Ha, N. N. (2013).
```

```
Pm_values = p
Tm_values = T

if Mat == "Al":
    T0 = 0.9335
    gamma_0 = 2.19
if Mat == "Au":
    Tm0 =  1.33733 # kK
    T0 =  1.33733 # kK
    gamma_0 = 3.08


# Mumerische Lösung

def f(Tm_values,q, k, k_):
    Pm = []
    for Tm in Tm_values:

        def func(z):
            sigma = z[0]
            Pm = z[1]
            F = np.empty((2))
            F[0] = T0*(sigma)**(2/3)*np.exp((2*gamma_0/q)*(1 - sigma**q))-Tm
            F[1] = 3*k*(sigma)**(-2/3) * (1 - sigma**(1/3)) * \
            np.exp(3/2 * (k_ - 1) * (1-sigma**(1/3))) -  Pm
            return F

        zGuess = np.array([0.25,0.2])
        z = fsolve(func,zGuess)
        Pm.append(z[1])

    return np.array(Pm)

x0 = (1.2, 170, 6)
popt,Pcov=curve_fit(f,Tm_values,Pm_values, x0)

q = popt[0]
k = popt[1]
k_ = popt[2]

print("q  :", q)
print("k  :", k)
print("k_  :", k_)



# Graphische Darstellung
Tm = np.linspace(0.9335,15,10000)
yapp=f(Tm,q, k, k_)
ax.plot(Tm,yapp,label='Approximation mit Daten aus [41] nach Gl. 103, 105' \
        , color = "red")
```

```
ax.legend(fontsize = 18)


ax.plot(t,  P_melt(t,A_param,B_param), \
label = "Approximation␣mit␣Daten␣aus␣[41]␣nach␣Gl.␣100", color = "orange")


ax.legend(fontsize = 18)
```

Anhang F: Code zur Berechnung des Flüssigdruckes

```
# -*- coding: utf-8 -*-

import numpy as np
from numpy import log, exp, arctan, sqrt
from numpy import arctan as atan
from scipy.signal import find_peaks
from scipy.interpolate import PchipInterpolator as pchip
from scipy import integrate
import matplotlib.pyplot as plt
from operator import xor
from matplotlib import colors
from numba import njit
from matplotlib import  colors, cm
from scipy.optimize import least_squares
import scipy
from scipy.optimize import curve_fit


xsize = 24
ysize = 24
titlesize = 40
xlabelsize = 36
ylabelsize = 36
tick_font_size = 18


class MidpointNormalize(colors.Normalize):
    def __init__(self, vmin=None, vmax=None, midpoint=None, clip=False):
        self.midpoint = midpoint
        colors.Normalize.__init__(self, vmin, vmax, clip)

    def __call__(self, value, clip=None):
        x, y = [self.vmin, self.midpoint, self.vmax], [0, 0.5, 1]
        return np.ma.masked_array(np.interp(value, x, y),np.isnan(value))


T_0 =0.293
beta0 = 0.046 #kJ /g/ kK ^2
R = 0.31  # kJ/g/kK
gamma_inf = 2/3
T_ca = 25
sigma_0m = 0.873



from Material import Mat
```

```
if Mat == "Al":

    rho0 = 2.6989 # g/cm^3 # Al
    rho0c = 2.72

    v_0 = 1/rho0

    v0_c = 1/rho0c

    Tm0 =  0.933 # kK


    v_c = 1.23
    T_c = 7.158
    rho_c = 0.80645
    p_c =1.41
    N = 1000
    t = np.linspace(0,10, N) + 1e-15  # kK
    r = np.linspace(0,rho0*1.3,N) + 1e-15 # g/cm^3
    rho, T= np.meshgrid(r, t)
    gamma_0s = 2.19

if Mat == "Au":

    rho0 = 19.32 # Au
    v_0 = 1/rho0
    v0 = v_0
    v0_c = v0


    Tm0 =  1.33733 # kK
    T_c = 7.800
    rho_c = 5.9
    v_c = 1/rho_c
    p_c =0.525 #GPa
    N = 2000
    t = np.linspace(0,10, N) + 1e-15  # kK
    r = np.linspace(0,20,N) + 1e-15 # g/cm^3
    rho, T= np.meshgrid(r, t)
    gamma_0s = 3.08


# Funktionen

from BindingPotential import Pc, v0_c, a1,a2,a3,a4,a5, l, m, n, Ac, Bc, Cc

from Fit_el import Pe, gamma0e

from Grüneisen import Bs_Ds_params
```

```
Bs = Bs_Ds_params[2][0]
Ds = Bs_Ds_params[2][1]


def Pas(T, rho):
    v = 1/rho
    D = Ds
    B = Bs
    gamma_0 = gamma_0s
    T_deb0 = 1 # Wert, da nicht in pas gekuerzt
    pas = -(3*R*T**2*exp(-((D**2+B**2)*(gamma_0-gamma_inf)*(arctan((log(v_0* \
    rho)+D)/B)-arctan(D/B)))/B)*(-((D**2+B**2)*T_deb0*(gamma_0-gamma_inf)* \
    (v_0*rho)**gamma_inf*exp(((D**2+B**2)*(gamma_0-gamma_inf)* \
    (arctan((log(v_0*rho)+D)/B)-arctan(D/B)))/B))/(B**2*T*((log(v_0*rho)+ \
    D)**2/B**2+1)*(1/rho))-(T_deb0*gamma_inf*(v_0*rho)**gamma_inf* \
    exp(((D**2+B**2)*(gamma_0-gamma_inf)*(arctan((log(v_0*rho)+D)/B)- \
    arctan(D/B)))/B))/(T/rho)))/(log(10)*T_deb0*(v_0*rho)**gamma_inf)
    return pas


def Ps(T,rho):
    ps = Pc(T,rho)+Pas(T,rho)+Pe(T,rho)
    if Mat == "Au":
        ps = Pc(T,rho)+0.4*Pas(T,rho)+Pe(T,rho)
    return ps



# Schmelzkurve
def melting_curve(ps):
    A_melt = ps  <= p_melt   # Bereich über der Schmelzkurve
    ps[A_melt] = 0
    tm = np.argmin(np.abs(ps), axis = 0)
    t_m = t[tm]
    r_m = r
    t_m = t_m[len(r)//2 :]
    r_m= r[len(r)//2 :]
    r_m= r_m[np.where(t_m >= Tm0)]
    t_m = t_m[np.where(t_m >= Tm0)]
    return r_m, t_m



from Melting_curve import A_param, B_param

def P_melt(T):

    return A_param*((T/Tm0)**(1/B_param) - 1)


p_melt = P_melt(T)
p_melt[(Ps(T, rho) > p_melt)] = 0
```

```
ps=Ps(T, rho)
p_melt = P_melt(T)

r_m = melting_curve(ps)[0]
t_m = melting_curve(ps)[1]


p0 = 1e-4
A=ps<p0

p = ps

r_m = melting_curve( Ps(T,rho))[0]
t_m = melting_curve( Ps(T,rho))[1]

x_2 = r_m
y_2 = t_m
P_2 = 0.4*Pas(T=y_2,rho=x_2)


def van_der_Waals(T, rho):
    v = 1/rho
    Vr = v/v_c
    Tr = T/T_c
    return (8*Tr/(3*Vr - 1)  -  3/Vr**2)*p_c

x_1 = np.ones(10)*rho_c
y_1 = np.linspace(T_c, 10, 10)
P_1 = van_der_Waals(T=y_1, rho=x_1)-Pc(T=y_1,rho=x_1)-Pe(T=y_1,rho=x_1)

x = np.append(x_1,x_2)
y = np.append(y_1,y_2)
P = np.append(P_1,P_2)

def Pt(xy, B_l, D_l, T_sa, T_T, sigma_T):
    rho = xy[0]
    T = xy[1]
    return -3*R*T**2*(v0_c*rho)**(-0.666666666666667)*(T + T_ca)* \
           (T_T*v0_c/((T + T_T)*(sigma_T + v0_c*rho)/rho) + 1)*\
           (-0.666666666666667*T_sa*(v0_c*rho)**0.666666666666667*(T + \
           T_deb0l*exp((B_l**2 + D_l**2)*(gamma_0l - 0.666666666666667)* \
           atan(B_l*log(v0_c*rho)/(B_l**2 + D_l*(D_l + log(v0_c*rho)))) \
           /B_l))/(T/rho*(T + T_ca)) + T_deb0l*T_sa*(v0_c*rho)** \
           0.666666666666667*(B_l**2 + D_l**2)*(gamma_0l -0.666666666667) \
           *(B_l*D_l*log(v0_c*rho)/((B_l**2 + D_l*(D_l+log(v0_c*rho))/rho) \
           **2) - B_l/((B_l**2 + D_l*(D_l + log(v0_c*rho)))/rho))* \
           exp((B_l**2 + D_l**2)*(gamma_0l - 0.666666666666667)*atan(B_l* \
           log(v0_c*rho)/(B_l**2 + D_l*(D_l + log(v0_c*rho))))/B_l)/(B_l*T \
           *(T + T_ca)*(B_l**2*log(v0_c*rho)**2/(B_l**2 + D_l*(D_l + \
           log(v0_c*rho)))**2 + 1)))/(2*T_sa*(T + T_deb0l*exp((B_l**2 + \
```

```
            D_l**2)*(gamma_0l - 0.666666666666667)*atan(B_l*log(v0_c*rho) \
            /(B_l**2 + D_l*(D_l + log(v0_c*rho))))/B_l))) - 3*R*T*(-T_T* \
            v0_c/((1/rho)**2*(T + T_T)*(sigma_T + v0_c*rho)) + T_T*v0_c \
            **2/((1/rho)**3*(T + T_T)*(sigma_T + v0_c*rho)**2))*log(T_sa* \
            (v0_c*rho)**0.666666666666667*(T + T_deb0l*exp((B_l**2 + \
            D_l**2)*(gamma_0l - 0.666666666666667)*atan(B_l*log(v0_c* \
            rho)/(B_l**2 + D_l*(D_l + log(v0_c*rho))))/B_l))/(T*(T+T_ca)))/2

T_deb0l = 174
gamma_0l = 1.55
T_ca = 25

if Mat == "Al":
    bounds=([0,-0.1],[5,0.1])
    bounds=([1.5,-0.001, 4, 30, 0.09], \
             [1.6,0.001, 4+1e-10, 30+1e-10, 0.09+1e-10])
    popt3,Pcov = scipy.optimize.curve_fit(Pt, [x,y], P, bounds=bounds)



elif Mat == "Au":
    bounds=([0,-0.1,10,10,0],[5,0.1,50,50,1])
    bounds=([0,0,0,0,0],[10,1e-6,10,1,50])
    popt3,Pcov = scipy.optimize.curve_fit(Pt, [x,y], P, bounds=bounds)
    T_T = 30
    sigma_T = 0.09
    T_sa = 4

B_l = popt3[0]
D_l = popt3[1]
T_sa= popt3[2]
T_T= popt3[3]
sigma_T= popt3[4]


print("B_l____", B_l)
print("D_l____", D_l)
print("T_sa____", T_sa)
print("T_T____", T_T)
print("sigma_T____", sigma_T)




# Korrekturterm Schmelzzone


def Pm(x, params):
    rho = x
    A_m= params[0]
```

```
    C_m = params[1]
    return -3*R*(-1.3333333333333*A_m*Tm0*v_0**2*(v_0/sigma_0m*rho)** \
           1.66666666666667/(sigma_0m**2/rho**3*(1 + 3*v_0/sigma_0m*rho)) \
           - 4*Tm0*v_0**2*(0.4*A_m*((v_0/sigma_0m*rho)**1.66666666666667 \
           - 1) + C_m)/(sigma_0m**2/rho**3*(1 + 3*v_0/sigma_0m*rho)) + \
           6*Tm0*v_0**3*(0.4*A_m*((v_0/sigma_0m*rho)**1.66666666666667 \
           - 1) + C_m)/(sigma_0m**3/rho**4*(1 + 3*v_0/sigma_0m*rho)**2))


if Mat == "Al":
    rho2 = 2.35
elif Mat == "Au":
    rho2 = 16.8

t_m = melting_curve(ps)[1]

# Thermische Expansionskurve

# Thermische Expansionskurve
def T_therm_exp(rho):
    gamma_0 = gamma_0s
    v0 = v_0
    T_deb0 = 1  # Wert, da in t_exp nicht gekuerzt
    t_exp = -1.30288344570976*R*(rho*v0)**gamma0e*(Bs**2*gamma_0 + Ds**2* \
    gamma_0 + 2.0*Ds*gamma_inf*log(rho) - 1.98502151557112*Ds*gamma_inf + \
    gamma_inf*log(rho)**2 - 1.98502151557112*gamma_inf*log(rho) + \
    0.98507760431994*gamma_inf)/(beta0*gamma0e*(Bs**2 + Ds**2 + 2.0*Ds*log(rho) \
    - 1.98502151557112*Ds + log(rho)**2 - 1.98502151557112*log(rho) + \
    0.98507760431994)) + 0.434294481903252*sqrt((rho*v0)**gamma0e* \
    (-10.6037962209568*Ac*Bs**4*beta0*gamma0e*v0_c*(rho*v0_c)**m - \
    21.2075924419136*Ac*Bs**2*Ds**2*beta0*gamma0e*v0_c*(rho*v0_c)**m - \
    42.4151848838272*Ac*Bs**2*Ds*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho) + \
    42.0975272906619*Ac*Bs**2*Ds*beta0*gamma0e*v0_c*(rho*v0_c)**m - \
    21.2075924419136*Ac*Bs**2*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho)**2 + \
    42.0975272906619*Ac*Bs**2*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho) - \
    20.8911243560724*Ac*Bs**2*beta0*gamma0e*v0_c*(rho*v0_c)**m - \
    10.6037962209568*Ac*Ds**4*beta0*gamma0e*v0_c*(rho*v0_c)**m - \
    42.4151848838272*Ac*Ds**3*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho) + \
    42.0975272906619*Ac*Ds**3*beta0*gamma0e*v0_c*(rho*v0_c)**m - \
    63.6227773257408*Ac*Ds**2*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho)**2 + \
    126.292581871986*Ac*Ds**2*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho) - \
    62.6733730682463*Ac*Ds**2*beta0*gamma0e*v0_c*(rho*v0_c)**m - \
    42.4151848838272*Ac*Ds*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho)**3 + \
    126.292581871986*Ac*Ds*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho)**2 - \
    125.346746136493*Ac*Ds*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho) + \
    41.4693313306198*Ac*Ds*beta0*gamma0e*v0_c*(rho*v0_c)**m - \
    10.6037962209568*Ac*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho)**4 + \
    42.0975272906619*Ac*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho)**3 - \
    62.6733730682463*Ac*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho)**2 + \
    41.4693313306198*Ac*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho) - \
    10.2896893769503*Ac*beta0*gamma0e*v0_c*(rho*v0_c)**m - 10.6037962209568* \
    Bs**4*Bc*beta0*gamma0e*v0_c*(rho*v0_c)**n - 10.6037962209568*Bs**4*Cc*beta0* \
```

```
gamma0e*v0_c*(rho*v0_c)**l + 9.0*Bs**4*R**2*gamma_0**2*(rho*v0)**gamma0e \
- 21.2075924419136*Bs**2*Bc*Ds**2*beta0*gamma0e*v0_c*(rho*v0_c)**n - \
42.4151848838272*Bs**2*Bc*Ds*beta0*gamma0e*v0_c*(rho*v0_c)**n*log(rho) + \
42.0975272906619*Bs**2*Bc*Ds*beta0*gamma0e*v0_c*(rho*v0_c)**n - \
21.2075924419136*Bs**2*Bc*beta0*gamma0e*v0_c*(rho*v0_c)**n*log(rho)**2 + \
42.0975272906619*Bs**2*Bc*beta0*gamma0e*v0_c*(rho*v0_c)**n*log(rho) - \
20.8911243560724*Bs**2*Bc*beta0*gamma0e*v0_c*(rho*v0_c)**n - \
21.2075924419136*Bs**2*Cc*Ds**2*beta0*gamma0e*v0_c*(rho*v0_c)**l - \
42.4151848838272*Bs**2*Cc*Ds*beta0*gamma0e*v0_c*(rho*v0_c)**l*log(rho) \
+ 42.0975272906619*Bs**2*Cc*Ds*beta0*gamma0e*v0_c*(rho*v0_c)**l - \
21.2075924419136*Bs**2*Cc*beta0*gamma0e*v0_c*(rho*v0_c)**l*log(rho)**2 + \
42.0975272906619*Bs**2*Cc*beta0*gamma0e*v0_c*(rho*v0_c)**l*log(rho) - \
20.8911243560724*Bs**2*Cc*beta0*gamma0e*v0_c*(rho*v0_c)**l + 18.0*Bs**2* \
Ds**2*R**2*gamma_0**2*(rho*v0)**gamma0e + 36.0*Bs**2*Ds*R**2*gamma_0* \
gamma_inf*(rho*v0)**gamma0e*log(rho) - 35.7303872802802*Bs**2*Ds*R**2* \
gamma_0*gamma_inf*(rho*v0)**gamma0e + 18.0*Bs**2*R**2*gamma_0*gamma_inf \
*(rho*v0)**gamma0e*log(rho)**2 - 35.7303872802802*Bs**2*R**2*gamma_0* \
gamma_inf*(rho*v0)**gamma0e*log(rho) + 17.7313968777598*Bs**2*R**2* \
gamma_0*gamma_inf*(rho*v0)**gamma0e - 10.6037962209568*Bc*Ds**4*beta0* \
gamma0e*v0_c*(rho*v0_c)**n - 42.4151848838272*Bc*Ds**3*beta0*gamma0e* \
v0_c*(rho*v0_c)**n*log(rho) + 42.0975272906619*Bc*Ds**3*beta0* \
gamma0e*v0_c*(rho*v0_c)**n - 63.6227773257408*Bc*Ds**2*beta0*gamma0e* \
v0_c*(rho*v0_c)**n*log(rho)**2 + 126.292581871986*Bc*Ds**2*beta0* \
gamma0e*v0_c*(rho*v0_c)**n*log(rho) - 62.6733730682463*Bc*Ds**2*beta0 \
*gamma0e*v0_c*(rho*v0_c)**n - 42.4151848838272*Bc*Ds*beta0*gamma0e* \
v0_c*(rho*v0_c)**n*log(rho)**3 + 126.292581871986*Bc*Ds*beta0*gamma0e*v0_c \
*(rho*v0_c)**n*log(rho)**2 - 125.346746136493*Bc*Ds*beta0*gamma0e*v0_c \
*(rho*v0_c)**n*log(rho) + 41.4693313306198*Bc*Ds*beta0*gamma0e*v0_c* \
(rho*v0_c)**n - 10.6037962209568*Bc*beta0*gamma0e*v0_c*(rho*v0_c)**n \
*log(rho)**4 + 42.0975272906619*Bc*beta0*gamma0e*v0_c*(rho*v0_c)**n* \
log(rho)**3 - 62.6733730682463*Bc*beta0*gamma0e*v0_c*(rho*v0_c)**n \
*log(rho)**2 + 41.4693313306198*Bc*beta0*gamma0e*v0_c*(rho*v0_c)**n \
*log(rho) - 10.2896893769503*Bc*beta0*gamma0e*v0_c*(rho*v0_c)**n - \
10.6037962209568*Cc*Ds**4*beta0*gamma0e*v0_c*(rho*v0_c)**l - \
42.4151848838272*Cc*Ds**3*beta0*gamma0e*v0_c*(rho*v0_c)**l*log(rho) + \
42.0975272906619*Cc*Ds**3*beta0*gamma0e*v0_c*(rho*v0_c)**l-63.6227773257408 \
*Cc*Ds**2*beta0*gamma0e*v0_c*(rho*v0_c)**l*log(rho)**2 + 126.292581871986* \
Cc*Ds**2*beta0*gamma0e*v0_c*(rho*v0_c)**l*log(rho) - 62.6733730682463*Cc* \
Ds**2*beta0*gamma0e*v0_c*(rho*v0_c)**l - 42.4151848838272*Cc*Ds*beta0*gamma0e \
*v0_c*(rho*v0_c)**l*log(rho)**3 + 126.292581871986*Cc*Ds*beta0*gamma0e*v0_c* \
(rho*v0_c)**l*log(rho)**2 - 125.346746136493*Cc*Ds*beta0*gamma0e*v0_c* \
(rho*v0_c)**l*log(rho) + 41.4693313306198*Cc*Ds*beta0*gamma0e*v0_c* \
(rho*v0_c)**l - 10.6037962209568*Cc*beta0*gamma0e*v0_c*(rho*v0_c)**l* \
log(rho)**4 + 42.0975272906619*Cc*beta0*gamma0e*v0_c*(rho*v0_c)**l* \
log(rho)**3 - 62.6733730682463*Cc*beta0*gamma0e*v0_c*(rho*v0_c)**l* \
log(rho)**2 + 41.4693313306198*Cc*beta0*gamma0e*v0_c*(rho*v0_c)**l* \
log(rho) - 10.2896893769503*Cc*beta0*gamma0e*v0_c*(rho*v0_c)**l + \
9.0*Ds**4*R**2*gamma_0**2*(rho*v0)**gamma0e + 36.0*Ds**3*R**2*gamma_0 \
*gamma_inf*(rho*v0)**gamma0e*log(rho) - 35.7303872802802*Ds**3*R**2* \
gamma_0*gamma_inf*(rho*v0)**gamma0e + 18.0*Ds**2*R**2*gamma_0*gamma_inf* \
(rho*v0)**gamma0e*log(rho)**2-35.7303872802802*Ds**2*R**2*gamma_0*gamma_inf* \
(rho*v0)**gamma0e*log(rho) + 17.7313968777598*Ds**2*R**2*gamma_0*gamma_inf* \
```

```
    (rho*v0)**gamma0e+36.0*Ds**2*R**2*gamma_inf**2*(rho*v0)**gamma0e*log(rho)**2 \
    - 71.4607745605604*Ds**2*R**2*gamma_inf**2*(rho*v0)**gamma0e*log(rho) \
    + 35.4627937555197*Ds**2*R**2*gamma_inf**2*(rho*v0)**gamma0e + 36.0*Ds* \
    R**2*gamma_inf**2*(rho*v0)**gamma0e*log(rho)**3 - 107.19116184084*Ds*R**2* \
    gamma_inf**2*(rho*v0)**gamma0e*log(rho)**2 + 106.388381266559*Ds*R**2* \
    gamma_inf**2*(rho*v0)**gamma0e*log(rho) - 35.1972043029964*Ds*R**2* \
    gamma_inf**2*(rho*v0)**gamma0e + 9.0*R**2*gamma_inf**2*(rho*v0)**gamma0e* \
    log(rho)**4 - 35.7303872802802*R**2*gamma_inf**2*(rho*v0)**gamma0e*log(rho)**3 \
    + 53.1941906332795*R**2*gamma_inf**2*(rho*v0)**gamma0e*log(rho)**2 - \
    35.1972043029964*R**2*gamma_inf**2*(rho*v0)**gamma0e*log(rho) + \
    8.73340097814798*R**2*gamma_inf**2*(rho*v0)**gamma0e))/(beta0*gamma0e* \
    (Bs**2 + Ds**2 + 2.0*Ds*log(rho) - 1.98502151557112*Ds + log(rho)**2 - \
    1.98502151557112*log(rho) + 0.98507760431994))
    t_exp = t_exp[len(r)//2 :]
    r_exp = r[len(r)//2 :]
    r_exp = r_exp[np.where(t_exp <= Tm0)]
    t_exp = t_exp[np.where(t_exp <= Tm0)]
    r_exp = r_exp[np.where(t_exp >= 0)]
    t_exp = t_exp[np.where(t_exp >= 0)]
    return r_exp, t_exp

rho1 = r[np.argmin(np.abs(rho - T_therm_exp(r)[0][0] ))]


def equations(z):
    A_m, C_m = z
    return (Pm(rho2, (A_m, C_m)) + (Pc(Tm0,rho2) + \
            Pt((Tm0,rho2), B_l, D_l, T_sa, T_T, sigma_T) + \
            Pe(Tm0,rho2) -  P_melt(Tm0)),
            Pm(rho_c, (A_m, C_m)))


if Mat == "Al":
    x0 = (5, -3)
    bounds = ((2, -3), (20, -2) )

if Mat == "Au":
    x0 = (0.75, 0.3)
    bounds = ((0.65, 0.2), (0.8, 0.3) )

    x0 = (0.8, 0.2)
    bounds = ((0.65, 0.1), (0.95, 0.2) )

    x0 = (0.9, -0.3)
    bounds = ((0.8, -0.35), (1.5, -0.25) )


res = least_squares(equations, x0 = x0, bounds = bounds)

A_m = res.x[0]
C_m = res.x[1]
```

```
print("A_m␣␣␣␣␣", A_m)
print("C_m␣␣␣␣␣", C_m)


if Mat == "Au":
    rho_plot = np.linspace(rho_c, 20, 1000)
if Mat == "Al":
    rho_plot = np.linspace(rho_c, 3.5, 1000)

fig, ax = plt.subplots()
ax.plot(rho2, P_melt(Tm0) - (Pc(Tm0,rho2) + Pt([rho2,Tm0], B_l, D_l, \
T_sa, T_T, sigma_T) + Pe(Tm0,rho2)) , "o", color = "red" )
ax.plot(rho_plot, Pm(rho_plot, [A_m, C_m]), color = "red")

ax.plot(rho_c, 0, "o",color = "orange")
ax.plot(rho_plot, Pm(rho_plot, (A_m, C_m)), color = "orange")


def P_m_v(v):
    pm =  Pm(x = 1/v, params = [A_m, C_m])
    return pm

print("Schmelzenthalpie␣␣␣␣", integrate.quad(P_m_v, 1/rho1, 1/rho2)[0])
```

Anhang G: Code zur Zusammensetzung der Zustandsgleichung

```
# -*- coding: utf-8 -*-

import numpy as np
from numpy import log, exp, arctan, sqrt
from numpy import arctan as atan
from scipy.signal import find_peaks
from scipy.interpolate import PchipInterpolator as pchip
from scipy.integrate import simps
import matplotlib.pyplot as plt
from operator import xor
from matplotlib import colors
from numba import njit
from matplotlib import  colors, cm
from scipy.optimize import least_squares

xsize = 24
ysize = 24
titlesize = 40
xlabelsize = 36
ylabelsize = 36
tick_font_size = 18

class MidpointNormalize(colors.Normalize):
    def __init__(self, vmin=None, vmax=None, midpoint=None, clip=False):
        self.midpoint = midpoint
        colors.Normalize.__init__(self, vmin, vmax, clip)
    def __call__(self, value, clip=None):
        x, y = [self.vmin, self.midpoint, self.vmax], [0, 0.5, 1]
        return np.ma.masked_array(np.interp(value, x, y),np.isnan(value))


T_0 =0.293
beta0 = 0.046 #kJ /g/ kK ^2
R = 0.31  # kJ/g/kK
gamma_inf = 2/3


from Material import Mat

if Mat == "Al":
    rho0 = 2.6989 # g/cm^3 # Al
    rho0c = 2.72
    v_0 = 1/rho0
    v0_c = 1/rho0c
    Tm0 =  0.933 # kK
    v_c = 1.23
```

```
    T_c = 7.158
    rho_c = 0.80645
    p_c =1.41
    N = 1000
    t = np.linspace(0,10, N) + 1e-15  # kK
    r = np.linspace(0,rho0*1.3,N) + 1e-15 # g/cm^3
    rho, T= np.meshgrid(r, t)
    gamma_0s = 2.19

if Mat == "Au":
    rho0 = 19.32 # Au
    v_0 = 1/rho0
    v0 = v_0
    v0_c = v0
    Tm0 =  1.33733 # kK
    T_c = 7.800
    rho_c = 5.9
    v_c = 1/rho_c
    p_c =0.525 #GPa
    N = 2000
    t = np.linspace(0,10, N) + 1e-15  # kK
    r = np.linspace(0,20,N) + 1e-15 # g/cm^3
    rho, T= np.meshgrid(r, t)
    gamma_0s = 3.08



from BindingPotential import Pc, v0_c, a1,a2,a3,a4,a5, l, m, n, Ac, Bc, Cc

from Fit_el import Pe, gamma0e

from Grüneisen import Bs_Ds_params


Bs = Bs_Ds_params[2][0]
Ds = Bs_Ds_params[2][1]


def Pas(T, rho):
    v = 1/rho
    D = Ds
    B = Bs
    gamma_0 = gamma_0s
    T_deb0 = 1 # Wert, da nicht in pas gekuerzt
    pas = -(3*R*T**2*exp(-((D**2+B**2)*(gamma_0-gamma_inf)*(arctan((log(v_0* \
    rho)+D)/B)-arctan(D/B)))/B)*(-((D**2+B**2)*T_deb0*(gamma_0-gamma_inf)* \
    (v_0*rho)**gamma_inf*exp(((D**2+B**2)*(gamma_0-gamma_inf)* \
    (arctan((log(v_0*rho)+D)/B)-arctan(D/B)))/B))/(B**2*T*((log(v_0*rho)+ \
    D)**2/B**2+1)*(1/rho))-(T_deb0*gamma_inf*(v_0*rho)**gamma_inf* \
    exp(((D**2+B**2)*(gamma_0-gamma_inf)*(arctan((log(v_0*rho)+D)/B)- \
    arctan(D/B)))/B))/(T/rho)))/(log(10)*T_deb0*(v_0*rho)**gamma_inf)
    return pas
```

```
def Ps(T,rho):
    ps = Pc(T,rho)+Pas(T,rho)+Pe(T,rho)
    if Mat == "Au":
        ps = Pc(T,rho)+0.4*Pas(T,rho)+Pe(T,rho)
    return ps


# Thermische Expansionskurve
def T_therm_exp(rho):
    v0 = v_0
    gamma_0 = gamma_0s
    T_deb0 = 1  # Wert, da in t_exp nicht gekuerzt
    t_exp = -1.30288344570976*R*(rho*v0)**gamma0e*(Bs**2*gamma_0 + Ds**2* \
    gamma_0 + 2.0*Ds*gamma_inf*log(rho) - 1.98502151557112*Ds*gamma_inf + \
    gamma_inf*log(rho)**2 - 1.98502151557112*gamma_inf*log(rho) + \
    0.98507760431994*gamma_inf)/(beta0*gamma0e*(Bs**2 + Ds**2 + 2.0*Ds*log(rho) \
    - 1.98502151557112*Ds + log(rho)**2 - 1.98502151557112*log(rho) + \
    0.98507760431994)) + 0.434294481903252*sqrt((rho*v0)**gamma0e* \
    (-10.6037962209568*Ac*Bs**4*beta0*gamma0e*v0_c*(rho*v0_c)**m - \
    21.2075924419136*Ac*Bs**2*Ds**2*beta0*gamma0e*v0_c*(rho*v0_c)**m - \
    42.4151848838272*Ac*Bs**2*Ds*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho) + \
    42.0975272906619*Ac*Bs**2*Ds*beta0*gamma0e*v0_c*(rho*v0_c)**m - \
    21.2075924419136*Ac*Bs**2*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho)**2 + \
    42.0975272906619*Ac*Bs**2*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho) - \
    20.8911243560724*Ac*Bs**2*beta0*gamma0e*v0_c*(rho*v0_c)**m - \
    10.6037962209568*Ac*Ds**4*beta0*gamma0e*v0_c*(rho*v0_c)**m - \
    42.4151848838272*Ac*Ds**3*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho) + \
    42.0975272906619*Ac*Ds**3*beta0*gamma0e*v0_c*(rho*v0_c)**m - \
    63.6227773257408*Ac*Ds**2*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho)**2 + \
    126.292581871986*Ac*Ds**2*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho) - \
    62.6733730682463*Ac*Ds**2*beta0*gamma0e*v0_c*(rho*v0_c)**m - \
    42.4151848838272*Ac*Ds*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho)**3 + \
    126.292581871986*Ac*Ds*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho)**2 - \
    125.346746136493*Ac*Ds*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho) + \
    41.4693313306198*Ac*Ds*beta0*gamma0e*v0_c*(rho*v0_c)**m - \
    10.6037962209568*Ac*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho)**4 + \
    42.0975272906619*Ac*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho)**3 - \
    62.6733730682463*Ac*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho)**2 + \
    41.4693313306198*Ac*beta0*gamma0e*v0_c*(rho*v0_c)**m*log(rho) - \
    10.2896893769503*Ac*beta0*gamma0e*v0_c*(rho*v0_c)**m - 10.6037962209568* \
    Bs**4*Bc*beta0*gamma0e*v0_c*(rho*v0_c)**n - 10.6037962209568*Bs**4*Cc*beta0* \
    gamma0e*v0_c*(rho*v0_c)**l + 9.0*Bs**4*R**2*gamma_0**2*(rho*v0)**gamma0e \
    - 21.2075924419136*Bs**2*Bc*Ds**2*beta0*gamma0e*v0_c*(rho*v0_c)**n - \
    42.4151848838272*Bs**2*Bc*Ds*beta0*gamma0e*v0_c*(rho*v0_c)**n*log(rho) + \
    42.0975272906619*Bs**2*Bc*Ds*beta0*gamma0e*v0_c*(rho*v0_c)**n - \
    21.2075924419136*Bs**2*Bc*beta0*gamma0e*v0_c*(rho*v0_c)**n*log(rho)**2 + \
    42.0975272906619*Bs**2*Bc*beta0*gamma0e*v0_c*(rho*v0_c)**n*log(rho) - \
    20.8911243560724*Bs**2*Bc*beta0*gamma0e*v0_c*(rho*v0_c)**n - \
    21.2075924419136*Bs**2*Cc*Ds**2*beta0*gamma0e*v0_c*(rho*v0_c)**l - \
    42.4151848838272*Bs**2*Cc*Ds*beta0*gamma0e*v0_c*(rho*v0_c)**l*log(rho) \
```

```
+ 42.0975272906619*Bs**2*Cc*Ds*beta0*gamma0e*v0_c*(rho*v0_c)**l - \
21.2075924419136*Bs**2*Cc*beta0*gamma0e*v0_c*(rho*v0_c)**l*log(rho)**2 + \
42.0975272906619*Bs**2*Cc*beta0*gamma0e*v0_c*(rho*v0_c)**l*log(rho) - \
20.8911243560724*Bs**2*Cc*beta0*gamma0e*v0_c*(rho*v0_c)**l + 18.0*Bs**2* \
Ds**2*R**2*gamma_0**2*(rho*v0)**gamma0e + 36.0*Bs**2*Ds*R**2*gamma_0* \
gamma_inf*(rho*v0)**gamma0e*log(rho) - 35.7303872802802*Bs**2*Ds*R**2* \
gamma_0*gamma_inf*(rho*v0)**gamma0e + 18.0*Bs**2*R**2*gamma_0*gamma_inf \
*(rho*v0)**gamma0e*log(rho)**2 - 35.7303872802802*Bs**2*R**2*gamma_0* \
gamma_inf*(rho*v0)**gamma0e*log(rho) + 17.7313968777598*Bs**2*R**2* \
gamma_0*gamma_inf*(rho*v0)**gamma0e - 10.6037962209568*Bc*Ds**4*beta0* \
gamma0e*v0_c*(rho*v0_c)**n - 42.4151848838272*Bc*Ds**3*beta0*gamma0e* \
v0_c*(rho*v0_c)**n*log(rho) + 42.0975272906619*Bc*Ds**3*beta0* \
gamma0e*v0_c*(rho*v0_c)**n - 63.6227773257408*Bc*Ds**2*beta0*gamma0e* \
v0_c*(rho*v0_c)**n*log(rho)**2 + 126.292581871986*Bc*Ds**2*beta0* \
gamma0e*v0_c*(rho*v0_c)**n*log(rho) - 62.6733730682463*Bc*Ds**2*beta0 \
*gamma0e*v0_c*(rho*v0_c)**n - 42.4151848838272*Bc*Ds*beta0*gamma0e* \
v0_c*(rho*v0_c)**n*log(rho)**3 + 126.292581871986*Bc*Ds*beta0*gamma0e*v0_c \
*(rho*v0_c)**n*log(rho)**2 - 125.346746136493*Bc*Ds*beta0*gamma0e*v0_c \
*(rho*v0_c)**n*log(rho) + 41.4693313306198*Bc*Ds*beta0*gamma0e*v0_c* \
(rho*v0_c)**n - 10.6037962209568*Bc*beta0*gamma0e*v0_c*(rho*v0_c)**n \
*log(rho)**4 + 42.0975272906619*Bc*beta0*gamma0e*v0_c*(rho*v0_c)**n* \
log(rho)**3 - 62.6733730682463*Bc*beta0*gamma0e*v0_c*(rho*v0_c)**n \
*log(rho)**2 + 41.4693313306198*Bc*beta0*gamma0e*v0_c*(rho*v0_c)**n \
*log(rho) - 10.2896893769503*Bc*beta0*gamma0e*v0_c*(rho*v0_c)**n - \
10.6037962209568*Cc*Ds**4*beta0*gamma0e*v0_c*(rho*v0_c)**l - \
42.4151848838272*Cc*Ds**3*beta0*gamma0e*v0_c*(rho*v0_c)**l*log(rho) + \
42.0975272906619*Cc*Ds**3*beta0*gamma0e*v0_c*(rho*v0_c)**l-63.6227773257408 \
*Cc*Ds**2*beta0*gamma0e*v0_c*(rho*v0_c)**l*log(rho)**2 + 126.292581871986* \
Cc*Ds**2*beta0*gamma0e*v0_c*(rho*v0_c)**l*log(rho) - 62.6733730682463*Cc* \
Ds**2*beta0*gamma0e*v0_c*(rho*v0_c)**l - 42.4151848838272*Cc*Ds*beta0*gamma0e \
*v0_c*(rho*v0_c)**l*log(rho)**3 + 126.292581871986*Cc*Ds*beta0*gamma0e*v0_c* \
(rho*v0_c)**l*log(rho)**2 - 125.346746136493*Cc*Ds*beta0*gamma0e*v0_c* \
(rho*v0_c)**l*log(rho) + 41.4693313306198*Cc*Ds*beta0*gamma0e*v0_c* \
(rho*v0_c)**l - 10.6037962209568*Cc*beta0*gamma0e*v0_c*(rho*v0_c)**l* \
log(rho)**4 + 42.0975272906619*Cc*beta0*gamma0e*v0_c*(rho*v0_c)**l* \
log(rho)**3 - 62.6733730682463*Cc*beta0*gamma0e*v0_c*(rho*v0_c)**l* \
log(rho)**2 + 41.4693313306198*Cc*beta0*gamma0e*v0_c*(rho*v0_c)**l* \
log(rho) - 10.2896893769503*Cc*beta0*gamma0e*v0_c*(rho*v0_c)**l + \
9.0*Ds**4*R**2*gamma_0**2*(rho*v0)**gamma0e + 36.0*Ds**3*R**2*gamma_0 \
*gamma_inf*(rho*v0)**gamma0e*log(rho) - 35.7303872802802*Ds**3*R**2* \
gamma_0*gamma_inf*(rho*v0)**gamma0e + 18.0*Ds**2*R**2*gamma_0*gamma_inf* \
(rho*v0)**gamma0e*log(rho)**2-35.7303872802802*Ds**2*R**2*gamma_0*gamma_inf* \
(rho*v0)**gamma0e*log(rho) + 17.7313968777598*Ds**2*R**2*gamma_0*gamma_inf* \
(rho*v0)**gamma0e + 36.0*Ds**2*R**2*gamma_inf**2*(rho*v0)**gamma0e*log(rho)**2 \
- 71.4607745605604*Ds**2*R**2*gamma_inf**2*(rho*v0)**gamma0e*log(rho) \
+ 35.4627937555197*Ds**2*R**2*gamma_inf**2*(rho*v0)**gamma0e + 36.0*Ds* \
R**2*gamma_inf**2*(rho*v0)**gamma0e*log(rho)**3 - 107.19116184084*Ds*R**2* \
gamma_inf**2*(rho*v0)**gamma0e*log(rho)**2 + 106.388381266559*Ds*R**2* \
gamma_inf**2*(rho*v0)**gamma0e*log(rho) - 35.1972043029964*Ds*R**2* \
gamma_inf**2*(rho*v0)**gamma0e + 9.0*R**2*gamma_inf**2*(rho*v0)**gamma0e* \
log(rho)**4 - 35.7303872802802*R**2*gamma_inf**2*(rho*v0)**gamma0e*log(rho)**3 \
+ 53.1941906332795*R**2*gamma_inf**2*(rho*v0)**gamma0e*log(rho)**2 - \
```

```
    35.1972043029964*R**2*gamma_inf**2*(rho*v0)**gamma0e*log(rho) + \
    8.73340097814798*R**2*gamma_inf**2*(rho*v0)**gamma0e))/(beta0*gamma0e* \
    (Bs**2 + Ds**2 + 2.0*Ds*log(rho) - 1.98502151557112*Ds + log(rho)**2 - \
    1.98502151557112*log(rho) + 0.98507760431994))
    t_exp = t_exp[len(r)//2 :]
    r_exp = r[len(r)//2 :]
    r_exp = r_exp[np.where(t_exp <= Tm0)]
    t_exp = t_exp[np.where(t_exp <= Tm0)]
    r_exp = r_exp[np.where(t_exp >= 0)]
    t_exp = t_exp[np.where(t_exp >= 0)]
    return r_exp, t_exp

if Mat == "Al":
    import pandas as pd

    df = pd.read_table("Thermal␣Expansion.txt", delim_whitespace=True)
    T_ = np.array(df["T"].tolist())
    alpha = np.array(df["alpha"].tolist())

    fig, ax_th_exp = plt.subplots()
    ax_th_exp.plot(T_, alpha*1e+6, "o", label = "Daten␣aus␣[40]")

    r_exp = T_therm_exp(r)[0]
    t_exp = T_therm_exp(r)[1]

    alpha = np.gradient((1/r_exp), t_exp*1e3)/((1/r_exp))
    ax_th_exp.plot(t_exp*1e3, alpha*1e+6, color = "red", \
                   label = "$\\alpha_{\mathrm{th}}$␣nach␣Gl.␣99")

    ax_th_exp.set_title("Wärmeausdehnungskoeffizient␣(Al)", \
                        fontsize=titlesize, pad=25)

    ax_th_exp.tick_params(axis="x", labelsize=xsize)
    ax_th_exp.tick_params(axis="y", labelsize=ysize)

    ax_th_exp.set_xlabel("$T$␣in␣K", fontsize=xlabelsize)
    ax_th_exp.set_ylabel("$\\alpha_{\mathrm{th}}$␣in␣$10^{-6}$/K", \
                         fontsize=xlabelsize)

    ax_th_exp.legend(fontsize = 26, loc = 'lower␣right')

    ax_th_exp.set_xlim(0,800)
    ax_th_exp.set_ylim(0,44)



# Schmelzkurve
def melting_curve(ps):
    A_melt = ps  <= p_melt    # Bereich über der Schmelzkurve
    ps[A_melt] = 0
    tm = np.argmin(np.abs(ps), axis = 0)
    t_m = t[tm]
```

```
    r_m = r
    t_m = t_m #[len(r)//2 :]
    r_m= r #[len(r)//2 :]
    #r_m= r_m[np.where(t_m >= Tm0)]
    #t_m = t_m[np.where(t_m >= Tm0)]
    return r_m, t_m

# Kalte Kompression
def cold_compression(rho_melt,Tm0):
    return r[:rho_melt], Tm0*np.ones_like(r[:rho_melt])


from Melting_curve import A_param, B_param

def P_melt(T):
    return A_param*((T/Tm0)**(1/B_param) - 1)


ps = np.zeros((len(r),len(t)))
ps[:,int(len(r)/1.5):] = Ps(T, rho)[:,int(len(r)//1.5):]



p_melt = P_melt(T)

r_m = melting_curve(ps)[0]
t_m = melting_curve(ps)[1]

r_exp = T_therm_exp(r)[0]
t_exp = T_therm_exp(r)[1]

p0 = 1e-4
A=ps<p0


r_m_r = r_m[np.where(t_m >= Tm0)]
t_m_r = t_m[np.where(t_m >= Tm0)]
r_m_l = r_m[np.where(t_m < Tm0)]
t_m_l = t_m[np.where(t_m < Tm0)]


# Liquid
T_sa = 4
T_deb0l = 174
gamma_0l = 1.55
T_ca = 25
sigma_0m = 0.873

from Liquid import A_m, C_m, B_l, D_l, sigma_T, T_T, T_sa


def Pm(T, rho):
```

```
    return -3*R*(-1.33333333333333*A_m*Tm0*v_0**2*(v_0/sigma_0m*rho)** \
           1.66666666666667/(sigma_0m**2/rho**3*(1 + 3*v_0/sigma_0m*rho)) \
           - 4*Tm0*v_0**2*(0.4*A_m*((v_0/sigma_0m*rho)**1.66666666666667 \
           - 1) + C_m)/(sigma_0m**2/rho**3*(1 + 3*v_0/sigma_0m*rho)) + 6* \
           Tm0*v_0**3*(0.4*A_m*((v_0/sigma_0m*rho)**1.66666666666667-1) + \
           C_m)/(sigma_0m**3/rho**4*(1 + 3*v_0/sigma_0m*rho)**2))


def Pt(T, rho):
    pt = -3*R*T**2*(v0_c*rho)**(-0.666666666666667)*(T + T_ca)*(T_T* \
    v0_c/((T + T_T)*(sigma_T + v0_c*rho)/rho)+1)*(-0.666666666666667*T_sa* \
    (v0_c*rho)**0.666666666666667*(T + T_deb0l*exp((B_l**2 + D_l**2)* \
    (gamma_0l - 0.666666666666667)*atan(B_l*log(v0_c*rho)/(B_l**2 + D_l* \
    (D_l + log(v0_c*rho))))/B_l))/(T/rho*(T + T_ca)) + T_deb0l*T_sa*(v0_c* \
    rho)**0.666666666666667*(B_l**2 + D_l**2)*(gamma_0l -0.666666666666667) \
    *(B_l*D_l*log(v0_c*rho)/((B_l**2 + D_l*(D_l + log(v0_c*rho))/rho)**2) - \
    B_l/((B_l**2 + D_l*(D_l + log(v0_c*rho)))/rho))*exp((B_l**2 + D_l**2)* \
    (gamma_0l - 0.666666666666667)*atan(B_l*log(v0_c*rho)/(B_l**2 + D_l* \
    (D_l+log(v0_c*rho))))/B_l)/(B_l*T*(T + T_ca)*(B_l**2*log(v0_c*rho)**2 \
    /(B_l**2 + D_l*(D_l + log(v0_c*rho)))**2 + 1)))/(2*T_sa*(T + T_deb0l* \
    exp((B_l**2 + D_l**2)*(gamma_0l - 0.666666666666667)*atan(B_l*log(v0_c* \
    rho)/(B_l**2 + D_l*(D_l + log(v0_c*rho))))/B_l))) - 3*R*T*(-T_T*v0_c/ \
    ((1/rho)**2*(T + T_T)*(sigma_T + v0_c*rho)) + T_T*v0_c**2/((1/rho)**3* \
    (T + T_T)*(sigma_T + v0_c*rho)**2))*log(T_sa*(v0_c*rho)** \
    0.666666666666667*(T + T_deb0l*exp((B_l**2 + D_l**2)*(gamma_0l - \
    0.666666666666667)*atan(B_l*log(v0_c*rho)/(B_l**2 + D_l*(D_l + \
    log(v0_c*rho))))/B_l))/(T*(T + T_ca)))/2
    return pt



def Pl(T, rho):
    return Pc(T, rho) + Pt(T, rho) + Pm(T, rho) + Pe(T, rho)


def melting_curve2(pl):
    pl_help = pl.copy()
    A_help = pl_help <= p_melt
    pl_help[A_help] = 0
    tm0 = np.argmin(np.abs(t-Tm0))
    t_l_r= t[np.argmin(np.abs(pl_help[tm0:-1]), axis = 0) + tm0 ] \
    * np.where(r > max(r)/3, 1, 0)
    t_l_r = t_l_r[np.argmin(np.abs(t_l_r - Tm0)) : ]
    t_l_r = t_l_r[t_l_r > Tm0]
    r_l_r = r[len(r)-len(t_l_r)-1:-1]

    pl_help = pl.copy()
    A_help = pl_help <= p_melt
    pl_help[A_help] = 0
    t_l_l= t[np.argmin(np.abs(pl_help[:tm0]), axis = 0)] * \
    np.where(r > max(r)/2, 1, 0)
```

```
    t_l_l = t_l_l[ : np.argmin(np.abs(t_l_l - Tm0)) ]
    t_l_l = t_l_l[t_l_l < Tm0]
    r_l_l = r[(len(r)-len(t_l_l)-len(t_l_r)-1) : (len(r)-len(t_l_r) -1)]

    r_l = np.append(r_l_l, r_l_r)
    t_l = np.append(t_l_l, t_l_r)


    r_l[np.argmin(t_l):]
    t_l[np.argmin(t_l):]

    return r_l, t_l



t_m0 = np.searchsorted(t, Tm0)
t_c = np.searchsorted(t, T_c)
r_c = np.searchsorted(r, rho_c)


pl = Pl(T, rho)
pl[:t_m0,:int(N/1.9)] = pl[:t_m0,:int(N/1.9)] * 0 # Korrektur

r_l = melting_curve2(pl)[0]
t_l = melting_curve2(pl)[1]


pl[(pl > p_melt)] = 0
pl[(Ps(T, rho) > p_melt)] = 0


pl[t_m0:,r_c:int(N/2)] = Pl(T, rho)[t_m0:,r_c:int(N/2)] # Korrektur

#pl[:t_m0] = 0

p_melt[(Ps(T, rho) > p_melt)] = 0
p_melt[(Pl(T, rho) < p_melt)] = 0

rho_melt = np.searchsorted(r, min(r_m))



# Gasphase

def van_der_Waals(Tr, vr):
    return (8*Tr/(3*vr - 1)  -  3/vr**2)


def spin(rho, Tm0):
    rho_melt = r[np.argmin(np.abs(rho - r_exp[0] ))]
    s = T_c*rho*v_c*(rho*v_c - 3)**2/4
    s[len(s)-len(t_l): np.argmin(np.abs(rho - r_exp[0] ))] = \
```

```
    s[len(s)-len(t_l): np.argmin(np.abs(rho - r_exp[0] ))]* \
    np.where(s[len(s)-len(t_l): np.argmin(np.abs(rho - r_exp[0] ))] > \
    t_l[len(s)-len(t_l): np.argmin(np.abs(rho - r_exp[0] ))] ,1, np.nan)
    s[np.argmin(np.abs(rho - r_exp[0] )) : ] = np.nan
    spin = s
    r_spin = rho    #[s>melting_curve2(pl)[1]] #[s>Tm0]
    return r_spin, spin

T_ , rho_ = np.meshgrid(spin(r, Tm0)[1], spin(r, Tm0)[0])
pl[(T <= T_)] = 0

p = pl + ps + p_melt
p[:t_m0,:int(N/1.9)] = p[:t_m0,:int(N/1.9)] * 0 # Korrektur




T_maxwell = t[(t > Tm0)]
T_maxwell = T_maxwell[(T_maxwell < T_c)]


rho_gas1 ,T_gas1 = np.meshgrid(r, T_maxwell)
p_gas1 = p_c*van_der_Waals(T_gas1/T_c, (1/rho_gas1)/v_c)

p_gas1[(rho_gas1 > rho_c)] = 0


rho_gas2, T_gas2 = np.meshgrid(r, t[(t>T_c)])
p_gas2 = p_c*van_der_Waals(T_gas2/T_c, (1/rho_gas2)/v_c)
p_gas2[(rho_gas2 > rho_c)] = 0


p_gas1 = p_c*van_der_Waals(T/T_c, (1/rho)/v_c)

p[t_m0:, :r_c] = p_gas1[t_m0:, :r_c]



v_r = np.linspace(0,50000,100000) # Gold
v_r = np.linspace(0,5000,500000)


bin_l = []
bin_r = []
p_l_g = np.zeros((len(t), len(r)))



fig, ax_PD = plt.subplots()
plt.subplots_adjust(top=0.915,
bottom=0.11,
left=0.056,
```

```
right=1.05,
hspace=0.2,
wspace=0.2)
tick_font_size = 18

fig, ax5 = plt.subplots()
ax5.set_xlim(0.4, 5)
ax5.set_ylim(-0.5, 1.5)
ax5.set_xlabel("$v_{\mathrm{r}}$", fontsize = xlabelsize)
ax5.set_ylabel("$p_{\mathrm{r}}$", fontsize = ylabelsize)
#ax5.set_title("Druck", fontsize = titlesize, pad = 25)

ax5.tick_params(axis="x", labelsize=xsize)
ax5.tick_params(axis="y", labelsize=ysize)


P_maxwell = []
for T_i in T_maxwell:
    T_r = T_i/T_c
    v_r = np.linspace(0,50000, 1000000)
    v_r = np.linspace(0,5000,1000000)
    def equations(z):

        v1, v2, p_maxwell = z

        return (-(8*T_r*np.log(3*v2-1))/3+p_maxwell*v2-3/v2+ \
                (8*T_r*np.log(3*v1-1))/3-p_maxwell*v1+3/v1,

                8*T_r/(3*v1 - 1) - 3/v1**2 - p_maxwell,

                8*T_r/(3*v2 - 1) - 3/v2**2 - p_maxwell)


    def sp(v_r):
        return (6/v_r**3 - 24*T_r/(3*v_r-1)**2)

    res1 = least_squares(sp, x0 = 0.4, bounds = ((0), (1)))
    res2 = least_squares(sp, x0 = 1.5, bounds = ((1), (200)))

    sp1 = res1.x
    sp2 = res2.x
    print(1/(sp1[0]*v_c))

    res=least_squares(equations, (v_r[np.argmax(van_der_Waals(T_r, v_r))], \
    sp2, -10), bounds =((v_r[np.argmax(van_der_Waals(T_r, v_r))],sp2 ,-10) \
    , (sp1, np.inf, 5000) )     )


    v1 = res.x[0]
    v2 = res.x[1]
    p_maxwell = res.x[2]
```

```
rho1 = 1/(v1*v_c)
rho2 = 1/(v2*v_c)
rho_sp1 = 1/(sp1[0]*v_c)

r_maxwell = r[np.searchsorted(r, rho1):np.searchsorted(r, 1)]

P_maxwell.append(p_maxwell)

r_maxwell_l = r_maxwell[(r_maxwell < r_c)]
r_maxwell_r = r_maxwell[(r_maxwell >= r_c)]

p[np.searchsorted(t, T_i), \
np.searchsorted(r, rho2):np.searchsorted(r, rho_sp1)] = p_maxwell*p_c
if len(bin_r) > 0:
    if rho2 < bin_r[-1]:
        bin_r.append(bin_r[-1])
    else:
        bin_r.append(rho2)
else:
    bin_r.append(rho2)

if len(bin_l) > 0:
    if rho1 > bin_l[-1]:
        bin_l.append(bin_l[-1])
    else:
        bin_l.append(rho1)
else:
    bin_l.append(rho1)


# Graphische Darstellung der Maxwell-Konstruktion

if T_i == max(T_maxwell):

    ax5.plot(v_r, van_der_Waals(T_r, v_r), color = "red", \
             label = "$T_\mathrm{c}$")
    ax5.plot(1, 1, "o", color="black")
    ax5.text(1.02,1.01, "$p_{\mathrm{r}}$", fontsize=24)
    ax5.legend(fontsize = 24)

if T_i == T_maxwell[np.searchsorted(T_maxwell, T_c*0.975)]:
    ax5.plot(v_r, van_der_Waals(T_r, v_r), color = "purple")
    ax5.plot([v1,v2], [p_maxwell, p_maxwell], color = "purple", \
             label = "0.975␣$T_\mathrm{c}$")
    ax5.plot([v1,v2], [p_maxwell, p_maxwell],"o", color = "black")

if T_i == T_maxwell[np.searchsorted(T_maxwell, T_c*0.95)]:
    ax5.plot(v_r, van_der_Waals(T_r, v_r), "b")
    ax5.plot([v1,v2], [p_maxwell, p_maxwell], color = "blue", \
             label = "0.95␣$T_\mathrm{c}$")
```

```
        ax5.plot([v1,v2], [p_maxwell, p_maxwell],"o", color = "black")

    if T_i == T_maxwell[np.searchsorted(T_maxwell, T_c*0.9)]:
        ax5.plot(v_r, van_der_Waals(T_r, v_r), color = "orange")
        ax5.plot([v1,v2], [p_maxwell, p_maxwell], color = "orange", \
                 label = "0.9 $T_\mathrm{c}$")
        ax5.plot([v1,v2], [p_maxwell, p_maxwell],"o", color = "black")



if Mat == "Al":
    ax_PD.plot(2.63, Tm0, "ob")

ax_PD.plot(rho_c, T_c, "ob", label = "Konstruktion")

if Mat == "Al":
    ax_PD.text(2.72, 0.75, r'$T_{m0}$', color = "blue", fontsize=20)
    ax_PD.text(rho_c-0.075, T_c-0.3, "$p_c$", color = "blue", fontsize=20)
    ax_PD.text(0.25, 8, r'$G$', fontsize=32)
    ax_PD.text(2.2, 7, r'$L$', fontsize=32)
    ax_PD.text(1.5, 5, r'$[L]$', fontsize=26)
    ax_PD.text(0.1, 5, r'$[G]$', fontsize=26)
    ax_PD.text(0.65, 4.3, r'$[G+L]$', fontsize=26)
    ax_PD.text(2.5, 0.15, r'$[S]$', fontsize=22)
    ax_PD.text(2.2, 0.15, r'$[S+L]$', fontsize=20)
    ax_PD.text(3.15, 2.6, r'$S+L$', fontsize=22)
    ax_PD.text(3.2, 1.1, r'$S$', fontsize=26)


# Graphische Darstellung Druck

rho_melt = np.searchsorted(r, min(r_m_r))
ax_PD.plot(cold_compression(rho_melt, Tm0)[0], \
cold_compression(rho_melt, Tm0)[1], color = "red")

ax_PD.plot(spin(r, Tm0)[0], spin(r, Tm0)[1], color = "blue")
ax_PD.plot(r_l , t_l)

ax_PD.plot(r_exp, t_exp)
ax_PD.plot(r_m_r, t_m_r, color = "orange")
ax_PD.plot(r_m_l, t_m_l,  "-.", color = "orange")

ax_PD.plot(bin_l, T_maxwell, color = "blue")
ax_PD.plot(bin_r, T_maxwell, color = "blue")

p_min = -20
p_max = 80

c = ax_PD.pcolormesh(rho, T, p, cmap ='seismic', vmin = p_min, \
                     vmax = p_max, norm=MidpointNormalize(p_min, p_max, 0.))
```

```
cbar = fig.colorbar(c, ax = ax_PD)
cbar.ax.tick_params(labelsize=tick_font_size)
cbar.set_label(label="$p$ in GPa",fontsize = 28)

ax_PD.set_title('matplotlib.axes.Axes.imshow() Examples')

ax_PD.set_xlabel(r'$\rho$ in $\mathrm{g}/\mathrm{cm}^3$',fontsize=xlabelsize)
ax_PD.set_ylabel(r'$T$ in kK',  fontsize = ylabelsize)

ax_PD.set_title("Druck", fontsize = titlesize , pad = 25)

ax_PD.tick_params(axis="x", labelsize=xsize)
ax_PD.tick_params(axis="y", labelsize=ysize)



if Mat == "Au":
    ax_PD.plot(196.97/26, 7.400, "o", color = "red", label = "[47]")

    ax_PD.errorbar(196.97/(26), 7.400, xerr=196.97/(26)  \
                -  196.97/(26+5), yerr=1.1, color = "red")

    ax_PD.plot(196.97/(45), 8.100, "o",  color = "green", label = "[49]")
    ax_PD.plot(196.97/(40), 8.267, "o",  color = "orange", label = "[50]")
    ax_PD.legend(title="kritische Punkte aus", title_fontsize=24, fontsize=22)
```

```
fig , ax = plt.subplots(1,2)
#fig.set_figheight(1)
ax[0].plot(cold_compression(rho_melt, Tm0)[0], \
cold_compression(rho_melt, Tm0)[1], color = "red")

ax[0].plot(spin(r, Tm0)[0], spin(r, Tm0)[1])
ax[0].plot(r_l , t_l)

ax[0].plot(r_exp, t_exp)
ax[0].plot(r_m_r, t_m_r, color = "orange")
ax[0].plot(r_m_l, t_m_l,  "-.", color = "orange")

ax[0].plot(bin_l, T_maxwell)
ax[0].plot(bin_r, T_maxwell)

p_min = -5
p_max = 10

c = ax[0].pcolormesh(rho, T, Pe(T, rho), cmap ='seismic', vmin = p_min, \
```

```
vmax = p_max, norm=MidpointNormalize(p_min, p_max, 0.))

cbar = fig.colorbar(c, ax = ax[0])
cbar.ax.tick_params(labelsize=tick_font_size)
#cbar.set_label(label="$p$ in GPa",fontsize = 28)

ax[0].set_title('matplotlib.axes.Axes.imshow()␣Examples')

ax[0].set_xlabel(r'$\rho$␣in␣$\mathrm{g}/\mathrm{cm}^3$', \
fontsize = xlabelsize)
ax[0].set_ylabel(r'$T$␣in␣kK',  fontsize = ylabelsize)

ax[0].set_title("$p_{\mathrm{e}}$␣in␣GPa",fontsize=titlesize, pad=25)

ax[0].tick_params(axis="x", labelsize=xsize)
ax[0].tick_params(axis="y", labelsize=ysize)



ax[1].plot(cold_compression(rho_melt, Tm0)[0], \
cold_compression(rho_melt, Tm0)[1], color = "red")

ax[1].plot(spin(r, Tm0)[0], spin(r, Tm0)[1])
ax[1].plot(r_l , t_l)

ax[1].plot(r_exp, t_exp)
ax[1].plot(r_m_r, t_m_r, color = "orange")
ax[1].plot(r_m_l, t_m_l,  "-.", color = "orange")

ax[1].plot(bin_l, T_maxwell)
ax[1].plot(bin_r, T_maxwell)

p_min = -20
p_max = 50

c = ax[1].pcolormesh(rho, T, p-Pe(T, rho), cmap ='seismic', vmin = \
p_min, vmax = p_max, norm=MidpointNormalize(p_min, p_max, 0.))

cbar = fig.colorbar(c, ax = ax[1])
cbar.ax.tick_params(labelsize=tick_font_size)
cbar.set_label(label="$p$␣in␣GPa",fontsize = 28)

ax[1].set_title('matplotlib.axes.Axes.imshow()␣Examples')
ax[1].set_xlabel(r'$\rho$␣in␣$\mathrm{g}/\mathrm{cm}^3$', \
fontsize = xlabelsize)
#ax[1].set_ylabel(r'$T$ in kK',  fontsize = ylabelsize)

ax[1].set_title("$p_{\mathrm{ph}}␣+␣p_{\mathrm{c}}$␣in␣GPa", \
fontsize = titlesize, pad = 25)

ax[1].tick_params(axis="x", labelsize=xsize)
ax[1].tick_params(axis="y", labelsize=ysize)
```

```
# Interpolant Druck

p_e = Pe(T,rho)


# ----------------------------

fig , ax = plt.subplots()

p_min = 0 #min(p)
p_max = 20 #max(p)
c = ax.pcolormesh(rho, T, p_e, cmap ='seismic', vmin = p_min, \
vmax = p_max, norm=MidpointNormalize(p_min, p_max, 0.))

fig.colorbar(c, ax = ax)
ax.set_title('matplotlib.axes.Axes.imshow() Examples')
ax.set_xlabel(r'Dichte $\rho$ / g/cm^3', fontsize = 22)
ax.set_ylabel(r'Temperatur $T$ / $10^3$ K', fontsize = 22)
ax.set_title("Druck in GPa", fontsize = 24)
# ----------------------------


from scipy import interpolate

p_e_inter = interpolate.interp2d(r, t, p_e)
p_ph = p - p_e
p_ph_inter = interpolate.interp2d(r, t, p)

np.save("p_ph_inter", p_ph_inter )
np.save("p_e_inter", p_e_inter )
```

Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 5. Juni 2023