Daniel Lüftner

Elastische Eigenschaften von NiTiHf Legierungen

Eine Ab-initio Studie

Diplomarbeit

Lehrstuhl für Atomistic Modelling and Design of Materials
Department für Materialphysik
Montanuniversität Leoben

Leoben, September 2011

Danksagung

Ich möchte mich sehr herzlich bei allen Mitarbeitern des Instituts für Atomistic Modelling and Design of Materials bedanken:

Bei der Leiterin des Instituts Frau Univ.-Prof. Dr.h.c. Mag. et Dr.rer.nat. Claudia Draxl für die Möglichkeit meine Diplomarbeit auf diesem Institut zu schreiben. Bei Herrn Priv.-Doz. Dipl.-Ing. Dr.rer.nat. Peter Puschnig, dem Betreuer der Diplomarbeit, für die Unterstützung während der Arbeit und der Hilfe beim Erstellen dieser. Außerdem möchte ich mich bei den Herrn Dr. Jürgen Spitaler und MSc. Rostam Golesorkhtabar bedanken, die mich unterstützt haben und sich immer Zeit genommen haben Fragen zu beantworten.

Der österreichischen Bundesregierung (insbesondere dem Bundesministerium für Verkehr, Innovation und Technologie und dem Bundesministerium für Wirtschaft, Familie und Jugend) sowie dem Land Steiermark, vertreten durch die Österreichische Forschungsförderungsgesellschaft mbH und die Steirische Wirtschaftsförderungsgesellschaft mbH, wird für die finanzielle Unterstützung der Forschungsarbeiten im Rahmen des von der Materials Center Leoben Forschung GmbH abgewickelten K2 Zentrums für „Materials, Processing and Product Engineering“ im Rahmen des Österreichischen COMET Kompetenzzentren Programms sehr herzlich gedankt.

Zuletzt möchte ich mich besonders bei meinen Eltern und bei meiner Freundin Christine bedanken, die mich während des Studiums immer unterstützt haben.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Seit der Entdeckung des Formgedächtniseffekts haben sich zahlreiche Anwendungen für Materialen mit diesen Eigenschaften durchgesetzt. Die Anwendungsbereiche sind dabei sehr vielfältig und reichen von der Autoindustrie bis hin zur Medizintechnik[1]. Grundlage für den Formgedächtniseffekt ist eine martensitische Umwandlung bei tiefen Temperaturen. Durch diese Umwandlung werden Phänomene wie die sogenannte Superelastizität oder der Formgedächtniseffekt erst möglich. Die wohl bekannteste und auch am häufigsten eingesetzte Legierung, die dieses Verhalten zeigt ist Nitinol, welche zu 50 Prozent aus Nickel und zu 50 aus Titanium besteht. Es ist bekannt [2], dass sich durch Zulegieren von Hafnium die Übergangstemperatur der Phasenumwandlung zu höheren Werten verschiebt. Ziel dieser Arbeit ist es, die in dem Nickel-Titanium- und im Nickel-Titanium-Hafnium-System auftretenden Phasen einer Untersuchng der elastischen Konstanten zu unterziehen. Die elastischen Konstanten können einerseits zum Verständnis der martensitischen Phasenumwandlung beitragen und werden anderseits als Eingabeparameter für die mikromechanische Simulation der martensitischen Umwandlung benötigt. Im Rahmen dieser Arbeit erfolgt die Bestimmung der elastischen Konstanten *ab-initio* mit Hilfe der Dichtefunktionaltheorie, kurz DFT. Die verwendeten DFT-Codes sind der exciting Code [3, 4] und Wien2k [5].

In dieser Arbeit werden zunächst alle nötigen theoretischen Hintergründe, die zum Verständnis der durchgeführten Berechnungen beitragen, erläutert. Dies sind zunächst die Grundzüge der Dichtefunktionaltheorie, und im speziellen die Art des eingesetzten Basissatzes. Im Anschluss daran werden die wichtigsten Phänomene, die Materialen, die den Formgedächtniseffekt zeigen, die metallphysikalischen Vorgänge und ihre Voraussetzungen beschrieben. Danach werden das Phasendiagramm und die darin vorkommenden und für den Formgedächtniseffekt bedeutenden Phasen beschrieben. Der letzte Teil der theoretischen Literaturstudie betrifft die elastischen Konstanten. Zunächst wird ihre Rolle bei elastischen Deformationen diskutiert und auch, wie sie mit Hilfe von DFT-Berechnugnen er-

mittelt werden können und zuletzt ihren Einfluss auf die martensitische Phasenumwandlung.

Das nächste Kapitel der Arbeit beschäftigt sich mit einem, für den weiteren Verlauf der Berechnungen benötigten, Optimierungsprogramm. Die wesentlichen Funktionen und Algorithmen werden darin beschrieben. Es folgen die Presentation aller Ergebnisse, die im Laufe der durchgeführten Berechnungen erhalten wurden. Dies sind unter anderem die elastischen Konstanten der elementaren Festkörper von Nickel, Titanium und Hafnium, die der binären Nickel-Titanium Legierungen und zuletzt von zwei verschiedenen ternären Nickel-Titanium-Hafnium Legierungen mit unterschiedlicher Zusammensetzung.

Kapitel 2

Grundlagen

2.1 Grundzüge der Dichtefunktionaltheorie

2.1.1 Allgemein

Festkörper kann man sich vereinfacht als eine Ansammlung einer großen Zahl Atome mit zugehörigen Elektronen aufgebaut vorstellen. Um Eigenschaften eines Festkörpers durch theoretische Berechnungen beziehungsweise Simulationen zu bestimmen, muss das Quanten-Vielteilchenproblem gelöst werden. In der folgenden Gleichung ist der Hamiltonoperator für einen solchen Fall definiert [6]:

$$\begin{aligned}\hat{H} = \sum_{i=1}^{N_K}\left(-\frac{\hbar^2}{2M_i}\right)\triangle_i + \sum_{j=1}^{N_{el}}\left(-\frac{\hbar^2}{2m}\right)\triangle_j - \frac{1}{4\pi\varepsilon_0}\sum_{i,j}\frac{e^2 Z_i}{|\overrightarrow{R_i} - \overrightarrow{r_j}|} \\ + \frac{1}{8\pi\varepsilon_0}\sum_{i\neq j}\frac{e^2}{|\overrightarrow{r_i} - \overrightarrow{r_j}|} + \frac{1}{8\pi\varepsilon_0}\sum_{i\neq j}\frac{e^2 Z_i Z_j}{|\overrightarrow{R_i} - \overrightarrow{R_j}|}\end{aligned} \tag{2.1}$$

Die ersten beiden Terme der Gleichung sind kinetische Energieoperatoren für die Bewegung der Atomkerne und der Elektronen. Die weiteren Terme beschreiben die Coulombwechselwirkung zwischen Kernen und Elektronen, den Elektronen untereinander und die der Kerne untereinanander. Da diese Gleichung zu komplex ist, um sie exakt zu lösen, müssen vorher noch einige Vereinfachungen getroffen werden. Eine erste Möglichkeit das Problem zu vereinfachen wurde von Born und Oppenheimer veröffentlicht [7].

Born-Oppenheimer Näherung

Da die Masse von Atomkernen um einige Größenordnungen höher liegt als die der Elektronen und diese aufgrund der Trägheit um vieles langsamer sind, kann man sich die Kerne an ihrer Position eingefroren vorstellen, ihre Geschwindigkeit wird also zu Null. Im Hamiltonoperator des Vielteilchenproblems fällt somit der kinetische Term der Kerne vollständig weg und der letzte Term wird zu einer Konstante. Zurück bleibt ein System aus Z sich bewegender und interagierender Elektronen pro Atomkern. Von den Atomkernen bleibt ein positives Potential zurück, das sogenannte externe Potential. Gleichung 2.1 vereinfacht sich damit zu:

$$\hat{H} = \sum_{j=1}^{N_{el}} \left(-\frac{\hbar^2}{2m} \right) \triangle_j - \frac{1}{4\pi\varepsilon_0} \sum_{i,j} \frac{e^2 Z_i}{|\overrightarrow{R_i} - \overrightarrow{r_j}|} + \frac{1}{8\pi\varepsilon_0} \sum_{i \neq j} \frac{e^2}{|\overrightarrow{r_i} - \overrightarrow{r_j}|} + V_{ext} \quad (2.2)$$

Durch die Born-Oppenheimer-Näherung hat sich das Problem zwar erheblich vereinfacht, bleibt aber weiterhin ein Vielteilchenproblem, welches für die große Anzahl an Elektronen im Festkörper nicht handelbar ist.

Das Prinzip von Hohenberg und Kohn

Im Lauf der Zeit gab es einige Ansätze dazu [8], um dieses Problem zu umgehen; einer, welcher auch für Festkörper zu guten Ergebnissen führt, ist die DFT welche sich auf ein Theorem von Hohenberg und Kohn [9] stützt, das 1964 veröffentlicht wurde. Dieses Hohenberg-Kohn theorem lautet wie folgt:

Bewegt sich ein System von interagierenden Elektronen in einem externen Potential, so ist die Gesamtenergie ein Funktional der Elektronendichte ρ und hat die Form:

$$E[\rho] = F_{HK}[\rho] + \int \rho(\overrightarrow{r}) V_{ext}(\overrightarrow{r}) dr \quad (2.3)$$

Hierbei bezeichnet F_{HK} ein universelles Funktional der Elektronendichte ρ und V_{ext} ist das elektrostatische Potential der Atomkerne. Hohenberg und Kohn konnten zeigen, dass die Grundzustandselektronendichte zu dem externen Potential das Energiefunktional minimiert, welches dann der Grundzustandsgesamtenergie entspricht.

Die Kohn-Shamgleichung

Kohn und Sham haben 1965 vorgeschlagen [10], wie aus dem Hohenberg-Kohn-Prinzip ein praktisch anwendbares Werkzeug abgeleitet werden kann. Dazu lässt sich das Hohenberg-Kohn-Funktional zunächst in folgende Form umschreiben:

$$F_{HK}[\rho] = T_0[\rho] + E_H[\rho] + E_{xc}[\rho] \tag{2.4}$$

Der erste Term steht für die kinetische Energie nicht wechselwirkender Elektronen, der zweite für die sogenannte Hartree-Energie und der letzte für die Austauschkorrelationsenergie. Das Gesamtenergiefunktional wird nun zu:

$$E[\rho] = T_0[\rho] + E_H[\rho] + E_{xc}[\rho] + E_{ext}[\rho] \tag{2.5}$$

Dieser Ausdruck kann als das Energiefunktional eines *nicht interagierenden* klassischen Elektronengases unter dem Einfluss von einem effektiven Potential interpretiert werden. Dieses effektive Potential setzt sich aus drei Anteilen zusammen. Dies sind der sogenannte Hartree-Term $V_H[\rho]$, das externe Potential $V_{ext}[\rho]$, das von den Atomkernen herrührt und das Austauschkorrelationspotential $V_{xc}[\rho]$. Der dazugehörige Hamiltonoperator, auch Kohn-Sham Hamiltonoperator genannt, lautet:

$$\hat{H}_{KS} = -\frac{\hbar^2}{2m_e}\overrightarrow{\nabla}^2 + V_H + V_{xc} + V_{ext} \tag{2.6}$$

Mit diesem Operator lassen sich jetzt die Kohn-Sham-Gleichungen aufstellen. Diese lauten:

$$\hat{H}_{KS}\Phi_i(\overrightarrow{r}) = \varepsilon_i\Phi_i(\overrightarrow{r}) \tag{2.7}$$

Bei diesen Gleichungen handelt es sich nunmehr um Ein-Teilchen Schrödingergleichungen. Die Eigenwerte ε_i sind die sogenannten Kohn-Sham-Energien, die $\Phi_i(\overrightarrow{r})$ die Ein-Teilchen-Wellenfunktionen. Aus den N niedrigsten Lösungen der Kohn-Sham Gleichungen [6] lässt sich nun die Grundzustandsdichte ermitteln.

$$\rho(\overrightarrow{r}) = \sum_{i=1}^{N} \Phi_i(\overrightarrow{r})^*\Phi_i(\overrightarrow{r}) \tag{2.8}$$

Es sei jedoch darauf hingewiesen, dass es sich bei den Ein-Teilchen Wellenfunktionen $\Phi_i(\overrightarrow{r})$ nicht um Wellenfunktionen von Elektronen handelt,

sondern vielmehr um Hilfs-Wellenfunktionen ohne wirkliche physikalische Bedeutung. Nur die Elektronendichte berechnet nach Gleichung 2.8 ist gleich der tatsächlichen Grundzustandsdichte des wechselwirkenden Elektronengases. [6]

Da aber sowohl der Hartree Operator als auch der Austauschkorrelationsoperator selbst von der Grundzustandsdichte abhängen, kann das Problem nur selbstkonsistent gelöst werden. Dabei wird folgendermaßen vorgegangen: Mit einer genäherten Startdichte $\rho_0(\overrightarrow{r})$ wird der Hamiltonoperator $\hat{H}$ aufgestellt. Das Eigenwertproblem wird gelöst und man erhält einen Satz von Φ_i , mit welchem die neue Elektronendichte bestimmt wird. Das Ergebnis wird mit der Startdichte verglichen und der Zyklus solange wiederholt, bis das Ergebnis zu einer Elektronendichte $\rho_f(\overrightarrow{r})$ konvergiert. Dieser Zusammenhang ist grafisch in Abbildung 2.1 zusammengefasst.

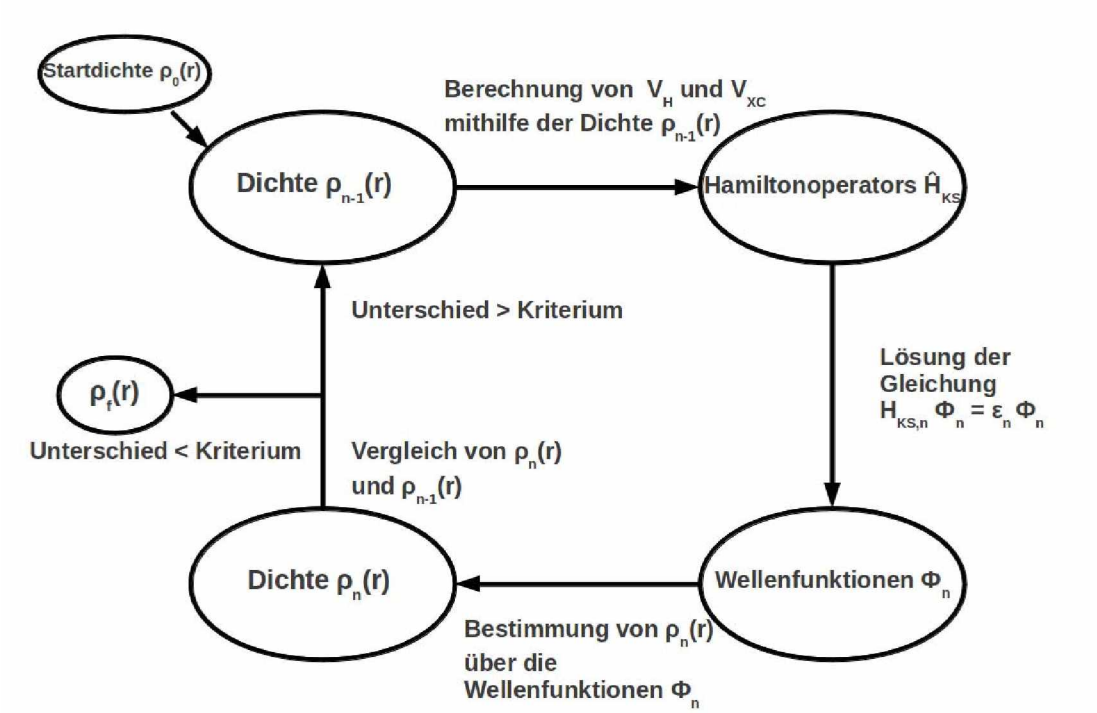

Abbildung 2.1: Grafische Darstellung des Selbskonsistenzzyklus für Berechnung der Elektronendichte im Rahmen der DFT. Der Index n steht dabei für den aktuellen Zyklus

Bei der Bestimmung der Dichte von periodischen Systemen muss auch über die Anzahl der k-Punkte in der Brillouinzone summiert werden:

$$\rho(\overrightarrow{r}) = \sum_{k}^{BZ} \sum_{n}^{bes.} |\Phi_i(\overrightarrow{r})|^2 \tag{2.9}$$

Die k-Punkte geben die Anzahl der Netzpunkte im reziproken Raum an und für sie gilt, umso mehr davon in der Brillouinzone sind, desto genauer wird das Ergebnis. Da aber für jeden k-Punkt die Kohn-Sham Gleichungen gelöst werden müssen, ist es andererseits unerlässlich, die Dichte des k-Punktnetzes möglichst klein zu halten. Aus diesem Grund müssen Konvergenztests für die Größe des k-Punktnetzes gemacht werden. In diesem Zusammenhang sei auch der sogenannte smearing-Parameter σ. Dieser Parameter ist vor allem wichtig für Metalle, da hier die Fermienergie innerhalb eines Bandes liegt. Das heißt, man muss über eine Funktion integrieren, die diskontinuierlich und daher nicht ableitbar ist. Dies führt zu einer sehr langsamen Konvergenz bezüglich des k-Punktnetzes und man bräuchte ein sehr dichtes, k-Punktnetz um genügend genaue Ergebnisse zu bekommen [11] was in langen Rechenzeiten resultiert. Um diesem Problem entgegenzuwirken, führt man künstlich eine Verschmierung der Beiträge der einzelnen k-Punkte zur Dichte ein. Dazu gibt es mehrere Ansätze, einer ist zum Beispiel eine Gauß'sche Funktion einzusetzen. Die Breite dieser Verschmierung ist der vorhin genannte Parameter σ. Durch diese Verschmierung konvergiert die Zielgröße der Konvergenztests bereits bei kleinerer Zahl der k-Punkte, jedoch kann das Ergebnis für zu große Werte von σ verfälscht werden. Das heißt, die Anzahl der k-Punkte und der Smearing-Parameter σ sind nicht unabhängig voneinander, und man muss diese Tatsache bei den Konvergenztests berücksichtigten.

Das Austauschkorrelationspotential

Das Austauschkorrealtionspotential ist bestimmt durch die funktionale Ableitung der Austauschkorrelationsenergie nach der Elektronendichte:

$$V_{xc} = \frac{\delta E_{xc}[\rho]}{\delta \rho} \tag{2.10}$$

Da die exakte Form des Austauschkorrelationsfunktionals allerdings nicht bekannt ist, kommt es an dieser Stelle zu Näherungen in der Theorie. Zu den in der Materialwissenschaft am häufigsten eingesetzten Näherungen gehören die „Local Density-Approximation“ (LDA) und die „General-Gradient-Approximation“ (GGA).

Die „Local Density-Approximation“ [12] ist die einfachste Form einer Näherung für das Austauschkorrelationspotential. Die Berechnung des Funktionals erfolgt dabei nach folgender Gleichung:

$$E_{xc}^{LDA} = \int \rho(\overrightarrow{r})\epsilon_{xc}(\rho(\overrightarrow{r}))d\overrightarrow{r} \tag{2.11}$$

Hierbei ist ϵ_{xc} die Austausch-Korrelationsenergie eines homogenen Elektronengases mit der Dichte ρ. Diese kann für ein homogenes Elektronengas numerisch bestimmt werden. Die Austauschkorrelationsenergie für eine bestimmte Dichteverteilung $\rho(\overrightarrow{r})$ wird nun dadurch bestimmt, dass der Raum in infinitesimal kleine Volumina mit konstanter Dichte aufgeteilt wird. Jedes dieser kleinen Volumina liefert einen Beitrag zur gesamten Austauschkorrelationenergie, die einem gleichen Volumen welches mit einem homogenen Elektronengas gefüllt ist entspricht. Über all diese Volumina wird intergriert und man erhält E_{xc}^{LDA} [13]. Diesen Zusammenhang beschreibt Gleichung 2.11. Die LDA liefert gute Ergebnisse für Systeme, in denen sich die Dichte nur langsam ändert.

Eine verbesserte Näherung für E_{xc} ist, nicht nur die lokale Dichte in einem Volumselement, sondern auch die Änderung der Dichte zu benachbarten Elementen in den Beitrag zur gesamten Energie miteinzubeziehen. Diesen Weg geht die "Generalisierte Gradienten-Näherung"kurz GGA [14], die wie folgt definiert ist:

$$E_{xc}^{GGA} = \int \rho(\overrightarrow{r})\epsilon_{xc}(\rho(\overrightarrow{r}), |\overrightarrow{\nabla}\rho(\overrightarrow{r})|)d\overrightarrow{r} \tag{2.12}$$

Die GG-Näherung zeigt verbesserte Ergebnisse für strukturelle Eigenschaften wie zum Beispiel die Gitterkonstanten für Systeme in denen sich die Dichte mit dem Ort stärker ändert. Ein Nachteil der Näherung ist jedoch, dass es nicht nur eine Variante wie bei der LDA sondern mehrere Möglichkeiten gibt die Ableitung der Dichte einzubauen. Es sei hier nur kurz erwähnt, dass es noch eine Vielzahl weiterer Näherungen gibt, auf diese aber an dieser Stelle nicht weiter eingegangen werden soll.

Lösen der Kohn-Sham-Gleichungen

Die Kohn-Sham Gleichungen 2.7 sollen nun für kristalline Festkörper, das heißt für eine räumlich periodische Anordnung von Atomen gelöst werden. Um dies zu bewerkstelligen, bedient man sich folgenden Ansatzes: Man wählt eine Linearkombination von Basisfunktionen $\Phi_p(\overrightarrow{r})$, welche der Symmetrie des Problems angepasst sind. Nun muss man die Koeffizienten c_p^i der Basisfunktionen $\Phi_p(\overrightarrow{r})$ bestimmen, aus denen man dann die Wellenfunktion $\psi_i(\overrightarrow{r})$ darstellen kann.

$$\psi_i(\overrightarrow{r}) = \sum_{p=1}^{P} c_p^i \Phi_p(\overrightarrow{r}) \tag{2.13}$$

Um das exakte Ergebnis zu bekommen müsste die Anzahl der Basisfunktionen P unendlich groß sein, was praktisch aber nicht durchführbar ist. Man muss nun versuchen einen endlich großen Satz an Basisfunktionen zu finden, mit denen das Ergebnis für die Wellenfunktion dem exakten möglichst nahe kommt. Wenn so ein Basissatz mit einer endlichen Anzahl an Gleichungen P gefunden ist, lässt sich Gleichung 2.13 in Gleichung 2.7 einsetzen, und man erhält durch weitere Umformung die sogenannte Säkulargleichung:

$$\sum_{q=1}^{P}(\langle\Phi_p|H_{KS}|\Phi_q\rangle - \varepsilon_i\langle\Phi_p|\Phi_q\rangle)c_q^i = 0 \tag{2.14}$$

Löst man diese Gleichung, erhält man die Koeffizienten c_p^i und die Eigenwerte ε_i und das Problem ist gelöst. Die Kunst besteht nun darin, einen geeigneten Basissatz zu finden, welcher einerseits möglichst klein sein soll, um die Rechenzeit kurz zu halten und andererseits groß genug sein muss, damit die Wellenfunktion genau genug beschrieben werden. Eine dieser Methoden ist die der „Linear-Augmented-Plane-Waves“ (LAPW).

2.1.2 LAPW-Methode

In diesem Kapitel wird die Methode der „Linear-Augmented-Plane-Waves“ [15] erklärt. Im Gegensatz zu Pseudopotential-Methoden [16], bei denen nur ebene Wellen als Basissatz verwendet werden, können bei Verwendung eines lokalen Basissatzes auch tiefer liegende Zustände behandelt werden, das heißt man kann alle Elektronen behandeln.

APW-Methode

Als "Vorstufe" zur LAPW-Methode, wird zunächst auf die Methode der "Augmented-Plane-Waves“ [17] eingegangen. Betrachtet man einen Festkörper, so spüren Elektronen ein unterschiedliches Potetential je nachdem, ob sie sich nah bei den Atomkernen aufhalten oder weiter weg von diesen im Zwischenraum sind. Diese Tatsache macht sich die sogenannte "muffin-tin Näherung" [17] zu Nutze, welche bei der APW-Methode zum Einsatz kommt. Die Einheitszelle wird dabei in zwei Bereiche aufgeteilt. Um jedes Atom herum verhalten sich die Elektronen eher wie in einem freien Atom und werden dort am besten durch atomähnliche Funktionen beschrieben. Diesen Bereich nennt man "Muffin-tin region"(S_α). Es sind kugelförmige Bereiche mit dem Radius R_{MT} . Im Zwischenraum weit weg von den Kernen verhalten sich die Elektronen wie in einem freien Elektronengas und werden am besten durch Ebene-Wellenfunktionen beschrieben. Diesen Bereich nennt man "Interstitial region"(I). Ebene Wellen sind Eigenwerte eines

Hamiltonoperators mit konstantem Potential. Auf diese Weise sind die Randbedingungen, die sich aus dem Bloch Theorem [18] ergeben erfüllt. Die Augmented Plane Waves sind wie folgt definiert:

$$\Phi_{\vec{G}}^{\vec{k}}(\vec{r}, E) = \begin{cases} \frac{1}{\sqrt{\Omega}} e^{i(\vec{k}+\vec{G})\vec{r}} & \vec{r} \epsilon I \\ \sum_{l,m} A_{lm}^{\alpha}(\vec{k} + \vec{G}) u_l^{\alpha}(r, E) Y_m^l(\hat{r}) & \vec{r} \epsilon S_{\alpha} \end{cases} \tag{2.15}$$

Hier ist Ω das Volumen der Einheitszelle und Y_m^l sind die Kugelfunktionen. $u_l^{\alpha}(r, E)$ sind die radialen Lösungen der Schrödingergleichungen für ein freies Atom α in einem sphärischen Potential. Die Koeffizienten $A_{lm}^{\alpha}(\vec{k} + \vec{G})$ müssen für jedes Atom so bestimmt werden, dass die Bedingung der Stetigkeit der Funktion $\Phi_{\vec{G}}^{\vec{k}}(r, E)$ in der gesamten Zelle gegeben ist. Diese Bedingung ist überall erfüllt außer auf der Oberfläche der Muffin-tin Kugel. Damit die Bedingung auch dort erfüllt ist, werden die $A_{lm}^{\alpha}(\vec{k} + \vec{G})$ bestimmt, indem die Werte für die ebenen Wellen und die der Funktionen innerhalb der Kugel an der Oberefläche gleichgesetzt werden. Damit die beiden Werte exakt gleich groß sind, bräuchte man prinzipiell eine unendliche Anzahl an Koeffizienten $A_{lm}^{\alpha}(\vec{k} + \vec{G})$. Praktisch wählt man einen Wert l_{max} mit $l_{max} = R_{\alpha}\, K_{max}$, bei dem die Werte ähnlich genug sind. Ein Problem der APW-Methode ist deren Abhängigkeit von der Energie E. Denn um eine Eigenfunktion exakt mit erweiterten ebenen Wellen zu beschreiben, muss der Wert für E gleich der Bandenergie $\epsilon_{\vec{k}}^n$ für den jeweiligen Zustand sein. Diese ist allerdings nicht bekannt mit der Folge, dass mit einem genäherten Wert für E gestartet wird und mit den daraus erstellten APW's werden die Hamilton Matrix und die Überlappmatrix erstellt und damit die Säkulargleichung bestimmt. $\epsilon_{\vec{k}}^n$ soll nun eine Wurzel der Gleichung sein. Ist dies nicht der Fall, nimmt man einen neuen Näherungswert für E und wiederholt diesen Vorgang so lange, bis eine Wurzel gefunden wird. Dieser Vorgang ist sehr zeitaufwendig und deshalb hat die APW-Methode in der Praxis kaum noch eine Bedeutung [19].

LAPW-Methode

Um den Nachteil der ursprünglichen Formulierung der APW-Methode zu beheben werden die Energien nahe einem Wert E_0 über eine Taylorreihenentwicklung bestimmt. Durch diesen Ansatz werden einerseits die Ableitungen von $u_l^{\alpha}(r, E_0)$ mit den zugehörigen Koeffizienten in den Basissatz mit aufgenommen, welche bestimmt werden müssen. Andererseits erhält man einen kleinen Fehler in der Form von $(\epsilon - E_0)^2$, durch den unbekannten Unterschied der Energie E_0 und der Eigenenergie ϵ des jeweiligen Zustands. In der LAPW-Methode wählt man innerhalb der Muffin-tin Kugel eine Linearkombination der nun bekannten Radialfunktionen $u_l^{\alpha}(r, E_0)$ und deren Ableitungen bezüglich der Energie. Der Basissatz ist somit wie folgt definiert:

$$\Phi_{\vec{G}}^{\vec{k}}(\vec{r},E) = \begin{cases} \frac{1}{\sqrt{\Omega}} e^{i(\vec{k}+\vec{G})\vec{r}} & \vec{r}\epsilon I \\ \sum_{l,m}(A_{lm}^{\alpha}(\vec{k}+\vec{G})u_l^{\alpha}(r,E_0) + B_{lm}^{\alpha}(\vec{k}+\vec{G})\dot{u}_l^{\alpha}(r,E_0))Y_m^l(\hat{r}) & \vec{r}\epsilon S_\alpha \end{cases} \tag{2.16}$$

Um nun die Koeffizientnen $A_{lm}^{\alpha}(\vec{k}+\vec{G})$ und $B_{lm}^{\alpha}(\vec{k}+\vec{G})$ zu bestimmen, muss an der Oberfläche der MT-Kugel neben dem Wert für die Funktionen auch deren Ableitung, innerhalb und außerhalb der Kugel, identisch sein. Um den Fehlerterm $O(\epsilon - E_0)^2$ gering zu halten wählt man nicht nur einen Wert E_0 für alle Zustände, sondern für jeden l-Wert eine eigene Linearisierungsenergie.

Im Lauf der Zeit wurden noch weitere Verbesserungen beziehungsweise Erweiterungen bezüglich des verwendeten Basissatzes vorgeschlagen. Diese sollen hier nur kurz erwähnt werden. Zunächst kann man die Basis bis zur zweiten Ableitung erweitern. Dieser Ansatz wird SuperLAPW oder kurz SLAPW genannt [20]. Im Bereich der Atomkerne lautet der Basissatz wie folgt:

$$\Phi_{\vec{G}}^{\vec{k}}(\vec{r},E) = \sum_{l,m}(A_{lm}^{\alpha}u_l^{\alpha}(r,E_0) + B_{lm}^{\alpha}\dot{u}_l^{\alpha}(r,E_0) + C_{lm}^{\alpha}\ddot{u}_l^{\alpha}(r,E_0))Y_m^l(\hat{r}) \tag{2.17}$$

Eine weitere Verbesserung soll durch die Einführung von sogenannten "lokalen Orbitalen" erzielt werden [21]. Diese sollen den Basissatz erweitern und haben folgende Form:

$$\Phi^{LO}(\vec{r},E) = \sum_{l,m}(A_{lm}^{\alpha,LO}u_l^{\alpha}(r,E_0) + B_{lm}^{\alpha,LO}\dot{u}_l^{\alpha}(r,E_0) + C_{lm}^{\alpha,LO}u_l^{\alpha}(r,E_1))Y_m^l(\hat{r}) \tag{2.18}$$

Es wird also ein weiterer Energieparameter für $u_l^{\alpha}(r,E_1)$ eingeführt. Diese können mit einem LAPW-Basissatz kombiniert werden. Um die Effizienz weiter zu erhöhen wurde vorgeschlagen [22], die lokalen Orbitale mit dem APW-Ansatz zu kombinieren, diese Methode wird APW+lo genannt. Auch eine Kombination von LAPW+lo und APW+lo wird eingesetzt.

Die LAPW-Methode beziehungsweise die APW+lo Methode ist in den DFT-codes Wien2k[5] und exciting [3, 4] implementiert. Diese Codes werden in dieser Arbeit verwendet.

2.2 Der Formgedächtniseffekt

Der sogenannte Formgedächtniseffekt wurde 1953 zufällig von Schweißern bei Arbeiten an gebogenen Blechen in einem amerikanischen U-Bootwerk entdeckt

[23]. Durch die Erwärmung beim Schweißen wurden Bleche wieder vollkommen flach. Das Material hat sich also an seine ursprüngliche Form "erinnert". Der Werkstoff der Bleche war eine Nickel-Titaniumlegierung, welche aufgrund ihrer guten Korrosionseigenschaften eingesetzt wurde. Zwar haben chinesische Forscher bereits 1932 an Legierungen ein seltsames gummiartiges Verhalten festgestellt, wissenschaftlich untersucht wurde der Effekt aber erst nach 1953.

Voraussetzungen

Die grundlegende Voraussetzung für Materialien, die den Formgedächtniseffekt zeigen, ist eine martensitische Umwandlung, bei der sowohl die Hochtemperaturphase als auch die Phase bei niedrigeren Temperaturen eine geordnete Struktur aufweisen. In Anlehnung an Stahl wird die Hochtemperaturphase *Austenit* und die Niedertemperaturphase *Martensit* genannt. Kühlt man nun einen Kristall im Bereich der Austenitphase ab, erfährt der Austenit ab der sogennannten Martensitstarttemperatur M_S eine Scherung. Dieser Vorgang wird als martensitische Umwandlung bezeichnet. Bei weiterer Abkühlung bildet sich kontinuierlich mehr Martensit in Form von Platten, bis der gesamte Krsitall bei der Martensitendtemperatur M_F vollständig umgewandelt ist. Dieser Vorgang ist reversibel, das heißt, beim Aufheizen eines Kristalls in martensitischer Phase wandelt das System wieder vollständig in die Hochtemperaturphase um. Dabei sind jedoch die Start- und Endtemperaturen um einige Kelvin verschoben, was bedeutet, dass die Austenitstarttemperatur A_S ungleich der Martensitendtemperatur ist. Das Material zeigt somit ein Hystereseverhalten. Die treibende Kraft für die Umwandlung ist geringere freie Enthalpie des Martensits bei geringeren Temperaturen. Dem entgegen wirkt die elastische Verzerrungsenergie, die aber bei einer Formgedächtnislegierung sehr klein ist und durch Akkomodation der Martensitplatten selbst und durch Zwillings bzw. - Stapelfehlerbildung innerhalb der Platten abgebaut wird. Das heißt, es kommt zu keinen oder nur sehr geringen plastischen Verformungen oder Versetzungsbewegungen. Dieser Aspekt ist sehr wichtig für die Reversibilität der Phasenumwandlung.

2.2.1 Einwegeffekt

Der Einwegeffekt ist eine natürliche Eigenschaft eines Formgedächtniskristalls [24]. Es ist jener Effekt, welcher 1953 zufällig entdeckt wurde. Als Ausgang hat man einen nicht verformten martensitischen Kristall. Dieser besteht wie vorher beschrieben aus Platten verzwillingten Martensits. In diesen Platten sind mehrere unterschiedliche Varianten des Martensits vorhanden. Belastet man nun den Kristall, beginnt dieser sich zunächst elastisch zu verformen bis die Verformungskurve ein Plateau erreicht. Ab dieser Dehnung gibt es keine Verfestigung mehr und der Kristall verformt sich nur durch Verschieben der Zwillingsgrenzen, wobei dadurch

die verschiedenen martensitischen Varianten reduziert werden bis nur noch eine vorhanden ist. Dem Martensitplateau folgt bei weiterer Dehnung ein weiterer elastischer Bereich und letztlich ein Bereich klassischer plastischer Verformung mit Versetzungsbildung und -bewegung, siehe Abbildung 2.2. Entlastet man das Material aus dem Plateaubereich bleibt der Kristall zunächst verformt. Wird der Werkstoff allerdings bis über die Austenitendtemperatur erhitzt, kehrt der Kristall in seine ursprüngliche Form zurück. Daher werden Verformungen im Plateaubereich auch pseudoplastische Verformungen genannt [25]. Bei der Umwandlung zurück in die Niedertemperaturphase bleibt die Form unverändert. Daher auch der Name Einwegeffekt, da eine Formänderung nur beim Aufheizen auftritt. Dieser Zusammenhang ist in Abbildung 2.3 grafisch dargestellt. Die Verformung in dieser Abbildung entspricht einer Bewegung entlang des Spannungsplateaus in Abbildung 2.2.

2.2.2 Zweiwegeffekt

Das Formgedächtnismaterial "erinnert" sich also nicht an die Form, die durch die pseudoplastische Verformung entstanden ist, sondern an die äußere Form vor der Verformung. Jedoch kann das Material "lernen", sich auch an die Verformung zu "erinnern" [24]. Dies nennt man den Zweiwegeffekt, da auch bei der Abkühlung eine Formänderung auftritt. Man erhält den Effekt durch zyklische Beanspruchung des Materials. Ziel ist es, durch Verformung über den Plateaubereich hinaus eine bestimmte Versetzungsstruktur zu erhalten. Erwärmt man anschließend den Kristall, wird der reversible Anteil der Verformung rückgängig gemacht, wobei der Kristall nicht vollständig in seine ursprüngliche Form zurückkehrt. Kühlt man den Kristall jetzt allerdings ab, werden durch das vorhandene Spannungsfeld der Versetzungen nun bestimmte Martensitvarianten bevorzugt und der Kristall nimmt eine bestimmte Form an. Das Material "erinnert" sich nun an zwei Formen und durchläuft sowohl bei Aufheizung als auch bei Abkühlung eine Formänderung. Die erreichbare Dehnung ist jedoch wesentlich geringer als beim Einwegeffekt.

2.2.3 Superelastizität

Ein weiterer Effekt, den Formgedächtnislegierungen zeigen, ist die Superelastizität [25]. Dieser tritt bei erhöhten Temperaturen auf, und zwar über der Austenitendtemperatur A_F, jedoch noch unter der Temperatur M_d, über der eine spannungsinduzierte Martensitbildung energetisch nicht mehr möglich ist. Wenn man nun in diesem Temperaturbereich das Material Spannungen aussetzt, kann es zur eben genannten spannungsinduzierten Martensitbildung kommen. Da hier wieder alle Martensitvarianten gleichberechtigt sind, bildet sich wieder ein Plateau aus und es sind sehr große Dehnungen möglich. Lässt man die Spannung nach, ist der Martensit aufgrund der hohen Temperatur nicht mehr stabil und das

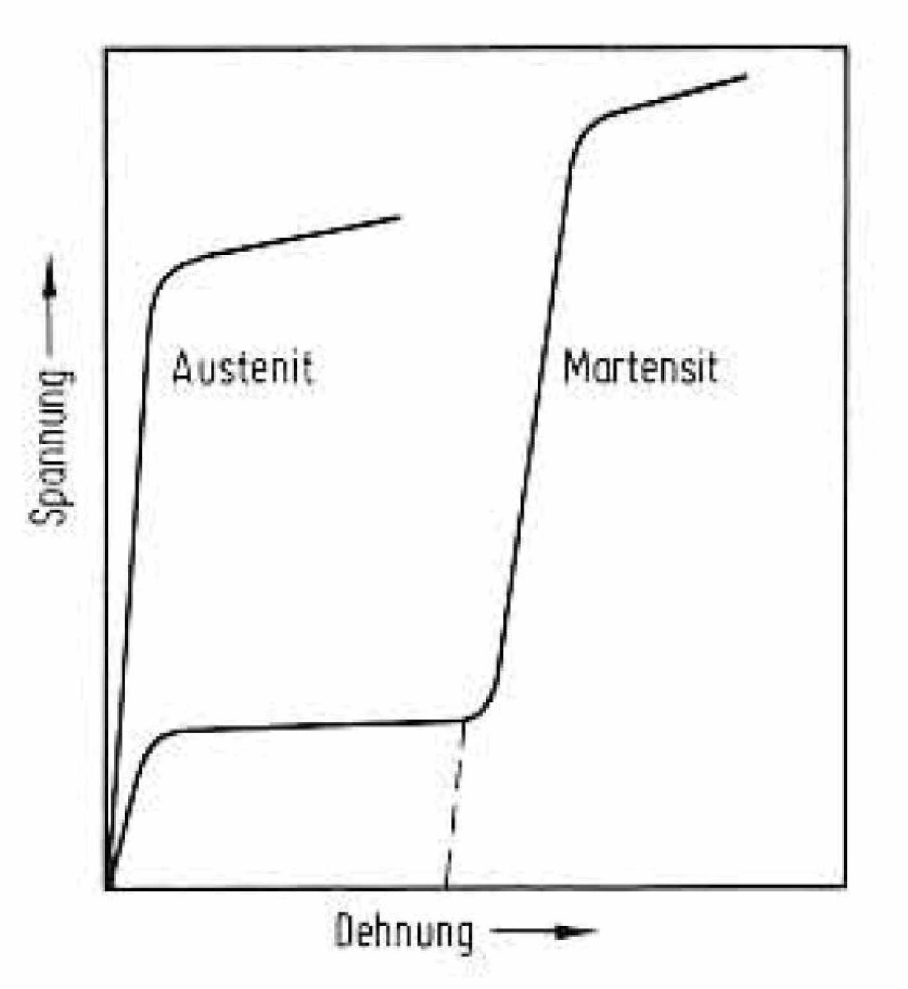

Abbildung 2.2: Spannungs-Dehnungskurve für die austenitische Phase und die martensitische Phase [24].

Material kehrt wieder über eine Hysterese in seine ursprüngliche Form zurück. Daher wird der Effekt auch Pseudoelastizität genannt.

2.3 Das NiTi-System

Legierungen aus Nickel und Titanium, auch Nitinol genannt, gehören zu den am meisten eingesetzten Formgedächtnislegierungen. Die technologisch interessanten Legierungen haben dabei eine stöchiometrische oder beinahe stöchiometrische Zusammensetzung. Abbildung 2.4 zeigt das Phasendiagramm des binären Systems. Sie haben neben dem Formgedächtniseffekt noch gute Festigkeitseigenschaften und eine hohe Korrosionsbeständigkeit.

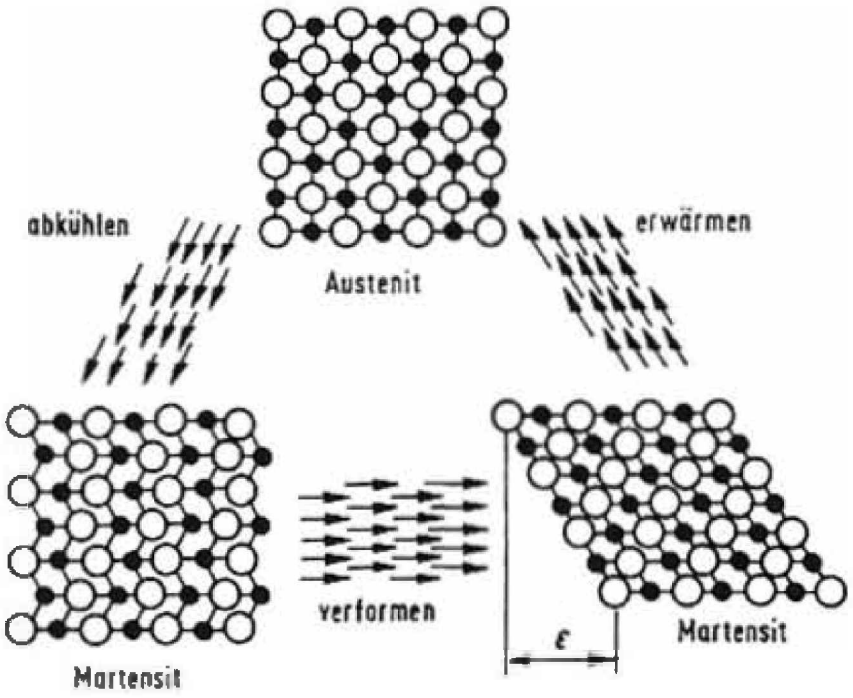

Abbildung 2.3: Schematische Darstellung des Kristalls bei den verschiedenen Zuständen während Durchlaufen eines Formgedächtniszyklusses [24].

Die Hochtemperaturphase ist eine B2 Phase und nur in einem sehr engen Zusammensetzungsbereich homogen vorhanden. Weicht die Zusammensetzung von der stöchiometrischen ab, so kommt es zu Entmischungen und Bildung von anderen Phasen. Die B2 Phase hat eine geordnete Struktur und ein kubisch raumzentriertes Gitter, wobei im Zentrum und an den Ecken der Einheitszelle ein unterschiedliches Atom sitzt. Die Martensitphase im Nitinol-System ist eine monokline $B19'$ Phase. Der Phasenübergang von der geordneten Austenitphase in die monokline B19' Phase ist einzigartig in diesem System. Bei vielen anderen Legierungen mit einer B2 Hochtemperaturphase kommt es beim Abkühlen zwar ebenfalls zu einem Phasenübergang, jedoch erfolgt dieser nur durch eine Änderung der Stapelfolge der (110) Ebenen [2] und die Niedertemperaturphase ist eine orthorhombische B19 Phase. Bei stöchiometrischen Nickel-Titan-Legierungen erfolgt eine zusätzliche Scherung und aus der orthorhombische Phase, wie sie bei anderen Legierungen gefunden wird, wird eine monokline, welche wie beschrieben eine wichtige Voraussetzung für Formgedächtnislegierungen sind.

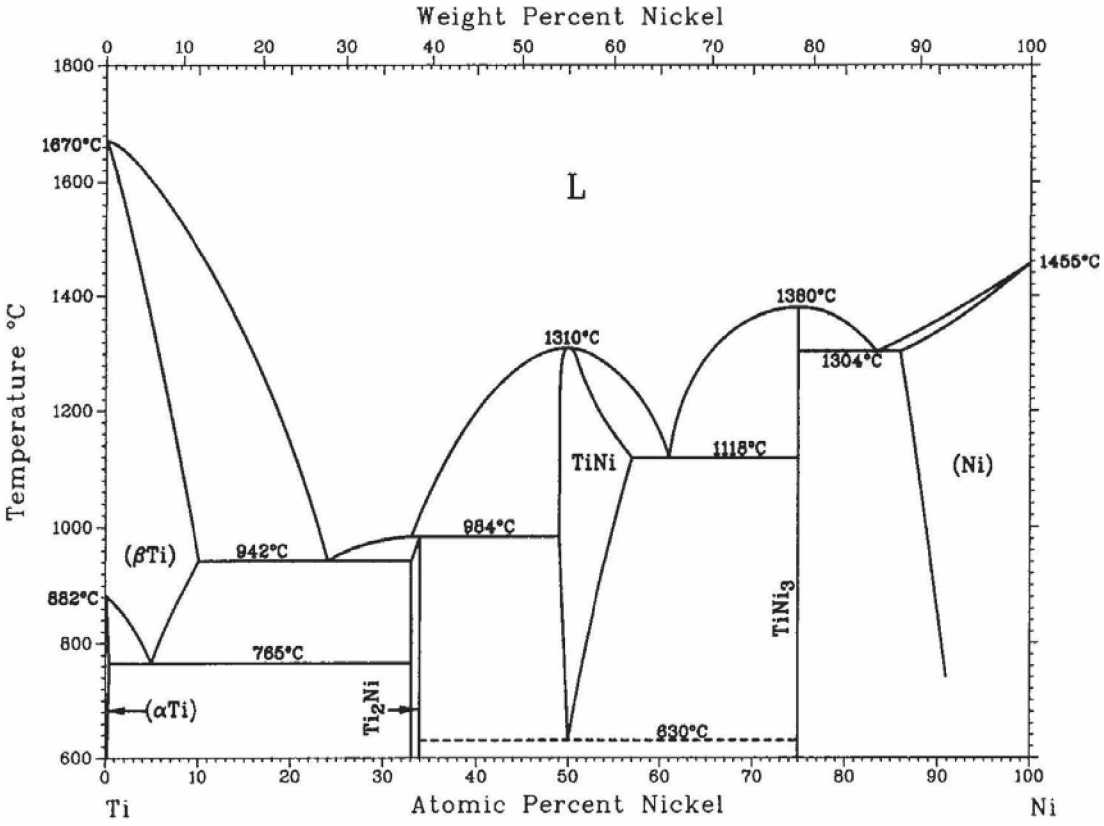

Abbildung 2.4: Binäres Phasendiagramm von Nickel-Titanium nach Nash [26]. Der Bereich TiNi ist jener mit B2-Struktur

2.4 Elastische Konstanten

2.4.1 Spannungs-Dehnungsbeziehungen

Verformt man einen Kristall mit geringen Dehnungen, verhält dieser sich vollkommen elastisch. Das heißt, wenn die Belastung weggenommen wird, kehrt der Krstall in seine ursprüngliche Form zurück. Oft besteht ein linearer Zusammenhang zwischen der aufgebrachten Spannung und der daraus folgenden Dehnung. Diesen Zusammenhang beschreibt das Gesetz von Hooke

$$\sigma_{i,j} = c_{i,j,k,l}\delta_{k,l}, \tag{2.19}$$

wobei $\sigma_{i,j}$ dem Spannungstensor entspricht, $\delta_{k,l}$ dem Dehnungstensor und $c_{i,j,k,l}$ dem sogenannten Tensor der linearen elastischen Konstanten. Dieser ist ein Tensor vierter Stufe und enthält 81 Elemente. Man kann diesen Zusammenhang auch in Matrix - oder Voigt'scher Schreibweise anschreiben. Durch die Symmetrie des

Tabelle 2.1: Die verschiedenen Bravaisgitter und ihre unabhängigen elastischen Konstanten. Aus dem Elastic-Programm [27].

Kristallstruktur	Unabhängige elastische Konstanten $C_{i,j}$
kubisch	C_{11},C_{12},C_{44}
hexagonal	C_{11},C_{12},C_{13},C_{33},C_{44}
trigonal I	C_{11},C_{12},C_{13},C_{14},C_{33},C_{44}
trigonal II	C_{11},C_{12},C_{13},C_{14},C_{15},C_{33},C_{44}
tetragonal I	C_{11},C_{12},C_{13},C_{33},C_{44},C_{66}
tetragonal II	C_{11},C_{12},C_{13},C_{16},C_{33},C_{44},C_{66}
orthorhombisch	C_{11},C_{12},C_{13},C_{22},C_{33},C_{44},C_{55},C_{66}
monoklin	C_{11},C_{12},C_{13},C_{15},C_{22},C_{23},C_{25},C_{33},C_{35},C_{44},C_{55},C_{66}
triklin	C_{11},C_{12},C_{13},C_{14},C_{15},C_{16},C_{22},C_{23},C_{24},C_{25},C_{26},C_{33},C_{34},C_{35},C_{36}, C_{44},C_{45},C_{46}, C_{55},C_{56},C_{66}

Spannungstensors und des Dehnungstensors werden diese zu 6 dimensionalen Vektoren und das Hooke'sche Gesetz kann folgendermaßen angeschrieben werden.

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36} \\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{46} \\ C_{51} & C_{52} & C_{53} & C_{54} & C_{55} & C_{56} \\ C_{61} & C_{62} & C_{63} & C_{64} & C_{65} & C_{66} \end{bmatrix} \times \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \\ \delta_5 \\ \delta_6 \end{bmatrix} \tag{2.20}$$

oder kurz

$$\sigma_i = c_{i,j}\delta_j \tag{2.21}$$

Da $c_{i,j}$ eine symmetrische Matrix ist, gibt es nun je nach Symmetrie der Kristallstruktur maximal 21 unabhängige elastische Konstanten. Wieviele und welche unabhängige elastische Konstanten es für die verschiedenen Kristallsysteme gibt, ist in Tabelle 2.1 dargestellt.

2.4.2 Berechnung von elastischen Konstanten

Zur Berechnung von elastischen Konstanten mithilfe eines DFT-Codes braucht man nun einen Zusammenhang zwischen der Energie des Systems und den Konstanten. Die Änderung der Dehnungsenergie in einem elastisch verformten, homogenen Kristall ist gegeben durch [28]:

$$dE = \sigma_{i,j} d\delta_{i,j} \tag{2.22}$$

Durch Gleichsetzen von Gleichung 2.22 und Gleichung 2.21 und durch weitere Umformung erhält man den Tensor der elastischen Konstanten durch zweimalige Ableitung der Energie,

$$c_{i,j,k,l} = \frac{1}{V} \frac{\partial^2 E}{\partial \delta_{i,j} \partial \delta_{k,l}} \tag{2.23}$$

wobei V dem Volumen des Systems entspricht. Die Energie eines Kristalls kann unter elastischen Dehnungen durch eine Taylor-Reihenentwicklung der Energie bezüglich des Dehungstensor ausgedrückt werden [29]:

$$E(V) = E(V_0, 0) + V_0 \sum_{i,j} \sigma_{i,j} \delta_{i,j} + \frac{V_0}{2!} \sum_{i,j,k,l} c_{i,j,k,l} \delta_{i,j} \delta_{k,l} + ... \tag{2.24}$$

In dieser Gleichung ist V_0 das Volumen des ungedehnten Kristalls, und $E(V_0, 0)$ die dazugehörige Energie beim Gleichgewichtsvolumen V_0. Schreibt man Gleichung 2.24 in Voigt'scher Notation ergibt sich

$$E(V) = E(V_0, 0) + V_0 \sum_{i} \sigma_i \delta_i + \frac{V_0}{2} \sum_{i,j} c_{i,j} \delta_i \delta_j + ... \tag{2.25}$$

Somit hat man einen Zusammenhang zwischen der Energie eines Systems und den elastischen Konstanten. Um nun für eine bestimmte Dehnungsbeanspruchung die elastischen Konstanten zweiter Ordnung zu bestimmen braucht man den Koeffizienten der zweiten Ordnung eines Polynoms, welches die Kurve Energiedichte der Dehnung über der Dehnung selbst fittet. Um alle unabhängigen Konstanten für ein bestimmtes Kristallsystem zu berechnen, muss man den Kristall für jede Konstante einer eigenen unabhängigen Deformation aussetzen. Welcher Art diese Deformationen sind wird bei den einzelnen, untersuchten Kristallstrukturen in Kapitel 4 angegeben.

2.4.3 Elastische Konstanten und martensitische Phasenumwandlung

Nun ist noch zu klären welchen Einfluss die elastischen Konstanten auf die martensitische Phasenumwandlung und damit den Formgedächtniseffekt haben. Zu diesem Zweck wird die Strukturumwandlung im Nickel-Titanium-System betrachtet. Bei einer solchen Phasenumwandlung spielt Diffusion keine Rolle [30]. Die Änderung der Kristallstruktur kommt nur durch Umklappen des Gitters zustande. Wie bereits in den vorhergehenden Kapiteln erwähnt, kommt es bei dieser Phasenumwandlung zu einer elastischen Deformation des Gitters, da neben, zwar

nur geringen, Volumsänderungen auch eine Scherung des Gitters erfolgt. Hier spielen die elastischen Konstanten eine Rolle, da Energie überwunden werden muss, um diese Verfomung zu ermöglichen, und die elastischen Konstanten auf gewisse Weise ein Maß für diese Verformungsenergie ergeben. Je weicher die elastischen Konstanten bezüglich dieser Scherung sind, desto geringer ist die gegenwirkende Energie, und die Phasenumwandlung kann zum Beispiel bereits bei höheren Temperatuen stattfinden.

Kapitel 3

Zellen-Optimierungsprogramm

Um für weitere Berechnungen verwendet zu werden, muss für eine Kristallstruktur zunächst die Einheitszelle des Kristalls im ungedehnten Zustand gefunden werden. Zu diesem Zweck wurde im Rahmen dieser Diplomarbeit ein Programm geschrieben, welches es ermöglicht, so eine Optimierung mit Hilfe des exciting Codes durchzuführen. Die Startdaten für eine solche Optimierung können aus verschiedenen Quellen kommen, wie zum Beispiel aus Experimenten oder auch aus diversen Berechnungen, wie einige der hier verwendeten Strukturen aus vorangegangen Simulationen von Dr. Jürgen Spitaler mit Hilfe der sogenannten „cluster expansion“ Methode stammen. Da dabei im NiTi - und NiTiHf-System die verschiedensten Kristallsysteme auftreten, soll das Programm möglichst allgemein anwendbar sein. Je nach Gitterstruktur muss die Einheitszelle nach verschiedenen Parametern optimiert werden. So muss eine kubische Einheitszelle aufgrund der hohen Symmetrie nur bezüglich ihres Volumens optimiert werden. Mit sinkender Symmetrie müssen immer mehr Parameter optimiert werden. Welche Parameter für eine bestimmte Kristallstruktur zu optimieren sind, wird in Tabelle 3.1 dargestellt.

Je komplexer die Form der Einheitszelle ist, umso mehr Parameter sind also zu behandeln. Wenn man nun zum Beispiel eine monokline Einheitszelle optimieren will muss man bei 5 Punkten pro Parameter $5^4 = 625$ Berechnungen, bei einem triklinen System sogar $5^6 = 15625$ - oder bei nur 3 Punkten $3^6 = 729$ Berechnungen, durchführen. Diese sind anschließend mit einem 4 – beziehungsweise 6- dimensionalen Polynom zu fitten, und daraus kann dann die optimierte Struktur gefunden werden. Dies ist aber mit einem sehr hohen Rechenaufwand verbunden und im Rahmen einer Diplomarbeit kaum durchzuführen. Aus diesem Grund geht das hier entwickelte Optimierungsprogramm einen anderen Weg. Es werden nacheinander je ein Parameter optimiert, im monoklinen Fall wären dies zuerst eine Volums- gefolgt von einer b/a- , c/a- und einer Winkeloptimierung.

Tabelle 3.1: Die 7 Kristallklassen und die zu optimierenden Parameter

	Kristallstruktur	Zu optimierende Parameter
1	kubisch	Volumen
2	hexagonal	Volumen, c/a Verhältnis
3	trigonal	Volumen, c/a Verhältnis
4	tetragonal	Volumen, c/a Verhältnis
5	orthorhombisch	Volumen, b/a Verhältnis, c/a Verhältnis
6	monoklin	Volumen, b/a Verhältnis, c/a Verhältnis, Gamma-Winkel
7	triklin	Volumen, b/a Verhältnis, c/a Verhältnis, Alpha-Winkel, Beta-Winkel, Gamma-Winkel

Dabei wird in jeder darauffolgenden Optimierung das Ergebnis der vorigen als Input verwendet. Nach Beendigung eines solchen Zyklus beginnt man wieder mit dem Volumen und führt so eine beliebige Anzahl an Zyklen durch. Der Vorteil dieser Methode liegt darin, dass bei 5 Punkten pro Parameter bei zum Beispiel 6 Zyklen für den monoklinen Fall nur 5*4*6 = 120 und für den triklinen Fall 5*6*6 = 150 Berechnungen benötigt werden. Ein Nachteil ist allerdings, dass man immer auf das Ergebnis des vorhergegangen Optimierungsschrittes warten muss, und so bei fünf Punkten pro Parameter nur fünf Berechnungen parallel laufen können. Das Optimierungprogramm wurde in der Programmiersprache Python geschrieben und besteht aus drei Hauptbestandteilen. Diese drei Teile werden im folgenden kurz beschrieben.

3.1 Setup

Das Setup-Programm hat die Aufgabe aus jeder beliebigen Input-Datei des exciting Codes einen Verzeichnisbaum für die gewünschte Art der Optimierung und den angegebenen Dehnungen zu erstellen. Dazu wird dem Programm eine exciting Inputdatei übergeben und es erkennt aus dieser die Gitterstruktur. Je nach Gitterstruktur kann der Benutzer aus den in Tabelle 3.1 dargestellten zu optimierenden Parametern wählen, wobei es auch möglich ist eine gesamte Optimierung zu wählen. Bei einer gesamten Optimierung werden die Teilschritte automatisch nacheinander ausgeführt, bis eine gewünschte Anzahl an Zyklen erreicht ist oder sich die Parameter nicht mehr wesentlich ändern. Ist die Art der Optimierung ausgewählt, werden nur noch die maximale physikalische Dehnung beziehungweise die maximale Änderung der Winkel und die Anzahl der zu berechnenenden Punkte angegeben, und das Setupprogramm erstellt den Ordnerbaum mit den zugehörigen Inputdateien entsprechend den jeweiligen Dehnungen. Die Inputdateien werden dabei so erstellt, dass bei Optimierung eines einzelnen Parameters

die anderen nicht verändert werden. So bleiben zum Beispiel bei einer Volumsoptimierung das c/a- bzw. das b/a Verhätnis und jegliche Winkel der Einheitszelle konstant. Die Dehnung für einen bestimmten Punkt n wird dabei folgendermaßen berechnet:

$$\varepsilon_n = \frac{\varepsilon_{max} n}{n_{ges}} \tag{3.1}$$

Wobei n Werte zwischen $-\left(\frac{n_{ges}-1}{2}\right)$ und $+\left(\frac{n_{ges}-1}{2}\right)$ annimmt. ε_{max} ist die maximale Dehnung und n_{ges} die ungerade Gesamtzahl der Punkte.

Bei einer Volumsoptimierung wird die 3×3 Matrix der Einheitszelle

$$\begin{pmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{pmatrix}$$

mit folgender Deformationsmatrix multipliziert:

$$\begin{pmatrix} 1+\varepsilon & 0 & 0 \\ 0 & 1+\varepsilon & 0 \\ 0 & 0 & 1+\varepsilon \end{pmatrix}$$

Dadurch ist gewährleistet, dass sich die anderen Parameter, wie zum Beispiel das c/a-Verhältnis, nicht verändern. Bei Optimierung einer der anderen Parameter ist es nicht so einfach eine solche Deformationsmatrix aufzustellen, bei der sich keiner der anderen Parameter ändert. Daher wurde für diese Parameter ein anderer Weg genommen. Ausgegangen wird von der allgemeinen Formel für das Volumen einer Einheitszelle:

$$V = abc\sqrt{1 - cos^2(\alpha) - cos^2(\beta) - cos^2(\gamma) + 2cos(\alpha)cos(\beta)cos(\gamma)} \tag{3.2}$$

Es gilt nun die Längen der die Einheitszelle aufspannenden Vektoren $\vec{a}$, $\vec{b}$ und $\vec{c}$ so zu verändern, dass sich nur der ausgewählte Parameter ändert und die restlichen unverändert bleiben. Soll also zum Beispiel das b/a Verhältnis optimiert werden, darf sich weder das Volumen, das c/a Verhältnis noch einer der drei Winkel α, β oder γ ändern. Wobei α, β und γ die Winkel zwischen den die Einheitszelle aufspannenden Vektoren sind. Daher lauten die Bedingungen für diesen Fall , dass $V_{neu} = V_{alt}$, $(c/a)_{neu} = (c/a)_{alt}$, $\alpha_{neu} = \alpha_{alt}$, $\beta_{neu} = \beta_{alt}$ und $\gamma_{neu} = \gamma_{alt}$ sein sollen. Geändert werden, können dabei, wie gesagt, nur die Längen a, b und c der Gittervektoren. Dies soll mithilfe von Gleichung 3.2 geschehen, welche davor noch umgeformt wird.

$$V = a^3 \frac{b}{a} \frac{c}{a} \sqrt{1 - cos^2(\alpha) - cos^2(\beta) - cos^2(\gamma) + 2cos(\alpha)cos(\beta)cos(\gamma)} \tag{3.3}$$

Nun können altes und neues Volumen gleichgesetzt werden und aus den anderen vorher genannten Bedingungen ergibt sich für a_{neu} :

$$a_{neu} = \sqrt[3]{a_{alt}^3 \left(\frac{\left(\frac{b}{a}\right)_{alt}}{\left(\frac{b}{a}\right)_{neu}} \right)} \tag{3.4}$$

Dabei wird $\left(\frac{b}{a}\right)_{neu}$ für jeden Punkt über die jeweilige Dehnung ε berechnet wird:

$$\left(\frac{b}{a}\right)_{neu} = \left(\frac{b}{a}\right)_{alt} (1 + \varepsilon) \tag{3.5}$$

Die noch fehlenden Gittervektorenlängen b_{neu} beziehungsweise c_{neu} werden auf folgende Art ermittelt.

$$b_{neu} = a_{neu} \left(\frac{c}{a}\right)_{neu} \tag{3.6}$$

$$c_{neu} = c_{alt} \left(\frac{a_{neu}}{a_{alt}}\right) \tag{3.7}$$

Die Winkel α, β und γ bleiben in diesem Optimierungsfall unverändert

Will man das c/a Verhältnis optimieren, so ist in Gleichungen 3.4 und 3.5 ledeglich $\left(\frac{b}{a}\right)$ durch $\left(\frac{c}{a}\right)$ zu ersetzen. Bezüglich der neuen Werte von b und c ändern sich Gleichungen 3.6 und 3.7 zu:

$$b_{neu} = b_{alt} \left(\frac{a_{neu}}{a_{alt}}\right) \tag{3.8}$$

$$c_{neu} = a_{neu} \left(\frac{c}{a}\right)_{neu} \tag{3.9}$$

Auch hier ändern sich die Größen der Winkel zwischen den Vektoren nicht.

Zur Optimierung einer der drei Winkel bleiben jeweils die beiden anderen unverändert und der dritte ändert sich gemäß Gleichung 3.5. Da jetzt unterschiedliche Winkel im Spiel sind, wird die neue Länge des Vektors $\overrightarrow{a}$ folgendermaßen berechnet.

$$a_{neu} = \sqrt[6]{a_a^6 \left(\frac{(1 - cos^2\alpha_a - cos^2\beta_a - cos^2\gamma_a + 2cos\alpha_a cos\beta_a cos\gamma_a)}{(1 - cos^2\alpha_n - cos^2\beta_n - cos^2\gamma_n + 2cos\alpha_n cos\beta_n cos\gamma_n)} \right)} \tag{3.10}$$

In Gleichung 3.10 steht der Index a für alt und Index n für neu. Zuletzt fehlen noch die Formeln zur Berechnung der neuen Werte von b und c bei einer Winkeloptimierung.

$$b_{neu} = b_{alt} \left(\frac{a_{neu}}{a_{alt}} \right) \tag{3.11}$$

$$c_{neu} = c_{alt} \left(\frac{a_{neu}}{a_{alt}} \right) \tag{3.12}$$

Des Weiteren wird eine Info-Datei erstellt in der alle für die späteren Programmteile wichtigen Informationen wie die maximale Dehnung, die Anzahl der Punkte oder die Art der Optimierung gespeichert werden.

3.2 Ausführung

Der zweite Teil des Programms hat die Aufgabe, die im ersten Teil vorbereiteteten Inputdateien in den jeweiligen Ordnern einer exciting-Berechnung zu unterziehen, wobei die einzelnen Berechnungen seriell oder parallel gestartet werden können. Dabei ist jedoch zu beachten, dass immer nur ein Parameter parallel optimiert werden kann. Wenn eine Gesamtoptimierung gestartet wird, muss nach jeder Berechnung der Energien der einzelnen Punkte ein Fit durchgeführt werden, um daraus die minimale Energie und die dazugehörige Einheitszelle zu bestimmen. Aus diesem Grund ist in den Ausführungsteil für diese Fälle auch der Auswertungsteil eingebaut, der im folgenden Kapitel genauer beschrieben wird. Für die einzelnen Inputdateien erhält man also während der Ausführung der Rechnungen die jeweiligen exciting-Outputdateien. Als Ergebnis dieses Teils des Programms hat man also einen Ordnerbaum, in dem sich die Resultate der SCF-Zyklen der

jeweilig im Setup-Teil bestimmten Dehnungen für die zu optimierende Struktur befinden. Aus diesen Daten kann nun das Gesamtergebnis des Programms mithilfe des letzten Teils bestimmt werden.

3.3 Auswertung

Der letzte Teil des Programms ist verantwortlich für die Resultate. Hierbei ist das Resultat des Programms wieder eine exciting-Inputdatei mit optimierter Einheitszelle, also jene mit der niedrigsten Gesamtenergie. Nach Beendigung des Ausführungsschritts hat man für jeden Dehnungspunkt einen Ordner, in dem sich die Output-Daten einer abgeschlossen exciting Grundzustansberechnung befinden. Um nun die neue Input-Datei mit der niedrigsten Energie zu erhalten, werden die Energien der einzelnen Dehnungspunkte eingelesen, und daraus das Minimum bestimmt. Die konvergierten Gesamtenergien lassen sich aus den jeweiligen TOTENERGY.OUT Dateien lesen, in denen diese gespeichert sind, und die Volumina aus den dazugehörigen INFO.OUT Dateien, in denen sämtliche Informationen der exciting-Berechnung gespeichert sind. Bei allen anderen zu optmimierenden Parametern wird die jeweilige Dehnung aus den, in der INFO-Datei gespeicherten Daten, maximale Dehnung und Gesamtanzahl der Punkte, berechnet. Diese Daten werden im Anschluss gefittet. Dabei nimmt die Volumsoptimierung eine Sonderstellung ein.

Birch-Murnaghan Fit

Bei einer Volumsoptimierung erhält man bekanntlich die Energie für verschiedene Volumina. Die Abhängigkeit der Energie vom Volumen wird durch eine Zustandsgleichung beschrieben. So macht es auch Sinn diese Punkte durch eine solche Gleichung zu fitten. Nebenbei erhält man auf diesem Weg bei einer Volumsoptimierung auch den Kompressionsmodul B, welcher sich als nützlicher Parameter heraustellt. Das einfachste Modell für so einen Fall wäre das eines harmonischen Festkörpers. Die dazugehörige Zustandsgleichung lautet:

$$E = E_0 + \frac{1}{2} B_0 \frac{(V - V_0)^2}{V_0} \tag{3.13}$$

Eine ausgereiftere Zustandsgleichung wurde von Francis Murnaghan 1944 veröffentlicht [31]. Es wird davon ausgegangen, dass die erste Ableitung des Kompressionsmoduls B' sich mit dem Druck nicht ändert also als konstant angenommen werden kann.

$$B' = \left(\frac{\partial B}{\partial P}\right)_P = B_0' \tag{3.14}$$

Somit ist der Kompressionsmodul abhängig vom Druck:

$$B = B_0 + B_0' P \tag{3.15}$$

Verknüpft man Gleichung 3.15 mit der Definition des Kompressionsmoduls:

$$B = -V \left(\frac{\partial P}{\partial V} \right)_T \tag{3.16}$$

erhält man nach Umformung und Integration den Druck in Abhängigkeit vom Volumen:

$$P = \frac{B_0}{B_0'} \left(\left(\frac{V}{V_0} \right)^{B_0'} - 1 \right) \tag{3.17}$$

Nun setzt man Gleichung 3.17 in Gleichung 3.18 ein, welche sich aus der Definition des Druckes ergbibt:

$$E = E_0 - \int P dV \tag{3.18}$$

und erhält letztendlich die Zustandsgleichung für die Energie nach Murnaghan:

$$E = E_0 + \frac{B_0 V}{B_0'} \left(\frac{(V/V_0)^{B_0'}}{B_0' - 1} + 1 \right) - \frac{B_0 V_0}{B_0' - 1} \tag{3.19}$$

Eine andere Möglichkeit der Zustandsgleichung wurde von Birch [32] 1947 vorgestellt. Dabei wird von dem Zusammenhang des Druckes p und der freien Energie F ausgegangen:

$$p = - \left(\frac{\partial F}{\partial V} \right)_T \tag{3.20}$$

Wobei die freie Energie eines Festkörpers von Birch als Reihenentwicklung bezüglich der Eulerschen Dehnungen mit druckabhängigen Konstanten aufgestellt wurde. Man erhält dadurch eine Funktion des Druckes in Abhängigkeit von den Parametern V_0, B_0 und B_0' sowie vom Volumen V. Durch Einsetzen dieser in Gleichung 3.18 erhält man die Zustandsgleichung für die Energie nach Birch-Murnaghan:

$$E = E_0 + \frac{9 B_0 V_0}{16} \left(\left(\frac{V}{V_0} \right)^{2/3} - 1 \right)^3 B_0' + \left(\left(\frac{V}{V_0} \right)^{2/3} - 1 \right)^2 \left(6 - 4 \left(\left(\frac{V}{V_0} \right)^{2/3} \right) \right) \tag{3.21}$$

Das Programm verwendet bei einer Volumsoptimierung nun die Zustandsgleichung nach Birch-Murnaghan als Fit-Polynom. Die Lösung dieses nicht linearen Fitproblems geschieht folgendermaßen: Das Programm ändert die Parameter E_0, V_0, B_0 und B_0' solange, bis das quadratische Mittel des Abstands zwischen Kurve und zuvor berechneten Daten ein Minimum erreicht. Damit diese Parameter allerdings richtig konvergiert werden, braucht man zuerst geeignete Startwerte für E_0, V_0 und B_0. Zu diesem Zweck wird vor dem Birch-Murnaghan-Fit ein quadratischer Fit durchgeführt und die Startwerte für die Minimierung des quadratischen Mittelwertes nach Gleichung 3.13 bestimmt. Für den vierten Parameter B_0' ist es ausreichend einen Startwert von Eins zu verwenden. Als Ergebnis erhält man letztendlich das gesuchte Gleichgewichtsvolumen V_0 für die Erstellung einer neuen Inputdatei, und nebenbei gibt das Programm den Kompressionsmodul B_0 in [GPa] aus.

Für alle anderen Optimierungen wird als Fitkurve ein Polynom vierter Ordnung benutzt und daraus die Dehnung mit der niedrigsten Energie bestimmt. Auch hier erstellt der Resultatteil eine neue Inputdatei mit der optimierten Einheitszelle, welche für weitere Optimierungen zur Verfügung steht. Das Ergebnis einer Volumsoptimierung mithilfe des Programms ist in Abbildung 3.1 als Beispiel dargestellt.

Nach Abschluss aller drei Programmteile wird wie erwähnt der nächste Parameter optimiert und der Vorgang wiederholt, bis sich die Form und Größe der Einheitszelle kaum mehr ändert.

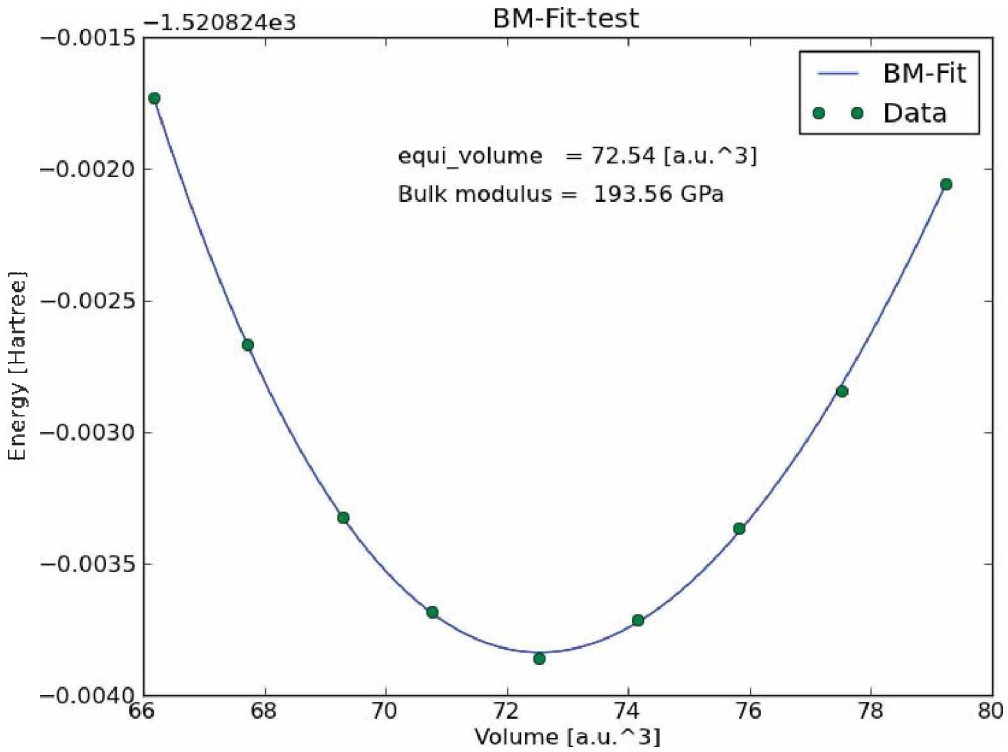

Abbildung 3.1: Ergebnis einer Volumsoptimierung mit dem Cell-Optimization-exciting Programm. Die Optimierung wurde an elementarem Nickel durchgeführt

Kapitel 4

Ergebnisse

4.1 Elemente Ni, Ti und Hf

Als erster Schritt der theoretischen Untersuchung der elastischen Konstanten im NiTi - bzw NiTiHf-System sollen die elementaren Festkörper von Nickel, Titanium und Hafnium untersucht werden. Das Programm zur Berechnung der elastischen Konstanten von beliebigen Kristallstrukturen heißt ElaStic [27]. Es erzeugt für jede unabhängige elastische Konstante in einem bestimmten Kristallsystem eine Serie von Berechnungen, wobei für die unterschiedlichen Verzerrungen für jede Auslenkung eine DFT-Rechnung ausgeführt wird. Als Ergebnis erhält man dabei die Energie über der jeweiligen Dehnung und kann aus den polynomialen Fits dieser Kurven alle unabhängigen elastischen Konstanten bestimmen. Die dahintersteckende Theorie ist in Kapitel 2.4 beschrieben. Durch die Verformung der Einheitszelle bei den verschiedenen Dehnungen kann es dazu kommen, dass die Atompositionen nicht mehr an der energetisch günstigsten Position liegt. Daraus folgt, dass Kräfte auf diese Atome wirken. Aus diesem Grund wurden die Simulationen mit relaxierenden Positionen gerechnet. Das heißt, nach Abschluss einer konvergierten SCF-Rechnung werden bei vorhandenen Kräften die Atompositionen geändert, und die Berechnung wiederholt, bis die Kräfte kleiner einem eingegebenen Kriterium sind.

Um die Verlässlichkeit der berechneten elastischen Konstanten sicherzustellen, werden zunächst Konvergenztests bezüglich der Größe von RK_{max}, der Anzahl der k-Punkte in der ersten Brillouinzone und des smearing-Parameters durchgeführt. Der Parameter RK_{max} beschreibt die Größe des Basissatzes. Er ist das Produkt aus dem kleinsten "muffin-tin"Radius $R_{MT,min}$ und dem "Cutoff"für die ebenen Wellen K_{max} (Kapitel 2.1.1.) . Zunächst wird ein k-Punkt Konvergenztest für verschiedene Werte von σ gemacht, da diese voneinander abhängig sind. Dabei kann jede dieser Kurven auf einen anderen Wert konvergieren. Bei diesen Tests ist RK_{max} immer gleich groß. Im Anschluss daran wird mit den zuvor ermittelten

Werten, ein Konvergenztest bezüglich RK_{max} durchgeführt. Als Zielgröße für die Konvergenztests wird der Kompressionsmodul B [GPa] gewählt, da es einen Zusammenhang zwischen den elastischen Konstanten und dem Kompressionsmodul gibt. So hat Voigt folgende Gleichung zur Berechnung von B aus verschiedenen elastischen Konstanten aufgestellt[33].

$$B = \frac{1}{9}(C_{11} + C_{22} + C_{33} + 2(C_{12} + C_{13} + C_{23})) \tag{4.1}$$

Der Kompressionsmodul, in Gleichung 3.16 definiert, gibt die Änderung des Volumen in Abhängigkeit vom auftretenden Druck an. Nach Abschluss der Konvergenztests werden, wenn nötig, die Einheitszellen der einzelnen Elemente optimiert und als letzter Schritt ihre elastischen Konstanten zweiter Ordnung berechnet. Für die Optimierung wurde das für diese Diplomarbeit geschriebene *Cell – Optimization – exciting* Programm verwendet.

4.1.1 Nickel

Nickel ist das Element mit der Ordnungszahl 28. Seine Kristallstruktur ist kubisch flächenzentriert und seine Raumgruppe hat die Nummer 225 mit dem Namen $Fm3m$. Die primitive Einheitszelle des Gitters ist in Abbildung 4.1 dargestellt. Da das Gitter somit eine sehr hohe Symmetrie besitzt, muss nur das Volumen der Elementarzelle optimiert werden. Dies geschieht im Rahmen der Konvergenztests automatisch, da B_0 als Konvergenzparameter herangezogen wird, ist eine weitere Optimierung nicht vonnöten. Als Ausgangswert für die Länge eines Gittervektors a wird der experimentelle Wert von 6.65 [a.u.] verwendet [34].

Konvergenztests

Für die Konvergenztests von Nickel wurde als Austauschkorrelationspotential die Näherung nach Perdew, Burke und Ernzerhof [14] verwendet, somit eine GG-Näherung. Zuerst wurde die Anzahl der k-Punkte und der Smeringparameter σ variiert. Diese waren $\sigma_1 = 0.01$ [Ha], $\sigma_2 = 0.005$ [Ha] und $\sigma_3 = 0.001$ [Ha]. Für die Größe des k-Punktnetztes wurden als Input jeweils $8 \times 8 \times 8$, $12 \times 12 \times 12$, $16 \times 16 \times 16$, $20 \times 20 \times 20$, $24 \times 24 \times 24$ und $28 \times 28 \times 28$ verwendet. Bei allen diesen Berechnungen wurde derselbe Wert für $RK_{max} = 7.0$ benutzt. Die Tests wurden mithilfe einer Volumsoptimierung des im letzten Kapitels beschriebenen Optimierungsprogrammes durchgeführt. Für die maximale physikalische Dehnung wurde immer 0.04 verwendet mir einer Schrittweite von 0.01. Die Ergebnisse der Konvergenztests sind in den folgenden Abbildungen 4.2 und 4.3 dargestellt.

In Abbildung 4.2 ist der Kompresionsmodul über der Gesamtanzahl der k-Punkte in der durch die Symmetrie irrereduziblen Brillouinzone (IBZ) dargestellt. Dabei

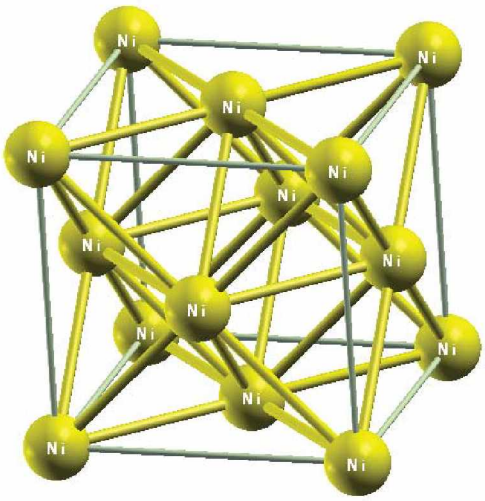

Abbildung 4.1: Elementarzelle des elementaren Nickel. Abbildung erstellt mithilfe von XCrySDen [35].

ist ersichtlich, dass sich bei σ_1 ab einer Anzahl von 250 Punkten der Kompressionsmodul nur mehr um weniger als 1 Prozent ändert. Des Weiteren unterscheiden sich die Ergebnisse für die verschiedenen σ kaum. 256 k-Punkte in der IBZ entsprechen dabei einem k-Punktnetz von $20 \times 20 \times 20$ für das k-Punktnetz. Dieser Input wurde für die weiteren Berechnungen mit Nickel, also für den Konvergenztest von RK_{max} und die Berechnung der elastischen Konstanten, verwendet. Für die Konvergenztests von RK_{max} wurde der Kompressionsmodul für Werte von 6.0 bis 9.0, in 0.5 Schrittgröße berechnet.

Aus Abbildung 4.3 ist ersichtlich, dass der Kompressionsmodul ab einem Wert von 8.0 für RK_{max} konvergiert ist. Für die weiteren Berechnungen wurde dabei 8.5 verwendet. Bei der Verwendung der konvergierten Parameter ergibt sich für das Gleichgewichtsvolumen ein Wert von 72.54 $[a.u.^3]$. Dies bedeutet, das der Gitterparameter $a = 6.62$ [a.u.] beträgt. Es sei darauf hingewiesen, dass es sich bei dem Volumen um jenes der primitiven Einheitszelle handelt.

Elastische Konstanten

Für die Berechnung der elastischen Konstanten wurde eine Inputdatei mit dem in den Konvergenztests optimierten Wert für den Gitterparameter a benutzt.

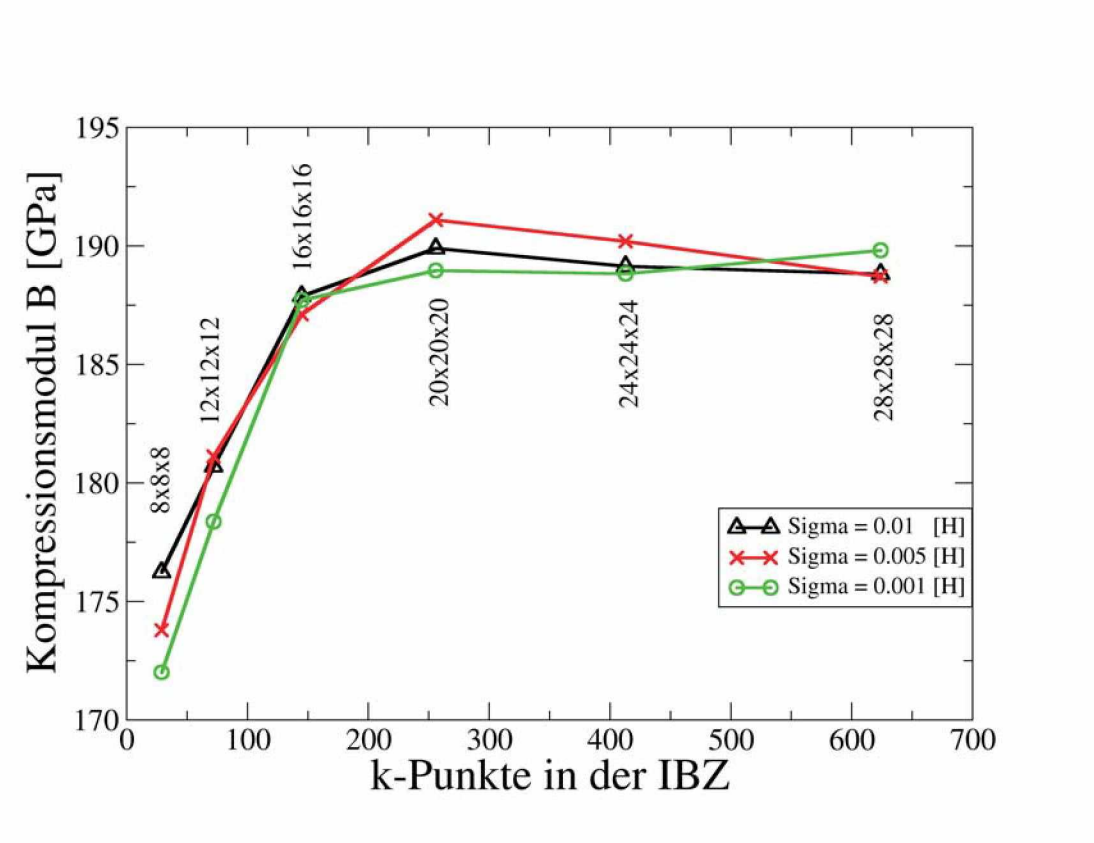

Abbildung 4.2: Ergebnis der Konvergenztests für verschiedene Smearingparameter σ bezüglich der Anzahl der k-Punkte in der irreduziblen Brilloin-Zone für Nickel.

Als Austauschkorrelationpotential wurde wiederum PBE-GGA en die Ergebnisse der Konvergenztets verwendet. Die elastischen Konstanten wedgewählt [14]. Für RK_{max}, Anzahl der k-Punkte und σ werden wie bereits erwähnt mit dem ElaStic Programm berechnet. Hierfür braucht man, bei Verendung des exciting Codes nur noch die entsprechende Inputdatei. Des Weiteren benötitgt das Elastic-Programm als Eingabe noch die maximale Lagrange'sche Dehnung und die Anzahl zu berechnender Dehnungen. Diese Eingabewerte für Nickel waren eine maximale Dehnung von 0.05 bei 21 Punkten. Bei kubischer Struktur gibt es 3 unabhängige elastische Konstanten und die Matrix nimmt folgende Form an:

$$C_{i,j} = \begin{bmatrix} C_{11} & C_{12} & C_{12} & 0 & 0 & 0 \\ C_{12} & C_{11} & C_{12} & 0 & 0 & 0 \\ C_{12} & C_{12} & C_{11} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{44} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{44} \end{bmatrix}$$

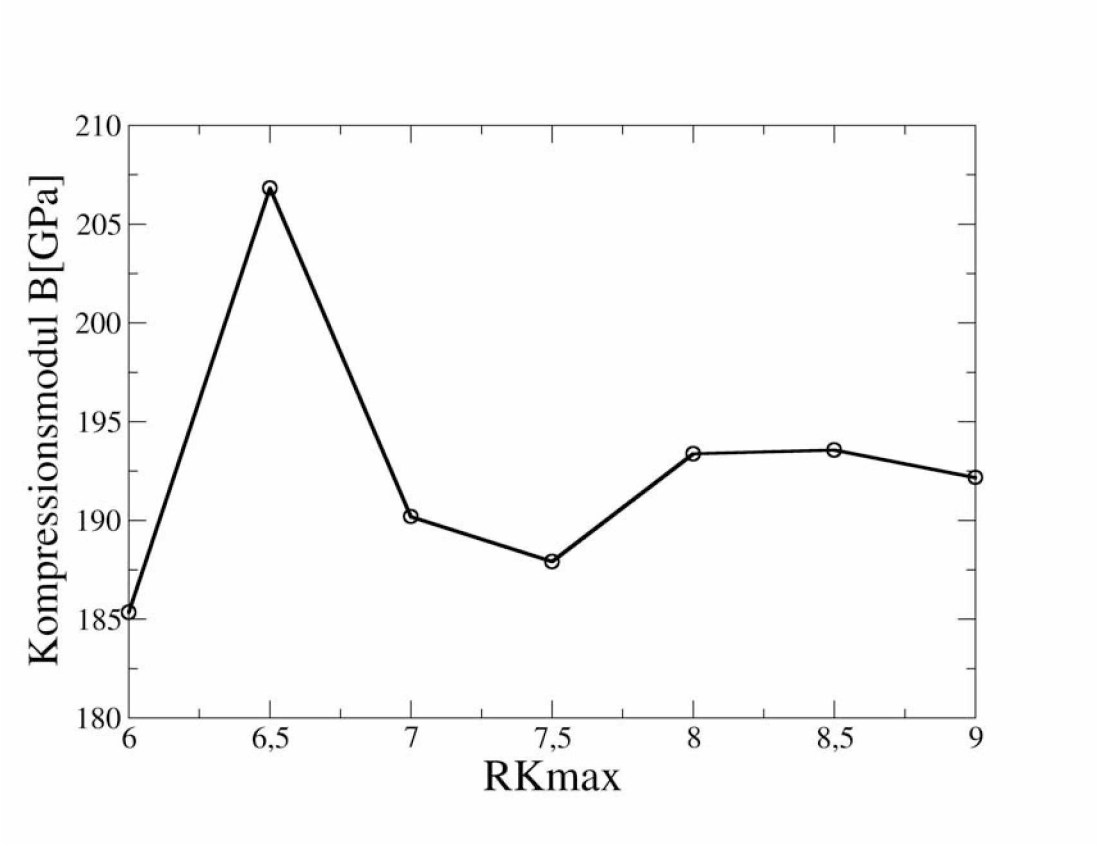

Abbildung 4.3: Ergebnis der Konvergenztests des Kompressionsmoduls bezüglich RK_{max} für Nickel.

Die drei unterschiedlichen Dehnungsarten, die für den kubischen Fall angewendet werden, sind eine gleichmäßige Dehnung in x, y und z Richtung, wobei für eine bestimmte Lagrange'sche Dehnung η der Dehnungsvektor folgende Form animmt (η, η, η, 0, 0, 0), eine ebene Dehnung in x- und y-Richtung mit dem Dehnungsvektor (η, η, 0, 0, 0, 0) und eine Scherung in drei Richtungen mit (0, 0, 0, 2η, 2η, 2η). Nach der Berechnung der Energien in Abhängigkeit von den jeweiligen Dehnungen müssen zur Bestimmung der elastischen Konstanten diese Kurven noch von einem geeigneten Polynom gefittet werden. Zu diesem Zweck gibt das Programm Kurven der zweiten Ableitung der Energie nach der Dehnung über der maximal verwendeten Dehnung für einen Fit zweiter, vierter beziehungsweise sechster Ordnung aus, wobei immer nur eine bestimmte Anzahl an zuvor berechneten Punkten durch die beschränkte maximale Lagrange'sche Dehnung verwendet wird. Ein solches Ergebnis für die gleichmäßige allseitige Dehnung wird beispielhaft in Abbildung 4.4 gezeigt. Um ein konvergiertes Ergebnis zu erhalten wählt man nun ein bestimmtes Polynom und eine zugehörige maximale Dehnung, an der diese Kurve ein Plateau zeigt. Für das gezeigte Besispiel wurde

ein Polynom 4ter Ordnung mit einer maximalen Dehnung von 0.045 verwendet. Das sich daraus ergebende Resultat und somit die elastischen Konstanten von Nickel lauten.

$$C_{i,j} = \begin{bmatrix} 250.0 & 164.6 & 164.6 & 0 & 0 & 0 \\ 164.6 & 250.0 & 164.6 & 0 & 0 & 0 \\ 164.6 & 164.6 & 250.0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 114.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 114.3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 114.3 \end{bmatrix} \text{[GPa]}$$

Berechnet man den Kompressionsmodul nach Gleichung 4.1, kommt man auf ein Ergebnis von 193.1 [GPa]. Vergleicht man dieses Ergebnis mit jenen aus den Konvergenztests, so liegt eine hervorragende Übereinstimmung vor. Der Unterschied ist kleiner als 1 Prozent.

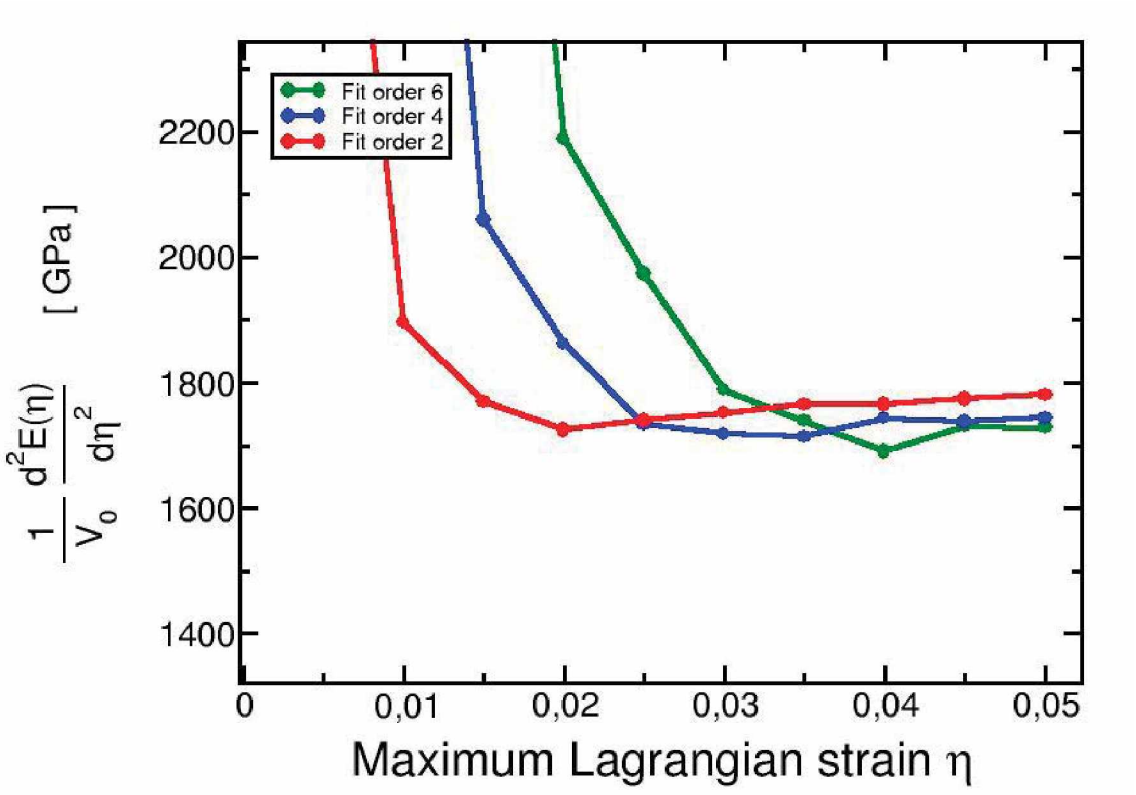

Abbildung 4.4: Analyse der Ergebnisse der Berechnung der Energie in Abhängigkeit von der Dehnung für eine bestimmte Deformation bei der Berechnung der elastischen Konstanten. Die Analyse dient dazu, einen geeigneten Wert für die maximale Dehnung und die Ordnung des Fit-Polynoms zu finden.

4.1.2 Titanium

Titanium ist das Element Nummer 22 im Periodensystem. Die Kristallstruktur von Titanium ist durch eine hexagonal dichtest gepackte Struktur charakterisiert und hat die Raumgruppe mit der Nummer 194. Das bedeutet, dass bei der Optimierung der Einheitszelle zwei Parameter berücksichtigt werden müssen, siehe Kapitel 3 . Auch bei Titanium wurden wieder experimentelle Daten als Startwert verwendet. Dies sind hier der Gitterparameter a mit 5.58 [a.u.] und das Verhältnis von $c/a = 1,588$ [36], die Einheitszelle ist in Abbildung 4.5 dargestellt.

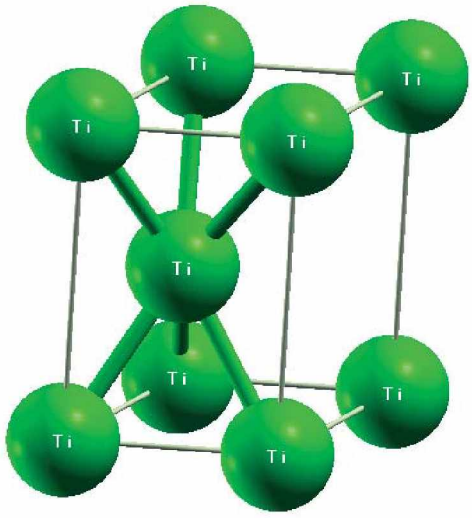

Abbildung 4.5: Primitive Elementarzelle des elementaren Nickel. Erstellt mithilfe von XCrySDen [35].

Konvergenztests

Sowohl das Austauschkorrelationspotential als auch die Werte für σ sind identisch mit den Werten, welche für Nickel eingesetzt wurden. Für den Input des k-Punktnetzes wurden aufgrund der unterschiedlichen Symmetrie andere Eingaben verwendet. Diese sind $8\times8\times5$, $12\times12\times8$, $16\times16\times10$, $20\times20\times13$, $24\times24\times15$ und $28\times28\times18$. Für RK_{max} wurde hierfür ein Wert von 8.0 gewählt. Abbildung

4.6 zeigt das Ergebnis der Konvergenztests bezüglich der k-Punkte für verschiedene σ, wobei auch hier der Kompressionsmodul über die Gesamtpunktezahl in der reduzierten Zone aufgetragen ist.

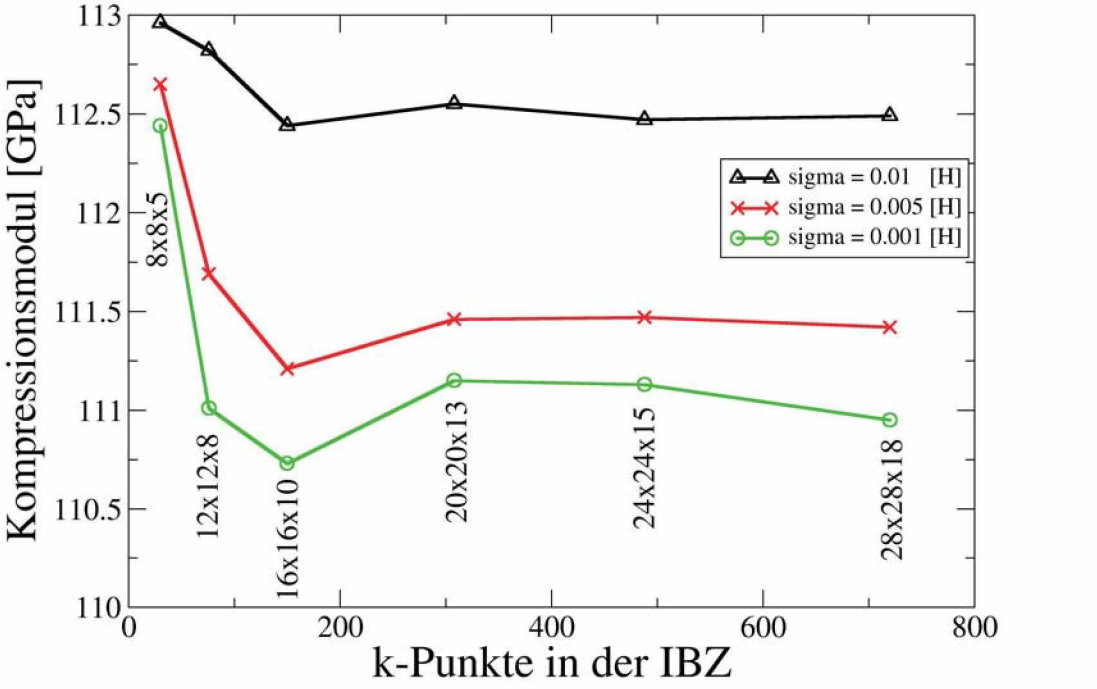

Abbildung 4.6: Ergebnis der Konvergenztests für verschiedene Smearingparameter σ bezüglich der Anzahl der k-Punkte für Titanium.

Aus dem Ergebnis der Konvergenztests ist ersichtlich, dass für Titanium der Kompressionsmodul für verschiedene Werte von σ auf minimal verschiedene Werte konvergiert. Der Unterschied der konvergierten Werte liegt allerdings nur im Bereich von etwa einem Prozent. Hierbei ist B_0 für $\sigma_1 = 0.01$ [Ha] bereits bei weniger als 200 k-Punkten konvergiert. Da bei der Berechnung der elastischen Konstanten eine zumindest gleich große Ungenauigkeit zu erwarten ist, genügt diese Genauigkeit für die folgenden Berechnungen und es wird $\sigma_1 = 0.01$ [Ha] und ein k-Punktnetz von 200 Punkten für die weiteren Berechnungen genutzt. Die 200 k-Punkte entsprechen einem Input von $20 \times 20 \times 13$. Die beiden anderen σ sind circa ab 300 k-Punkten konvergiert, außerdem benötigt eine Berechnung eines Dehnungspunktes bei diesen Smearing-Parametern mehr Zyklen um ein konvergiertes Ergebnis zu erhalten. Aus diesem Grund fällt die Entscheidung auf σ_1.

Für den Konvergenztest von RK_{max} wurden dieselben Eingaben bezüglich der Schrittweite wie bei Nickel verwendet.

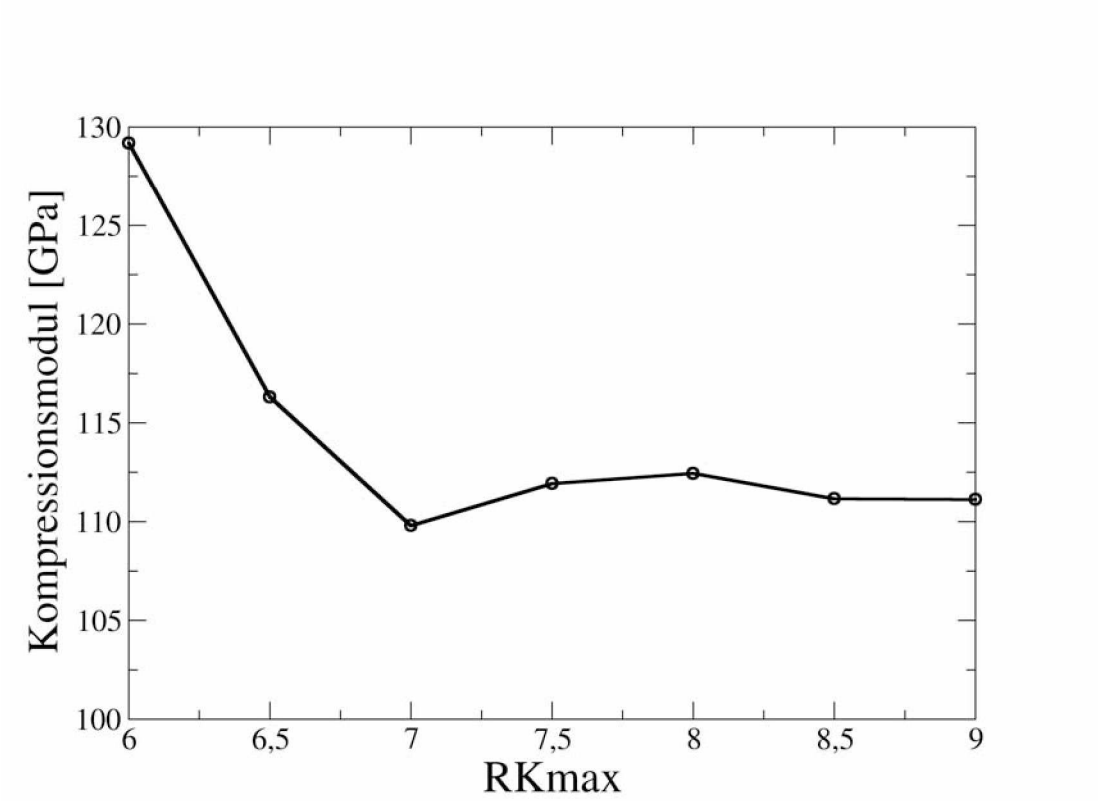

Abbildung 4.7: Ergebnis der Konvergenztests des Kompressionsmoduls bezüglich RK_{max} für Titanium.

Abbildung 4.7 zeigt das Resultat des Konvergenztests und es ist ersichtlich, dass sich B_0 ab einem RK_{max} von 7.5 kaum mehr ändert. Für die Berechnung der optimierten Einheitszelle und die elastischen Konstanten wurde daher 8.0 verwendet.

Optimierung

Aufgrund der hexagonalen Struktur von Titanium muss neben dem Volumen der Einheitszelle auch des Verhältnis der längeren c Achse zur kurzen a Achse berücksichtigt werden. Dazu wurde das in dieser Arbeit vorgestellte Optimisierungsprogramm mit den in den Konvergenztests gefundenen Parametern als Input verwendet. Nach nur wenigen Zyklen war das Ergebniss konvergiert und die neuen, optimierten Parameter lauten $a = 5.57$ [a.u.] und $c/a = 1.576$.

Als Ergebnis der Optimierung kann nun die exciting-Inputdatei für die Berechnung der elastischen Konstanten erstellt werden.

Elastische Konstanten

Als Eingabe für das ElaStic-Programm wird für die maximale Lagrange'sche Dehnung 0.05 und für die Anzahl der Punkte 21 gewählt. Die Matrix der elastischen Konstanten mit 5 unabhängigen Konstanten hat für ein hexagonales System folgende Gestalt:

$$C_{i,j} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 \\ C_{12} & C_{11} & C_{13} & 0 & 0 & 0 \\ C_{13} & C_{13} & C_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{44} & 0 \\ 0 & 0 & 0 & 0 & 0 & (C_{11} - C_{12})/2 \end{bmatrix}$$

Die fünf verschiedenen Verzerrungen zur Bestimmung der elastischen Konstanten eines hexagonalen Gitters haben folgende Dehnungsvektoren: Die erste Verformung entspricht wie im kubischen Fall einer allseitigen Normaldehnung mit $\delta_1 =$ (η, η, η, 0, 0, 0). Die zweite Verzerrung hat einen Vektor $\delta_2 =$ (0.5 η, 0.5 η, - η, 0, 0, 0), dies entspricht einer Stauchung in z -Richtung, wobei sich der Kristall bei konstantem Volumen verformt. Verzerrung Nummer drei und Nummer vier sind lineare Dehnung in z - beziehungsweise y -Richtung, mit den dazugehörenden Vektoren $\delta_3 =$ (0, 0, η, 0, 0, 0) und $\delta_4 =$ (0, η, 0, 0, 0, 0). Die letzte Verformung entspricht einer uniaxialen Dehnung in z -Richtung mit gleichzeitiger Scherung normal zu dieser Richtung. Der dazugehörige Verzerrungsvektor δ_5 lautet (0, 0, η, 2η, 0, 0). Folgende Tabelle zeigt die Ergebnisse der Berechnung:

$$C_{i,j} = \begin{bmatrix} 175.9 & 86.1 & 72.4 & 0 & 0 & 0 \\ 86.1 & 175.9 & 72.4 & 0 & 0 & 0 \\ 72.4 & 72.4 & 194.8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 46.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 46.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 47.6 \end{bmatrix} \text{[GPa]}$$

Der Kompressionsmodul nach Voigt ergibt für Titanium 113.95 [GPa]. Vergleicht man dieses Ergebnis mit jenen aus den Konvergenztests, so liegt der Unterschied bei etwa einem Prozent und zeigt daher gute Übereinstimmung.

4.1.3 Hafnium

Hafnium ist das schwerste der zu untersuchenden Elemente und hat die Ordnungszahl 72. Seine Kristallstruktur ist die gleiche wie die von Titanium und es hat ebenfalls die Raumgruppe Nummer 194. Somit gelten bezüglich der Optimierung dieselben Voraussetzungen wie für Titanium. Die Startwerte sind für den Gitterparameter a= 6.04 [a.u] und für c/a = 1.580 [37]. Die Einheitszelle hat dieselbe Form wie die von Titanium, darum sei zur Veranschaulichung auf Abbildung 4.5 verwiesen.

Konvergenztests

Bei sämtlichen Konvergenztests von Hafnium wurden dieselben Ausgangsdaten wie bei Titanium verwendet. Dies sind 0.01 [Ha], 0.005 [Ha] und 0.001 [Ha] für den Smearing-Parameter. Der konstante Wert für RK_{max} ist 7.0 während der k-Punkt Konvergenztests. Der Input für die k-Punkte selbst ist $8\times8\times5$, $12\times12\times8$, $16\times16\times10$, $20\times20\times13$, $24\times24\times15$ und $28\times28\times18$ für diese Tests. Abbildung 4.8 zeigt das Ergebnis.

Es ist ersichtlich, dass σ_1 und σ_2 ein ähnliches Konvergenzverhalten zeigen und ab circa 300 k-Punkten konvergiert sind.Bei σ_3 kommt es erst bei dichteren k-Punktnetzen zu keinen Änderungen mehr. Aus denselben Gründen wie bei Titanium wurde als Smearingparameter σ_1 mit einem k-Punkt Input von $20\times20\times13$ für die noch durchzuführenden Rechnungen genutzt.

Nach Berechnung des Kompressionsmoduls für Werte von RK_{max} von 6.0 bis 9.0 war noch keine Konvergenz erkennbar, daher wurde hier die Spanne bis 10.0 erweitert. In Abbildung 4.9 ist allerdings erkennbar, dass man ab 9.0 von einer Konvergenz ausgehen kann, und daher dieser Wert für RK_{max} verwendet werden kann.

Optimierung

Da auch Hafnium eine hexagonale Kristallstruktur aufweist, muss auch hier ein Optimierungsschritt eingebaut werden. Auch für Hafnium konvergieren die Werte sehr schnell und die optimierten Werte ergaben für a = 6.029 [a.u.] und für das Verhältnis von c zu a 1.604. Nach Abschluss der Optimierungsrechnungen sind nun alle Informationen verfügbar, um mit der Berechnung der elastischen Konstanten zu beginnen.

Elastische Konstanten

Bei der Bestimmung der elastischen Konstanten von Hafnium war die maximale Lagrange'sche Dehnung 0.06 und die Anzahl der Dehnungspunkte dazwischen

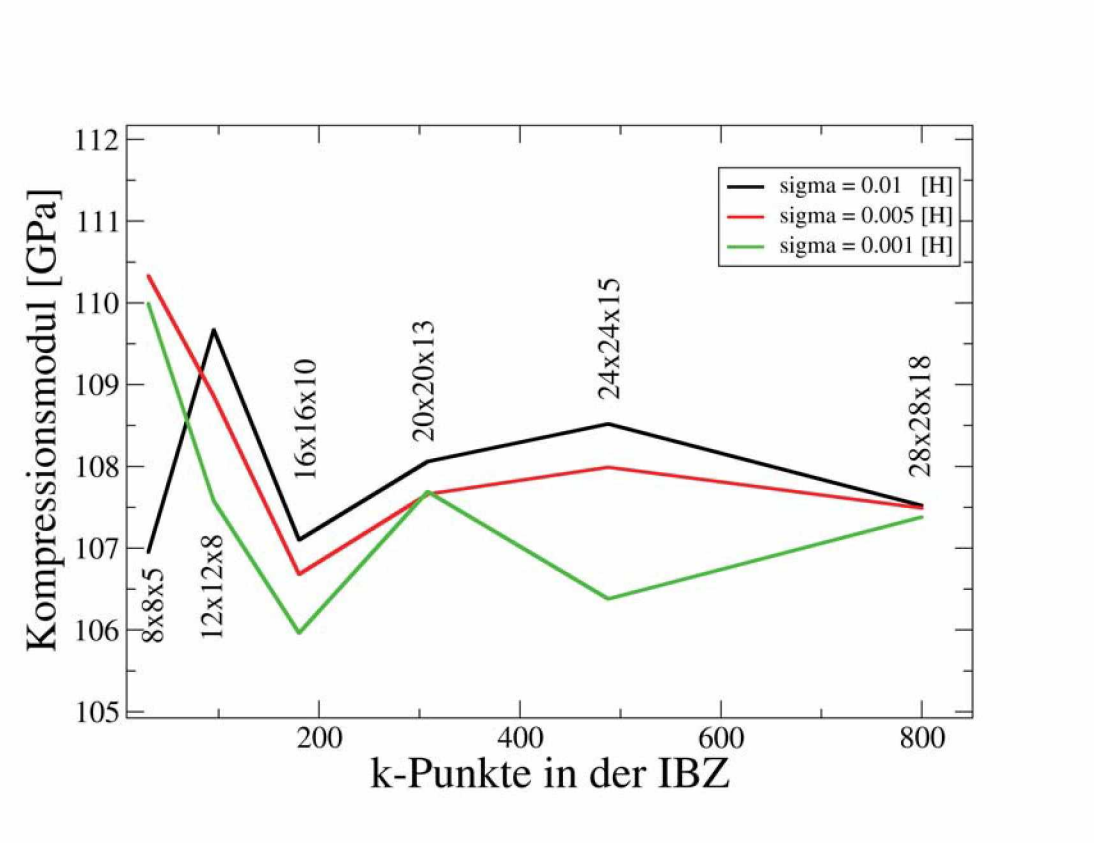

Abbildung 4.8: Ergebnis der Konvergenztests für verschiedene Smearingparameter σ bezüglich der Anzahl der k-Punkte für Hafnium.

beträgt 21. Die Matrix der Konstanten in Voigt'scher Notation wie auch die Dehnungsvektoren wurden bereits für Titanium aufgestellt. Das Ergebnis der Berechnung zeigt folgende Tabelle:

$$C_{ij} = \begin{bmatrix} 185.3 & 74.1 & 69.4 & 0 & 0 & 0 \\ 74.1 & 185.3 & 69.4 & 0 & 0 & 0 \\ 69.4 & 69.4 & 198.9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 52.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 52.3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 55.6 \end{bmatrix} \text{[GPa]}$$

Die Kompressionsmodul nach Voigt ist für Hafnium 111.23 [GPa].

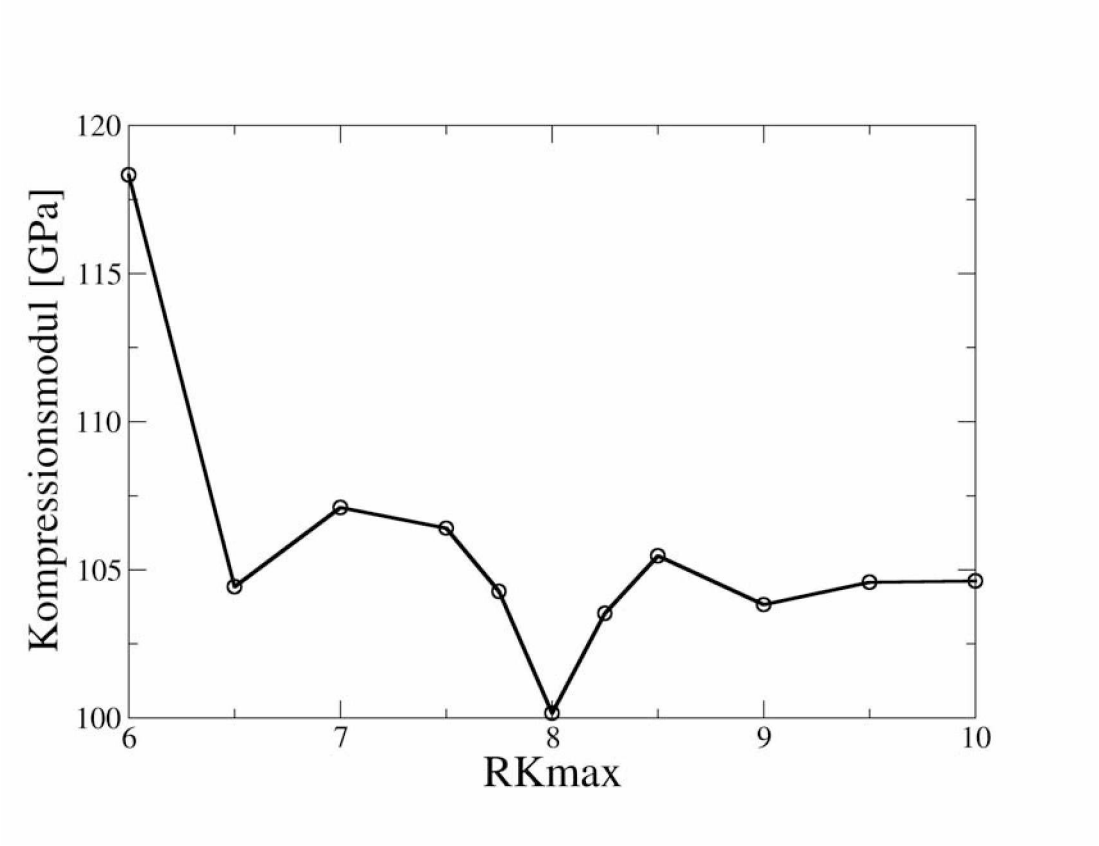

Abbildung 4.9: Ergebnis der Konvergenztests des Kompressionsmoduls bezüglich RK_{max} für Hafnium.

4.1.4 Vergleich

Zum Abschluss der Untersuchungen für die elementaren Strukturen von Ni, Ti und Hf sollen noch die Ergebnisse für die Gleichgewichtsvolumina und die Kompressionsmoduli für verschiedene Austauschkorrelationspotentiale verglichen werden. Diese sind neben LDA und der bereits verwendeten GGA noch ein drittes, nämlich eine weitere GG-Näherung, mit dem Namen PBEsol [38]. Des Weiteren werden die Ergebnisse, die mit dem exciting Programm erzielt wurden, auch mit entsprechenden Resultaten vom Wien2k Paket verglichen. Dabei sollen für beide DFT-Codes die gleichen Werte für RK_{max}, dieselbe Anzahl an k-Punkten und derselbe Wert für den Smearingparameter σ verwendet werden. Die Ergebnisse dieser Berechnungen sind in den Tabellen 4.1 und 4.2 zusammengefasst. Es sei jedoch darauf hingewiesen, dass für die Berechnungen der verschiedenen Näherungen nicht dieselbe Inputdatei eingesetzt wurde, da bei Verwendung einer gleichen maximalen Dehnung nicht immer das Gleichgewichtsvolumen erreicht werden kann. Darum wurde der Kompressionsmodul für jede Näherung

Tabelle 4.1: Ergebnisse bezüglich des Gleichgewichtsvolumen für verschiedene Austauschkorrelationspotentiale, berechnet mit exciting und Wien2k verglichen mit experimentellen Werten

Gleichgewichtsvolumen in a.u.	Ni	Ti	Hf
LDA Wien2k	67.3	218.5	281.1
LDA exciting	66.3	217.7	285.8
GGA Wien2k	72.8	235.4	304.1
GGA exciting	72.3	235.1	308.6
PBEsol Wien2k	69.5	226.5	291.6
PBEsol exciting	68.8	225.8	296.4
experiment. Wert	73.5[a]	238.9[b]	301.5[c]

[a]: nach [34]
[b]: nach [36]
[c]: nach [37]

um sein Gleichgewichtsvolumen, also mit optimierter Zelle bezüglich des Volumens unter diesem Potential, bestimmt. Für die exciting-Berechnungen wurde wieder das Cell-Optimization-exciting Programm verwendet und für die Wien2k-Berechnungen das mit dem Code mitgelieferte Optimierungsprogramm genutzt. Um Unterschiede durch den Fit auszuschließen, wurde für den Birch-Murnaghan-Fit ein Script verwendet, das dieselbe Methode nutzt wie das Cell-Optimization-exciting Programm.

Zunächst sollen die Ergebnisse der verschiedenen Austauschkorrelationspotentiale für einen DFT-Codes diskutiert werden. Aus den Ergebnissen ist ersichtlich, dass sowohl für alle Elemente als auch für beide Codes die bekannte Tatsache [39] bestätigt wird, dass die lokale Dichte-Näherung den Kompressionsmodul weiter überschätzt und die generalisierte Gradienten-Näherung geringere Werte liefert. Die Werte für die PBEsol Näherung liegen durchwegs zwischen den beiden anderen Werten. Als nächstes sollen die Resultate für die verschiedenen Codes betrachtet werden. Hier sind beträchtliche Unterschiede zwischen den einzelnen Elementen. Während bei Titanium keine nenneswerte Unterschiede bestehen und auch bei Hafnium der Unterschied nur wenige Prozent beträgt, ist dieser bei Nickel teilweise größer als 10 %. Die Ursache dieser relativ großen Abweichung bei Nickel konnte trotz Durchführung zusätzlicher Tests und Analysen im Rahmen dieser Diplomarbeit nicht gefunden werden.

Tabelle 4.2: Ergebnisse bezüglich des Kompressionsmoduls für verschiedene Austauschkorrelationspotentiale, berechnet mit exciting und Wien2k verglichen mit experimentellen Werten

Kompressionsmodul in [GPa]	Ni	Ti	Hf
LDA Wien2k	260.7	129.8	119.4
LDA exciting	234.1	128.7	116.5
GGA Wien2k	212.7	113.6	106.2
GGA exciting	192.1	113.2	104.1
PBEsol Wien2k	231.5	121.7	113.8
PBEsol exciting	217.8	121.2	110.3
experiment. Wert	186[a]	110.0[b]	110.6[b]

[a] : nach [47]
[b] : nach [50]

Bezüglich der Volumina der Einheitszellen verhält es sich etwas anders. So sind hier die Ergebnisse von Nickel und Titanium für beide Codes ähnlich beziehungsweise fast identisch, während für Hafnium der Unterschied etwas größer ist, und im Bereich von etwa 2 % liegt.

4.2 Binäre Legierungen

Bei den binären Legierungen wird die binäre Legierung mit stöchiometrischer Zusammensetzung untersucht, jedoch zwei verschiedene Phasen dieser Zusammensetzung. Dies sind die Hoch- und die Niedertemperaturphase mit den Namen B2und B19'. Es werden für beide Phasen Konvergenztests bezüglich der k-Punkte für einen fixen Wert von $\sigma = 0.01$ [Ha] durchgeführt, da auch für die in den Legierungen enthaltenen Elemente dieser Wert angewendet wurde. Anschließend wird eine Optimierung der Einheitszelle durchgeführt, um letztendlich die elastischen Konstanten der beiden Phasen zu bestimmen.

4.2.1 Hochtemperaturphase B2

Die B2-Struktur ist die geordnete Hochtemperaturphase von Nitinol (siehe Kapitel 2). Die Kristallstruktur der Phase ist kubisch, wobei sich an den Ecken des Würfels eine Atomsorte und im Zentrum die andere befindet. Diese Struktur wird auch Cäsiumchlorid-struktur genannt und hat die Raumgruppe mit der Nummer 221. Abbildung 4.10 zeigt die Elementarzelle. Wie bei Nickel ist durch die hohe Symmetrie nur die Optimierung des Einheiszellenvolumens notwendig,

und nach den Konvergenztets bereits abgeschlossen. Als Ausgangswert wurde ein experimenteller Wert für den Zellparameter a von 3.015 [Å] [40] eingesetzt.

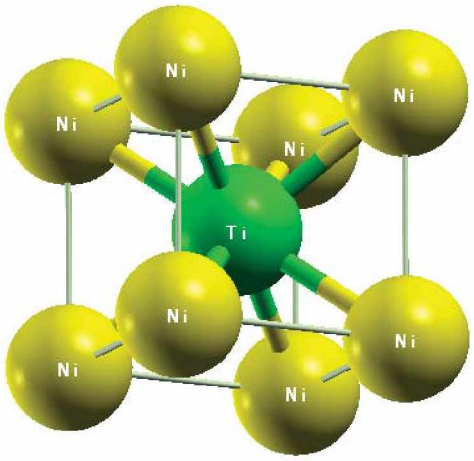

Abbildung 4.10: Elementarzelle der binären B2 Phase. Erstellt mithilfe von XCrySDen [35].

Konvergenztests

Zunächst wird die Konvergenz bezüglich der k-Punkte untersucht. Die in den Inputdateien eingesetzten Netze sind $4\times4\times4$, $8\times8\times8$, $12\times12\times12$, $16\times16\times16$, $20\times20\times20$, $24\times24\times24$ und $28\times28\times28$. Für RK_{max} wurde hier 7.0 eingesetzt, die maximale Dehnung 0.04 bei 11 zu berechnenden Punkten. Abbildung 4.11 zeigt das Ergebnis des Tests.

Man kann ab einer Gesamtzahl an k-Punkten von etwa 150 von einer Konvergenz des Kompressionsmoduls sprechen. Für die weiteren Berechnung wird dementsprechend ein Input von $20\times20\times20$ verwendet, was einer Gesamtzahl von 286 Punkten entspricht. Mit diesem Wert wurde anschließend ein Konvergenztest bezüglich von RK_{max} durchgeführt. Bei gleicher maximaler Dehnung und Anzahl an Berechnungspunkten erhält man das Ergebnis in Abbildung 4.12 .

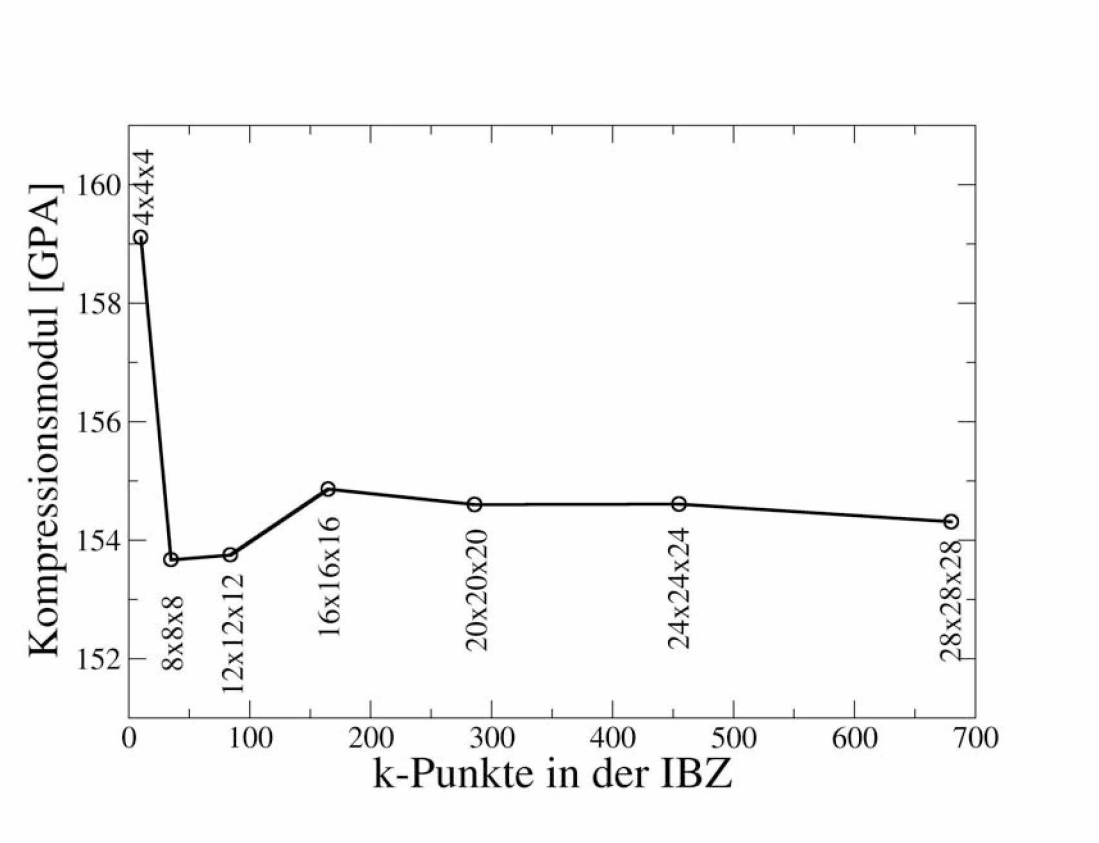

Abbildung 4.11: Ergebnis der Konvergenztests für den Smearingparameter $\sigma = 0.01$ [Ha] bezüglich der Anzahl der k-Punkte für die B2-Phase.

Das Ergebnis dieses Tests ist ein RK_{max} von 8.0. Das durch die Konvergenztests gefundene Volumen beträgt 181.18 $[a.u.^3]$. Damit hat man alle Informationen, um die Inputdatei für die Berechnung der elastischen Konstanten der B2 Phase zu erstellen.

Elastische Konstanten

Siehe Kapitel 4.1.1 für die allgemeine Form der Voigt'schen Matrix der elastischen Konstanten und die Verfomungsvektoren für ein Material mit kubischer Struktur. Für diese Berechnung wurde eine maximale Lagrange'sche Dehnung von 0.05 mit 11 Punkten eingegeben. Das Ergebnis zeigt folgende Matrix:

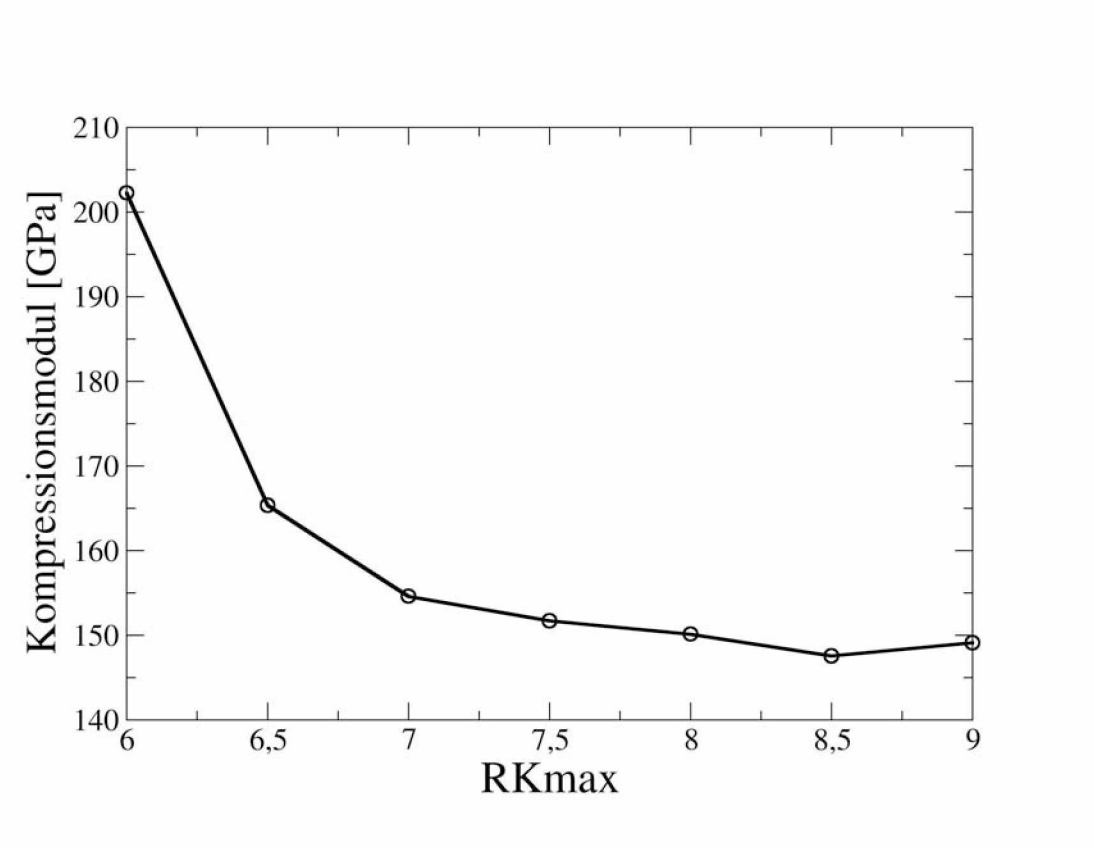

Abbildung 4.12: Ergebnis der Konvergenztests des Kompressionsmoduls bezüglich RK_{max} für die B2-Phase.

$$C_{i,j} = \begin{bmatrix} 176.2 & 137.5 & 137.5 & 0 & 0 & 0 \\ 137.5 & 176.2 & 137.5 & 0 & 0 & 0 \\ 137.5 & 137.5 & 176.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 34.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 34.3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 34.3 \end{bmatrix} \text{[GPa]}$$

Die Ermittlung des Kompressionsmoduls aus den elastischen Konstanten ergibt 152.77 [GPa], was wiederum gut mit den direkten Kalkulationen des Moduls während der Konvergenztests übereinstimmt.

4.2.2 Tieftemperaturphase B19'

Die zweite binäre Struktur mit stöchiometrischer Zusammensetzung ist die Tieftemperaturphase B19'. Die Kristallstruktur ist monoklin, womit die Einheitszelle

durch vier unabhängige Parameter definiert ist. Dies hat einen großen Einfluss auf den Rechenaufwand, sowohl für die Optimierung der Einheitszelle als auch die Bestimmung der elastischen Konstanten. Bei der Optimierung müssen 4 Parameter betrachtet werden, siehe auch Tabelle 3.1. Dazu gibt es nun 13 unabhängige elastische Konstanten. Des Weiteren befinden sich 4 Atome in der Elementarzelle, mit insgesamt 12 internen Freiheitsgraden Abbildung 4.13. Die Ausgangsgitterparameter sind $a = 8.91$ [a.u.], $b = 5.56$ [a.u.] und $c = 7.67$ [a.u.], der dazugehörige monokline Winkel $\gamma = 100.60°$ [40].

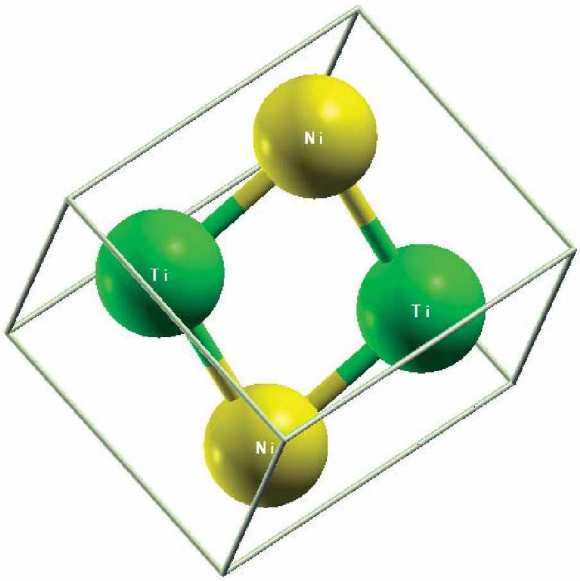

Abbildung 4.13: Elementarzelle der binären B19' Phase, erstellt mithilfe von XCrySDen [35].

Konvergenztests

Die Konvergenztests bezüglich des k-Punktnetzes wurden mit einer Maximaldehnung von 0.05 mit 11 dazwischenliegenden Punkten gemacht. Der Input der k-Punkte war $7 \times 11 \times 8$, $9 \times 15 \times 10$, $9 \times 17 \times 11$, $13 \times 20 \times 14$, und $14 \times 22 \times 16$, jener für RK_{max} immer 7.0. Das Ergebnis ist in Abbildung 4.14 dargestellt.

Ab einer Gesamtzahl von rund 500 k-Punkten ändert sich der Kompressionsmoduls mit nur noch gering. Dies führt zu der Entscheidung, als weiteren Input für die k-Punkte $9 \times 17 \times 11$ zu verwenden. Als nächster Schritt wurde die Konvergenz

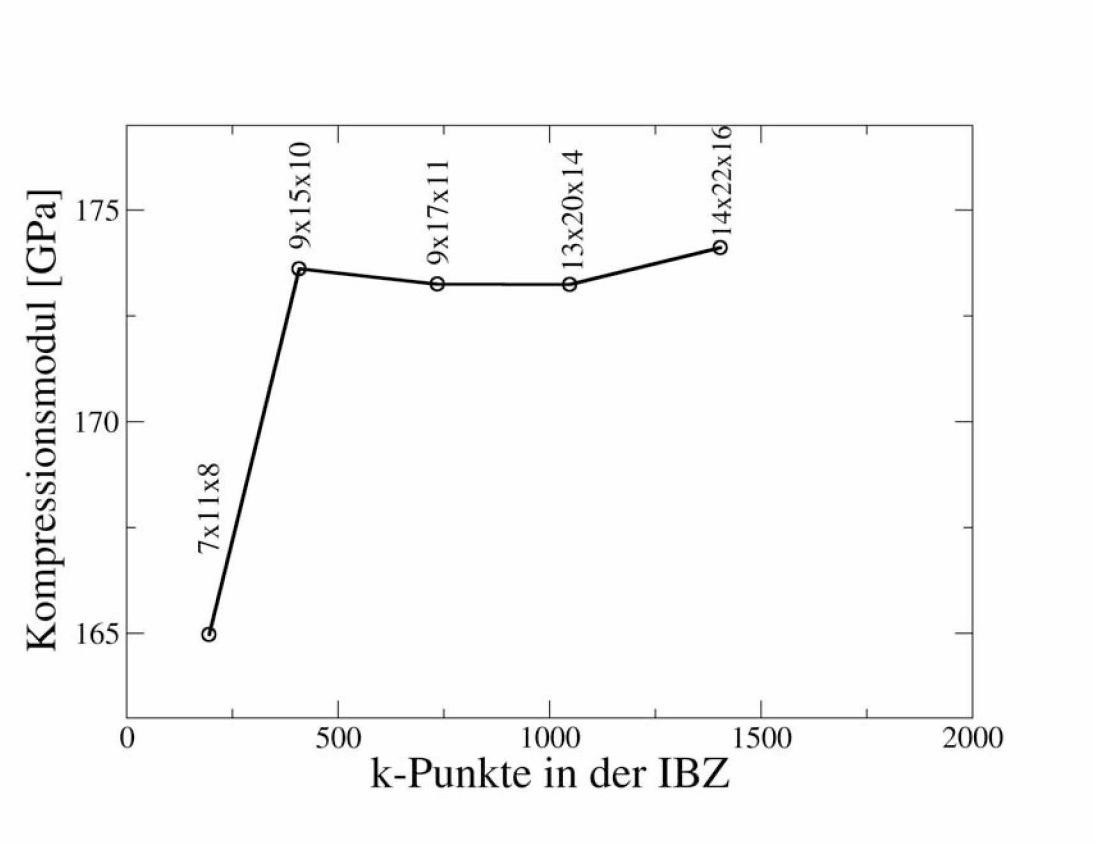

Abbildung 4.14: Ergebnis der Konvergenztests für den Smearingparameter $\sigma = 0.01$ [Ha] bezüglich der Anzahl der k-Punkte für die B19'-Phase.

bezüglich RK_{max} untersucht. ein konvergierter Wert bezüglich RK_{max} zu finden. Das Ergebnis davon ist Abbildung 4.15 zu entnehmen. Die Entscheidung für das weiter verwendete RK_{max} fällt auf einen Wert von 8.0.

Optimierung

Mit den gefundenen Werten kann somit die Optimierung der Einheitszelle durchgeführt werden. Den Verlauf der zu optimierenden Gittergrößen V, b/a, c/a und γ mit der Anzahl der Optimierungszyklen ist in Abbildung 4.16 zu sehen. Aus den konvergierten Werten für diese Größen lassen sich die Gittervektoren bestimmen. Die Längen dieser sind $a = 8.91$ [a.u.], $b = 5.61$ [a.u.] und $c = 7.45$ [a.u.]. Der monokline Winkel γ kann direkt abgelesen werden und ist 99.39°. Daraus lässt sich nun die Inputdatei für die elastische Konstantenberechnung bereitstellen.

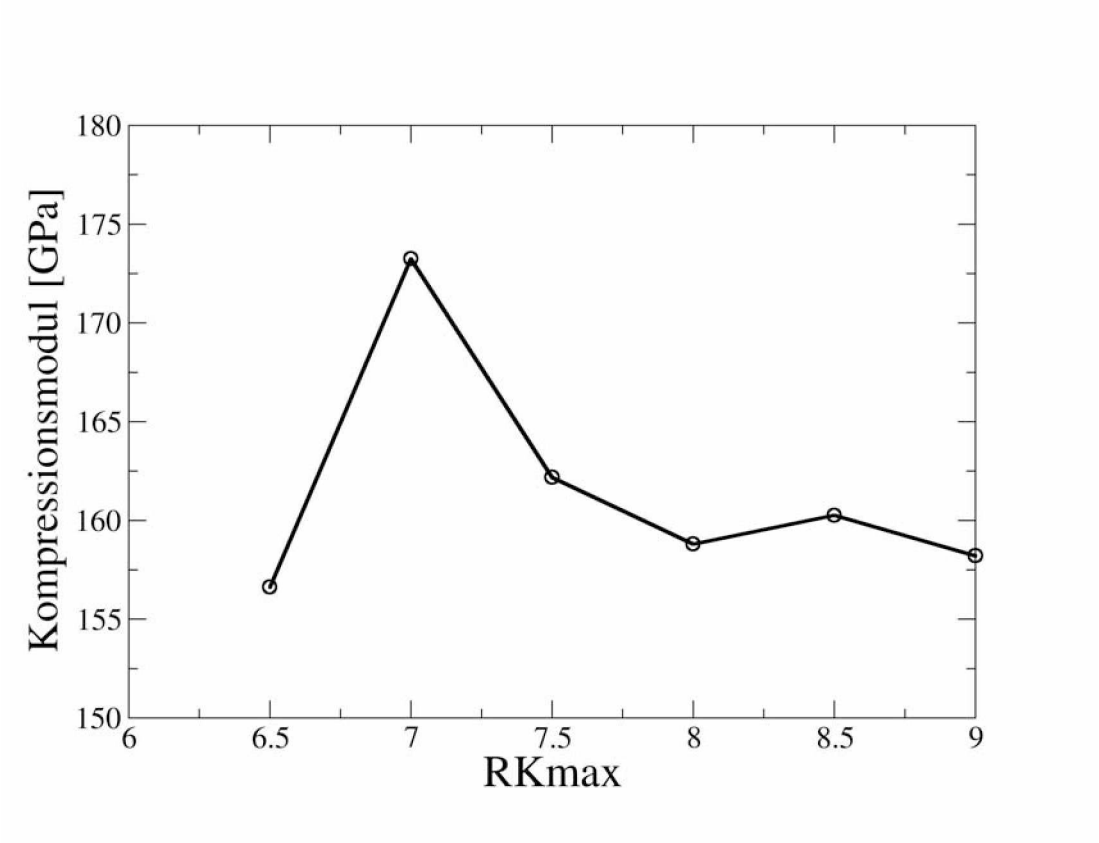

Abbildung 4.15: Ergebnis der Konvergenztests des Kompressionsmoduls bezüglich RK_{max} für die B19'-Phase.

Elastische Konstanten

Für die Berechnung der elastischen Konstanten wurde das Maximum der Dehnung auf 0.05 gesetzt, wobei für jeden der 13 Verformungstypen 15 Dehnungswerte berechnet wurden. Das ergibt insgesamt bereits 195 Einzelsimulationen und einen damit verbundenen großen Rechenaufwand. Die Matrix der elastischen Konstanten ist für ein monoklines System folgendermaßen aufgebaut:

$$C_{ij} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & C_{16} \\ C_{12} & C_{22} & C_{23} & 0 & 0 & C_{26} \\ C_{13} & C_{23} & C_{33} & 0 & 0 & C_{36} \\ 0 & 0 & 0 & C_{44} & C_{45} & 0 \\ 0 & 0 & 0 & C_{45} & C_{55} & 0 \\ C_{16} & C_{26} & C_{36} & 0 & 0 & C_{66} \end{bmatrix} \text{ [GPa]}$$

Diese 13 unabhängigen Konstanten bedeuten auch 13 unterschiedliche Arten der Verformung, die nötig sind, um diese zu bestimmen. Die jeweiligen Verformungs-

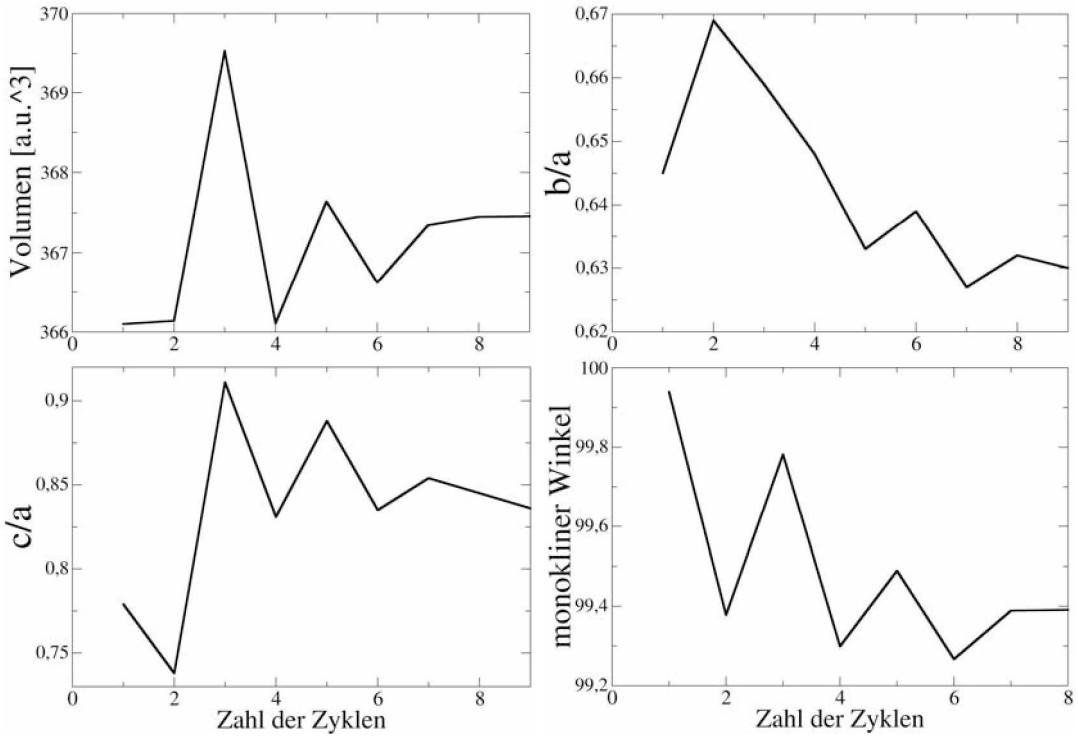

Abbildung 4.16: Darstellung der Gitterparameter der B19'-Phase mit der Anzahl der Optimierungszyklen: Volumen linkss oben, b/a rechts oben, c/a links unten und γ rechts unten.

vektoren einer bestimmten Lagrange'schen Dehnung und welche greifbare Bedeutung diese haben, wird im Folgenden beschrieben: Die erste Deformation ist wieder eine allseitige Normaldehnung mit $\delta_1 = (\ \eta, \eta, \eta, 0, 0, 0)$. Verzerrung Nummer 2 und 3 sind volumserhaltende Stauchungen in y- beziehungweise x -Richtung. Die Volumskonstanz wird durch entsprechende Dehnungen in den beiden anderen Richtungen mit umgekehrten Vorzeichen und halber Größe bewerkstelligt, die Vektoren lauten $\delta_2 = (0.5\eta, -\eta, 0.5\eta, 0, 0, 0)$ und $\delta_3 = (-\eta, 0.5\eta, 0.5\eta, 0, 0, 0)$. Die nächsten beiden Deformationen sind Kombinationen aus Normal- und Scherdehnungen, wobei Volumskonstanz durch eine gleichgroße, negative Normaldehnung in eine andere Richtung hervorgerufen wird. Die Verzerrungsvektoren lauten $\delta_4 = (\eta, -\eta, 0, 0, 0, 2\eta)$ und $\delta_5 = (0, \eta, -\eta, 0, 0, 2\eta)$. Verzerrung Nummer 6 entspricht einer Normaldehnung in x -Richtung bei gleichzeitiger umgekehrter Normaldehnung in y -Richtung $\delta_6 = (\eta, -\eta, 0, 0, 0, 0)$. Es folgt eine Scherung in zwei Richtungen mit $\delta_7 = (0\ , 0, 0, 2\eta, 2\eta, 0)$. Die nächste Deformation entspricht einer uniaxialen Normaldehnung in x -Richtung bei gleichzeitiger Scherung normal darauf: $\delta_8 = (\eta\ , 0, 0, 0, 0, 2\eta)$. Die letzten fünf Deformationen sind jeweils uniaxial, wobei die ersten beiden Normaldehnungen in y - und z -Richtung sind und die restlichen drei Scherungen in unterschiedliche Richtung. Die Deformati-

onsvektoren lauten: δ_9 =(0 , η, 0, 0, 0, 0), δ_{10} =(0 , 0, η, 0, 0, 0), δ_{11} =(0 , 0, 0, 2η, 0, 0), δ_{12} =(0 , 0, 0, 0, 2η, 0) und δ_{13} =(0 , 0, 0, 0, 0, 2η).

In nachfolgender Matrix ist das Ergebnis für die elastischen Konstanten der B19' Phase in GigaPascal [GPa] zu sehen:

$$C_{i,j} = \begin{bmatrix} 236.4 & 120.8 & 109.1 & 0 & 0 & 23.4 \\ 120.8 & 195.2 & 139.4 & 0 & 0 & 9.2 \\ 109.1 & 139.4 & 240.7 & 0 & 0 & -0.8 \\ 0 & 0 & 0 & 80.2 & -2.0 & 0 \\ 0 & 0 & 0 & -2.0 & 93.2 & 0 \\ 23.4 & 9.2 & -0.8 & 0 & 0 & 30.4 \end{bmatrix} \text{[GPa]}$$

4.3 Ternäre Legierungen

Im Rahmen der Untersuchung der elastischen Konstanten im NiTiHf-System werden zwei Strukturen mit verschiedener Zusammensetzung untersucht. Die Gitterstrukturen stammen aus vorhergehenden Ab-initio Berechnungen mithilfe der bereits in Kapitel 3 kurz erwähnten "cluster expansion" -Methode [41]. Die Berechnungen wurden von Dr. Jürgen Spitaler durchgeführt [42] , wobei das "Alloy Theoretic Automated Toolkit", kurz ATAT, [43] Programm eingesetzt wurde, unter Verwendung der ATAT@Wien2k Schnittstelle [44]. Zur genauen Funtionsweise der Programme und zur zugrundeliegender Theorie hinter der Methode sei auf die angegebenen Referenzen verwiesen. Als Ergebnis einer cluster expansion erhält man zunächst eine reihe von Strukturen, die die energetisch günstigste, geordnete Anordnung der Atome bei einer bestimmten Zusammensetzung. Mit diesen können weitere Berechnungen, wie zum Beispiel die Ermittlung der elastischen Konstanten, durchgeführt werden. Mithilfe der "cluster expansion"-Methode wurde nun das System mit 50 Prozent Ni und einem variablen Titanium-Hafnium Verhältnis untersucht, wobei Hafnium das Titanium substituiert. Man spricht in diesem Zusammenhang auch von einem quasibinären System. Das heißt, 50 Prozent der Gitterplätze werden entweder von Titanium oder Hafnium besetzt. Zwei Strukturen im titaniumreicheren Bereich wurden so gefunden. In der ersten Legierungen verteilen sich die 50 verbliebenen Prozent zu 25 beziehungsweise 75 Prozent auf Hafnium und Titanium während sich in der zweiten 33 - beziehungweise 67 Prozent auf Hafnium und Titanium verteilen. Des Weiteren ist zu erwähnen, dass die Konvergenztests und die Optimierung für beide Legierungen bereits von Dr. Spitaler mithilfe des Wien2k DFT-Codes im Rahmen der Untersuchung des Systems bereits durchgeführt worden sind. Die erhaltenen Inputdateien können daher direkt für die Berechnung der elastischen Konstanten eingesetzt werden für die auch der Wien2k code eingesetzt wurde. Die durchgeführten Konvergenztests ergaben für RK_{max} einen Wert von 8.0. Bezüglich der

Anzahl der k-Punkte verhält es sich folgendermaßen: Da mit diesem Programm viele unterschiedliche Strukturen berechnet werden, wurde das Produkt aus der Anzahl der Atome in der Einheitszelle N_{at} und der Zahl der k-Punkte in der nicht reduzierten Brillouinzone N_k gesucht. Mit einer solchen Wahl ist gewährleistet, dass die k-Punkt Dichte im reziproken Raum eine Konstante darstellt. Das Ergebnis dieser Tests ergab einen Wert von $N_{at} \times N_k = 8000$.

Bei Verwendung der Strukturdateien, die im Rahmen der cluster expansion erstellt werden, ist folgendes zu beachten: Um die unterschiedlichen Zusammensetzungen der Legierungen zu erreichen, werden sogenannte Superzellen erstellt. Dazu werden die Basiszellen mit einer Transformationsmatrix multipliziert und das Ergebnis daraus ist die neue Einheitszelle. Dadurch kann sich allerdings das kartesische Koordinatensystem, das vom Wien-Code für diese Zellen angewendet wird ändern. Dies ist für die Berechnungen im Rahmen der "cluster-expansion"-Methode kein Problem, möchte man jedoch die Ergebnisse von elastischen Konstantenberechnungen verschiedener Strukturen des Systems vergleichen, führt die Tatsache unterschiedlicher Koordinatensysteme zu Problemen. Denn die Darstellung eines Tensors hängt von der Wahl des Koordinatensystems ab. Da dies auch für die elastischen Konstanten gilt, muss für jede Struktur die Transformationsmatrix zwischen dem Koordinatensystem der Referenzzelle und dem aktuellen Koordinatensystem bestimmt werden, mithilfe dieser es dann möglich ist, mit einem im ElaStic-Paket enthaltenen Tool die elastischen Konstanten zu transformieren. Die Gittergrößen der Referenzzelle sind eine Kombination nach dem Gesetz von Vegard [45] aus denen der binären NiTi- und der binären NiHf-Struktur.

Um die benötigte Transformationsmatrix zu erhalten, wurde ein Script erstellt. Darin wird zunächst durch Multiplikation der Matrix mit den Referenzbasisvektoren und der zur Superzellenerstellung eingesetzten Transformationmatrix, die Basisvektoren der Superzelle ermittelt. Es ist bekannt [5], auf welche Weise der Wien-Code das Koordinatensystem bei bekannten Gitterparametern erstellt. Und zwar wird die z-Achse entlang des c-Vektors gelegt, die y-Achse wird normal auf die z-Achse in jene Ebene gelegt die c- und b-Vektor aufspannen, und die x-Achse wird normal auf die beiden anderen Achsen erstellt. Auf diese Weise wird aus den nun bekannten Einheitszellenmatrizzen die beiden Koordinatensysteme aufgestellt, und die Transformationsmatrix $\alpha_{i,j}$ wie folgt ermittelt:

$$\alpha_{i,j} = \begin{pmatrix} cos\left(\overrightarrow{e_R^X} \overrightarrow{e_B^X}\right) & cos\left(\overrightarrow{e_R^Y} \overrightarrow{e_B^X}\right) & cos\left(\overrightarrow{e_R^Z} \overrightarrow{e_B^X}\right) \\ cos\left(\overrightarrow{e_R^X} \overrightarrow{e_B^Y}\right) & cos\left(\overrightarrow{e_R^Y} \overrightarrow{e_B^Y}\right) & cos\left(\overrightarrow{e_R^Z} \overrightarrow{e_B^Y}\right) \\ cos\left(\overrightarrow{e_R^X} \overrightarrow{e_B^Z}\right) & cos\left(\overrightarrow{e_R^Y} \overrightarrow{e_B^Z}\right) & cos\left(\overrightarrow{e_R^Z} \overrightarrow{e_B^Z}\right) \end{pmatrix}$$

Die Vektoren e^i stehen für die Einheitsvektoren der beiden Koordinatensysteme. Der Index R steht für das Referenzkoordinatensystem und der Index B für das Koordinatensystem in dem die elastischen Konstanten berechnet wurden. Eine grafische Darstellung der Koordinatensysteme, sowie der Einheitszellen ist für die erste ternäre Legierung in Abbildung 4.17 zu sehen.

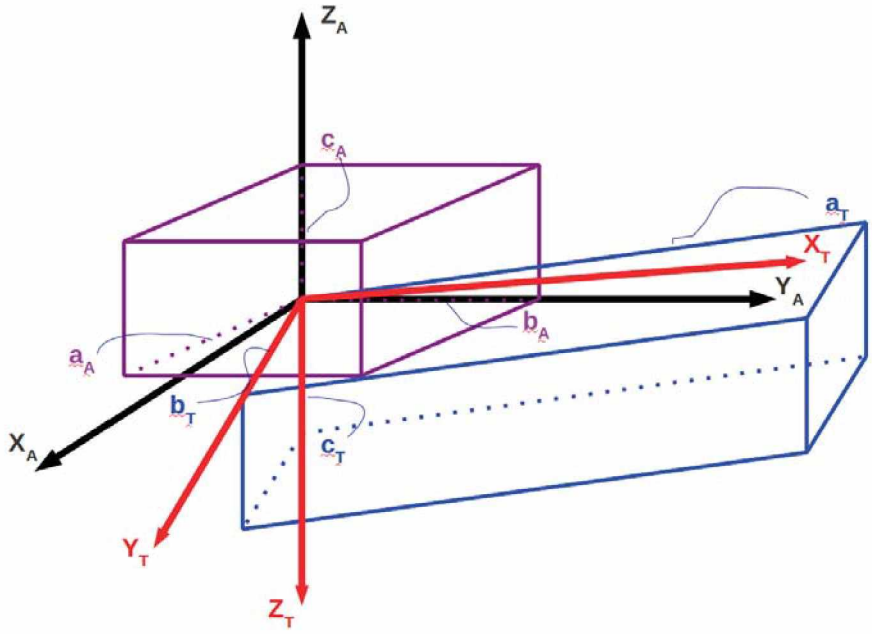

Abbildung 4.17: Darstellung des Referenzkoordinatensystems und des Koordinatensystems in dem die Berechnungen der elastischen Konstanten durchgeführt wurden (Berechnungskoordinatensystem) mit den dazugehörigen Einheitszellen der ternären NiTi75Hf25-Legierung. Ausgehend von der Referenzeinheitszelle (violett) wird das ursprüngliche kartesische Koordinatensystem definiert (schwarz). Durch Multiplikation der Basiszelle mit einer Transformationsmatrix erhält man die Superzelle (blau) und durch diese das damit verbundene kartesische Berechnungskoordinatensystem (rot).

4.3.1 NiTi75Hf25

Die erste untersuchte ternäre Legierung ist jene mit 50 Prozent Nickel, 37,5 Prozent Titanium und 12,5 Prozent Hafnium. Die Hafniumatome ersetzen die Titaniumatome auf deren Gitterplätzen. In der Einheitszelle befinden sich insgesamt 16 Atome, der Zusammensetzung entsprechend sind dies acht Nickel-, sechs Titanium- und zwei Hafniumatome. Die monokline Elementarzelle ist in Abbildung 4.18 dargestellt. Die bereits optimierten Gitterparameter der Zelle sind $a = 21.16$ [a.u.], $b = 9.65$ [a.u.] , $c = 7.66$ [a.u.] und der monokline Winkel $\gamma = 92.5°$. Mit den 16 Atomen und den Ergebnissen der Konvergenztests ergibt

sich ein Input von 500 k-Punkten. Das angewendete Austauschkorrelationpotential ist abermals eine generalisierte Gradienten-Näherung[?]. Die Eingaben des ElaStic-Programms sind 0.05 für die maximale Dehnung und 11 Punkte innerhalb derselben. Nach der Durchführung der Wien2k Berechnungen ergibt sich für den Elastizitätstensor im Koordinatensystem der Superzelle:

$$C^B_{i,j} = \begin{bmatrix} 167.9 & 123.7 & 128.7 & 0 & 0 & 22.9 \\ 123.7 & 233.2 & 184.6 & 0 & 0 & -23.6 \\ 128.7 & 184.6 & 241.0 & 0 & 0 & -1.1 \\ 0 & 0 & 0 & 84.2 & 4.2 & 0 \\ 0 & 0 & 0 & 4.2 & 85.4 & 0 \\ 22.9 & -23.6 & -1.1 & 0 & 0 & 29.6 \end{bmatrix} \text{[GPa]}$$

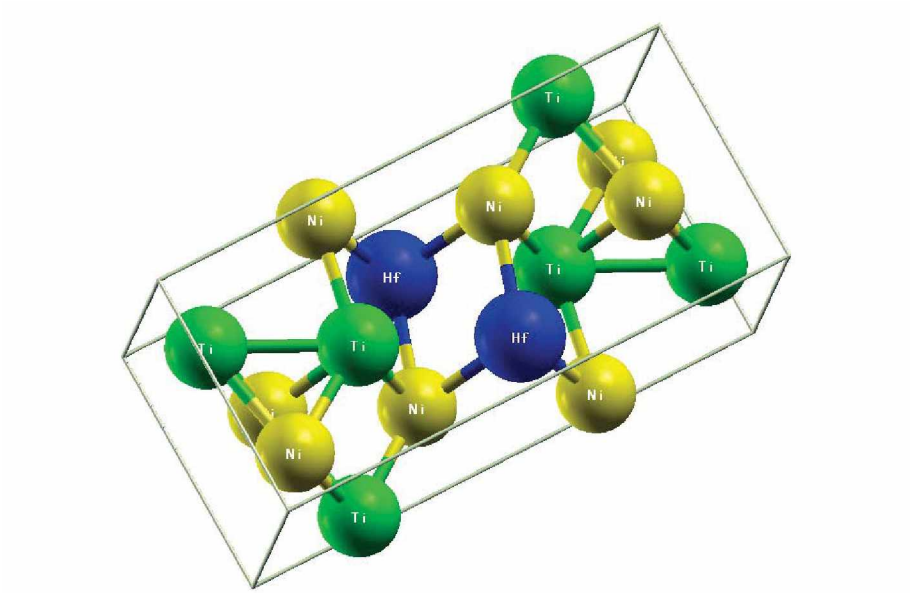

Abbildung 4.18: Elementarzelle der ternären Legierung mit 50 % Nickel, 37,5 % Titanium und 12,5 % Hafnium. Erstellt mithilfe von XCrySDen [35]

Zum Vergleich der Ergebnisse mit den Resultaten für die binäre Referenzzelle müssen diese elastische Konstanten wie erwähnt noch transformiert werden. Aus der Matrix der Basisvektoren der Referenzzelle:

$$\begin{pmatrix} 4.912690 & -1.482290 & 0 \\ 0 & 3.261600 & 0 \\ 0 & 0 & 4.093170 \end{pmatrix} \text{[Å]}$$

und der Transformationsmatrix zur Erstellung der Superzelle:

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & -1 \\ 1 & 2 & 0 \end{pmatrix}$$

ergibt sich nach Anwendung des vorhin beschriebenen Scripts folgende Transformationsmatrix zwischen den Koordinatensystemen.

$$\begin{pmatrix} -0.34054 & 0.94023 & 0 \\ 0.94023 & 0.34054 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

Mit dieser Transformationsmatrix kann nun der Elastizitätstensor in das neue kartesische Koordinatensystem transformiert werden. Die neue Matrix der elastischen Konstanten lautet:

$$C_{i,j} = \begin{bmatrix} 245.3 & 104.4 & 178.8 & 0 & 0 & 10.3 \\ 104.4 & 194.3 & 134.5 & 0 & 0 & 10.0 \\ 178.8 & 134.5 & 241.0 & 0 & 0 & 17.1 \\ 0 & 0 & 0 & 82.6 & -3.6 & 0 \\ 0 & 0 & 0 & -3.6 & 87.0 & 0 \\ 10.3 & 10.0 & 17.1 & 0 & 0 & 10.3 \end{bmatrix} \text{[GPa]}$$

4.3.2 NiTi67Hf33

Als zweite ternäre Legierung wurde eine Zusammensetzung gewählt von 50 Prozent Nickel, 33 Prozent Titanium und die restlichen 17 Prozent sind Hafnium. Die Einheitszelle enthält insgesamt 12 Atome und ist in Abbildung 4.19 dargestellt. Entsprechend der Zusammensetzung sind das sechs Nickelatome, vier Titanium- und zwei Hafniumatome. Den Konvergenztests nach sollen bei 12 Atomen mindestens 666 k-Punkte verwendet werden, als Input wurden 700 genutzt. Die Gitterparameter sind $a = 16.43$ [a.u.] , $b = 9.66$ [a.u.] und $c = 7.65$ [a.u.] . Der monokline Winkel beträgt 101.04° . Auch hier wurde bei 11 zu berechnenden Punkten pro Deformation, eine maximale Lagrange'sche Dehnung von 0.05 gewählt. Das Resultat der Berechnung ergibt folgende Matrix für das ursprüngliche Koordinatensystem:

$$C^B_{i,j} = \begin{bmatrix} 177.3 & 119.0 & 125.9 & 0 & 0 & 20.2 \\ 119.0 & 229.0 & 176.9 & 0 & 0 & -23.1 \\ 125.9 & 176.9 & 237.1 & 0 & 0 & -2.7 \\ 0 & 0 & 0 & 82.4 & -3.1 & 0 \\ 0 & 0 & 0 & -3.1 & 83.9 & 0 \\ 20.2 & -23.1 & -2.7 & 0 & 0 & 27.0 \end{bmatrix} \text{[GPa]}$$

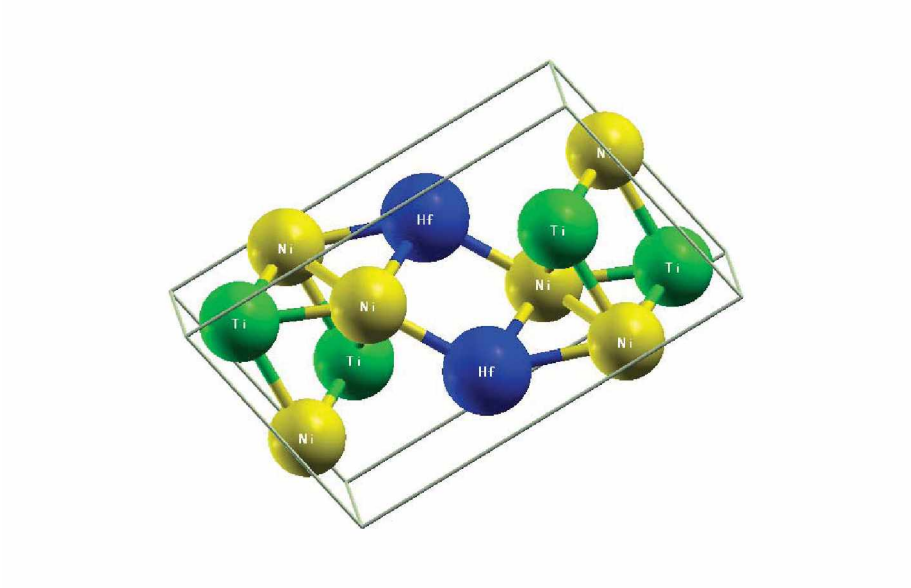

Abbildung 4.19: Elementarzelle der ternären Legierung mit 50 % Nickel, 33 % Titanium und 17 % Hafnium. Erstellt mithilfe von XCrySDen [35]

Auch für diese Legierung müssen abschließend noch die elastischen Konstanten in jene für das ursprüngliche Koordinatensystem gebracht werden. Die Referenzmatrix ist dieselbe wie die der vorherigen Legierung und die Matrix zur Erstellung der Superzelle lautet:

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & -1 \\ 1 & 3 & 0 \end{pmatrix}$$

Diese unterscheidet sich von der für die vorige ternäre Legierung nur in einer Position des dritten Vektors. Da aber dieser Vektor bei Erstellung der Superzelle nur Einfluss auf den a-Vektor hat und dieser letztendlich keinen Einfluss auf das neue Koordinatensystem, lautet die Transformationsmatrix zwischen den Koordinatensystemen auch hier

$$\begin{pmatrix} -0.34054 & 0.94023 & 0 \\ 0.94023 & 0.34054 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

.

Mit dieser können wieder die elastischen Konstanten im Koordinatensystem der Referenzstruktur ermittelt werden und diese lauten:

$$C^R_{i,j} = \begin{bmatrix} 240.0 & 103.9 & 172.7 & 0 & 0 & 10.7 \\ 103.9 & 196.6 & 130.1 & 0 & 0 & 3.6 \\ 172.7 & 130.1 & 237.1 & 0 & 0 & 14.3 \\ 0 & 0 & 0 & 81.7 & -2.9 & 0 \\ 0 & 0 & 0 & -2.9 & 84.6 & 0 \\ 10.7 & 3.6 & 14.3 & 0 & 0 & 11.9 \end{bmatrix} \text{[GPa]}$$

Kapitel 5

Diskussion der Ergebnisse

5.1 Qualität der Ergebnisse

5.1.1 Genauigkeit

Es gibt einige Einflüsse auf die Genauigkeit der vorgestellten Ergebnisse. Diese sind zunächst einmal die verwendete Näherung für das Austauschkorrelationspotential. Durch die dargestellten Ergebnisse der Kompressionsmoduli und der Gleichgewichtsvolumina für unterschiedliche Näherungen sieht man, dass die Ergebnisse allein dadurch bereits mehr oder weniger stark variieren können. So kann man davon ausgehen, dass die Ergebnisse für die elastischen Konstanten eine ähnliche Abhängigkeit vom Austauschkorrelationspotential zeigen, und dementsprechend stark von experimentellen Resultaten abweichen können.

Ein weiterer Punkt, der einen großen Einfluss auf die Genauigkeit der angegebenen elastischen Konstanten hat, ist die Anzahl an Dehnungspunkten und die maximale Dehnung für die Bestimmung dieser. Je größer diese Zahl ist, umso genauer wird das Resultat. Dies wird am besten dadurch ersichtlich, wie groß der Einfluss der Ordnung des eingsesetzten Fit-Polynoms ist. Wenn sich die Ergebnisse für verschiedene Ordnungen des Polynoms der zweiten Ableitung der Energie bezüglich der Dehnung für eine bestimmte Deformation um wenige Gigapascal unterscheiden, so kann auch davon ausgegeangen werden, dass der Fehler in den berechneten elastischen Konstanten im Bereich von Gigapascal liegt.

5.1.2 Vergleich mit Literarurwerten

Beim Vergleich der Resultate mit Werten aus der Literatur muss unterschieden werden, ob diese Werte ebenfalls aus theoretischen Berechnungen mit anderen Codes stammen, oder ob diese mittels experimenteller Methoden bestimmt worden sind. Beim Vergleich mit experimentellen Daten muss berücksichtigt werden,

dass die elastischen Konstanten und damit auch der Kompressionsmodul von der Temperatur abhängen [46], während theoretische Ergebnisse für $T = 0[K]$ berechnet werden.

Zunächst sollen die Ergebnisse für elementares Nickel mit Literaturdaten verglichen werden (Tabelle 5.1). Die berechneten elastischen Konstanten zeigen großteils gute Übereinstimmung mit den gefundenen Vergleichswerten. Ledeglich C_{44} weicht von den anderen Werten ab, jedoch sei noch einmal darauf hingewiesen, dass die experimentellen Werte bei erhöhten Temperaturen gemessen werden.

Tabelle 5.1: Vergleich von unterschiedlichen Ergebnissen bezüglich der elastischen Konstanten in [GPa] und des Gitterparameters a in [a.u.] von Nickel.

	diese Arbeit[a]	experimentell [47]	theoretisch[b]	theoretisch[c]
a	6.62	6.65	6.66	-
C_{11}	250.0	248	260.7	275.5
C_{12}	164.6	155	156.0	160.1
C_{44}	114.3	124	121.6	126.3
B	193.1	186	190.9	195.6

[a] : exciting [3]
[b] : "Projektor Augmented Wave"-Methode(PAW), GGA nach Perdew, Burke und Ernzerhof [48]
[c]: PAW-Methode , GGA nach Perdew und Wang [49]

Ein Vergleich der Gitterkonstanten, des Kompressionsmoduls sowie der elastischen Konstanten von Titanium mit Literaturdaten ist Tabelle 5.2 zu entnehmen. Auch für Titanium stimmen die Resultate für die elastischen Konstanten tendenziell mit den anderen Ergebnissen überein. Die Fehler liegen für bestimmte Konstanten jedoch im Bereich von 5 Prozent beim Vergleich mit den Werten aus Experimenten.

Als nächstes der Vergleich für Hafnium in Tabelle 5.3. Auch für Hafnium ist eine gute Übereinstimmung der Ergebnisse, vor allem innerhalb der berechneten Werte zu erkennen.

Die Ergebnisse der binären B2-Phase werden in Tabelle 5.4 miteinander verglichen. Bei der B2-Phase sind die Ergebnisse durchaus vergleichbar, wobei auch hier teilweise größere Unterschiede, wie bei C_{11}, erkennbar sind. Dieser Trend sollte auch für die restlichen berechneten Strukturen gelten, und so können auch diese Werte mit Bedacht auf die möglichen Ungenauigkeiten als Input für weitere Berechnungen verwendet werden. Es sei darauf hingewiesen, dass durch die wesentlich längeren Rechenzeiten für diese Systeme eine geringere Anzahl an Dehnungspunkten berechnet worden und dadurch höhere Ungenauigkeiten möglich sind.

Tabelle 5.2: Vergleich von unterschiedlichen Ergebnissen bezüglich der elastischen Konstanten in [GPa] und der Gitterparameter in [a.u.] von Titanium.

	diese Arbeit [a]	experimentell [50, 36]	theoretisch[b]	theoretisch[c]
a	5.57	5.58	5.55	-
c/a	1.576	1.588	1.583	-
C_{11}	175.9	176.1	175.0	174.8
C_{12}	86.1	86.9	82.6	87.5
C_{13}	72.4	68.3	74.7	80.0
C_{33}	194.8	190.5	196.0	189.5
C_{44}	46.1	50.8	41.8	40.9
B	113.2	110.0	112.1	112.8

[a] : exciting [3]
[b] : PAW-Methode, GGA nach Perdew, Burke und Ernzerhof [48]
[c]: PAW-Methode , GGA nach Perdew und Wang [49]

Für die B19'-Phase stehen keine experimentellen Ergebnisse zur Verfügung, da es nicht möglich ist, Einkristalle dieser Struktur herzustellen. Also ist nur ein Vergleich mit theoretisch ermittelten Werten möglich. Diese sind in Tabelle 5.5 zusammengefasst.

Die Vergleichswerte aus der Literatur [53, 54] für die B19'-Phase wurden mit β als monokliner Winkel bestimmt. Dadurch unterscheidet sich die Form der Matrix der elastischen Konstanten. Diese muss zum Vergleich erst transformiert werden. Diese Tasachen und, dass die Konstanten in einem monoklinen System stark von der maximalen Dehnung und vor allem von der Anzahl der Dehnungspunkten abhängt, kann man also die Werte nur bedingt vergleichen, sondern nur Trends in den Konstanten erkennen. Dieser ist im allgemeinen gegeben wobei sich jedoch einige stark divergieren. Zur Qualität der vorgestellten Resultate kann also allgemein gesagt werden, dass sie als Anhaltswerte für die tatsächlichen elastischen Konstanten eingesetzt werden können, und größtenteils eine gute Übereinstimmung mit experimentell ermittelten Konstanten zeigen.

5.2 Zusammenfassung und Ausblick

In dieser Arbeit wurden zunächst die elastischen Konstanten der elementaren Festkörper für Nickel, Titanium und Hafnium bestimmt. Die Ergebnisse dieser Berechnungen sind großteils gut konvergiert und vergleichbar mit anderen in der Literatur vorhandenen Ergebnissen. Dasselbe gilt auch für die Hochtemperaturphase einer stöchiometrischen Nickel-Titanium-Legierung. Im Anschluss daran wurden zwei ternäre NiTiHf-Legierungen betrachtet mit dem Ziel, den Einfluss

Tabelle 5.3: Vergleich von unterschiedlichen Ergebnissen bezüglich der elastischen Konstanten in [GPa] und der Gitterparameter in [a.u.] von Hafnium.

	diese Arbeit[a]	experimentell [50]	theoretisch[b]	theoretisch[c]
a	6.029	6.03	-	6.051
c/a	1.604	1.581	-	1.582
C_{11}	185.3	190.1	183.6	183.0
C_{12}	74.1	74.5	71.1	70.2
C_{13}	69.4	65.5	71.1	68.2
C_{33}	198.9	204.4	197.9	192.5
C_{44}	52.3	60.0	51.1	52.8
B	111.2	110.6	109.1	108.0

[a] : exciting [3]
[b] : PAW-Methode , GGA nach Perdew und Wang [49]
[c]: PAW-Methode , GGA nach Perdew, Burke und Ernzerhof [51]

Tabelle 5.4: Vergleich von unterschiedlichen Ergebnissen bezüglich der elastischen Konstanten der B2 Phase in [GPa] und des Gitterparameter a in [a.u.].

	diese Arbeit[a]	experimentell [40, 52]	theoretisch[b]	theoretisch[c]
a	5.66	5.69	5.69	5.70
C_{11}	176.2	162.4	170.5	183
C_{12}	137.5	129.2	145.5	146
C_{44}	34.3	34.8	39.8	46
B	152.8	140.3	153.9	159

[a] : exciting [3]
[b] : PAW-Methode, GGA nach Perdew, Burke und Ernzerhof [48]
[c]: FPLAPW-Methode, GGA nach Perdew, Burke und Ernzerhof [53]

der Hf-Konzentration auf die elastischen Konstanten zu untersuchen. Dazu wurden zwei geordente Ni-Ti-Hf Verbindungen mit 12,5- und 17 Prozent Hf-Gehalt und insgesamt 16 beziehungsweise 12 Atomen pro Zelle berechnet. Daraus kann eine Tendenz des Einflusses dieser Konzentration auf die elastischen Konstanten abgelesen werden, welche in Abbildung 5.1 dargestellt ist.

Hier sieht man die Entwicklung der elastischen Konstanten mit der Hafniumkonzentration. Zunächst ist erkennbar, dass die beiden ternären Legierungen sehr ähnliche Ergebnisse liefern. Des weiteren beeinflussen die Hafniumatome die verschiedenen elastischen Konstanten in unterschiedlicher Weise. So werden einige Konstanten erhöht, wie zum Besipiel C_{11} und andere kleiner, C_{66}. Manche der Konstanten zeigen ein Extremum zwischen reinem Nickel-Titanium um reinem Nickel-Hafnium. Das heißt also, je nach Art der Verformung wirken die substi-

Tabelle 5.5: Vergleich von unterschiedlichen Ergebnissen bezüglich der elastischen Konstanten der B19' Phase in [GPa] und der Gitterparameter in [a.u.].

	diese Arbeit[a]	theoretisch[b]	theoretisch[c]	theoretisch[d]
a	8.91	8.91	8.84	8.85
b	5.61	5.56	5.51	5.56
c	7.45	7.67	7.70	7.63
γ	99.4°	100.6°	98°	97.8°
C_{11}	236	226	212	200
C_{12}	121	115	107	99
C_{13}	109	121	125	125
C_{16}	23	28	-1	4
C_{22}	195	186	249	223
C_{23}	139	131	129	129
C_{26}	9	-3	15	17
C_{33}	241	239	245	241
C_{36}	-1	-6	-3	-9
C_{44}	80	80	86	77
C_{45}	-2	-3	-4	-4
C_{55}	93	86	87	76
C_{66}	30	23	66	21
B	157	154	159	152

[a] : exciting [3]
[a] : Wien2k [27]
[c] : FPLAPW-Methode, GGA nach Perdew, Burke und Ernzerhof [53]
[d]: PAW-Methode, GGA nach Perdew, Burke und Ernzerhof [54]

tuierten Atome unterschiedlich auf die Deformation. Diese kann durch die Hafniumatome sowohl erleichtert als auch behindert werden. Betrachtet man die Strukturen der beiden Legierungen, so kann man erkennen das die Hafniumatome Schichten im Kristall bilden. C_{66} ist von einer Scherungsdeformation abhängig und C_{11} hauptsächlich von Normaldeformationen. Die Scherungsdeformation welche die elastische Konstante C_{66} in einem monoklinen System beeinflusst ist δ_{13}. In Abbildung 5.2 ist zu erkennen, dass diese Scherung normal zu den Schichten ist, die die Hafniumatome bilden. Es wird also eine solche Scherung durch die Zulegierung von Hafnium erleichtert.

Man kann also sagen, dass die Tendenzen die erkennbar sind im Rahmen der oben erwähnten Genauigkeit durchaus als richtig angenommen werden können.

Um den Einfluss der Hf-Konzentration genauer studieren zu können, wären Berechnungen für weitere ternäre NiTiHf-Strukturen erforderlich. Des Weiteren kann die Genauigkeit der Ergebnisse durch eine erhöhte Anzahl an Dehnungs-

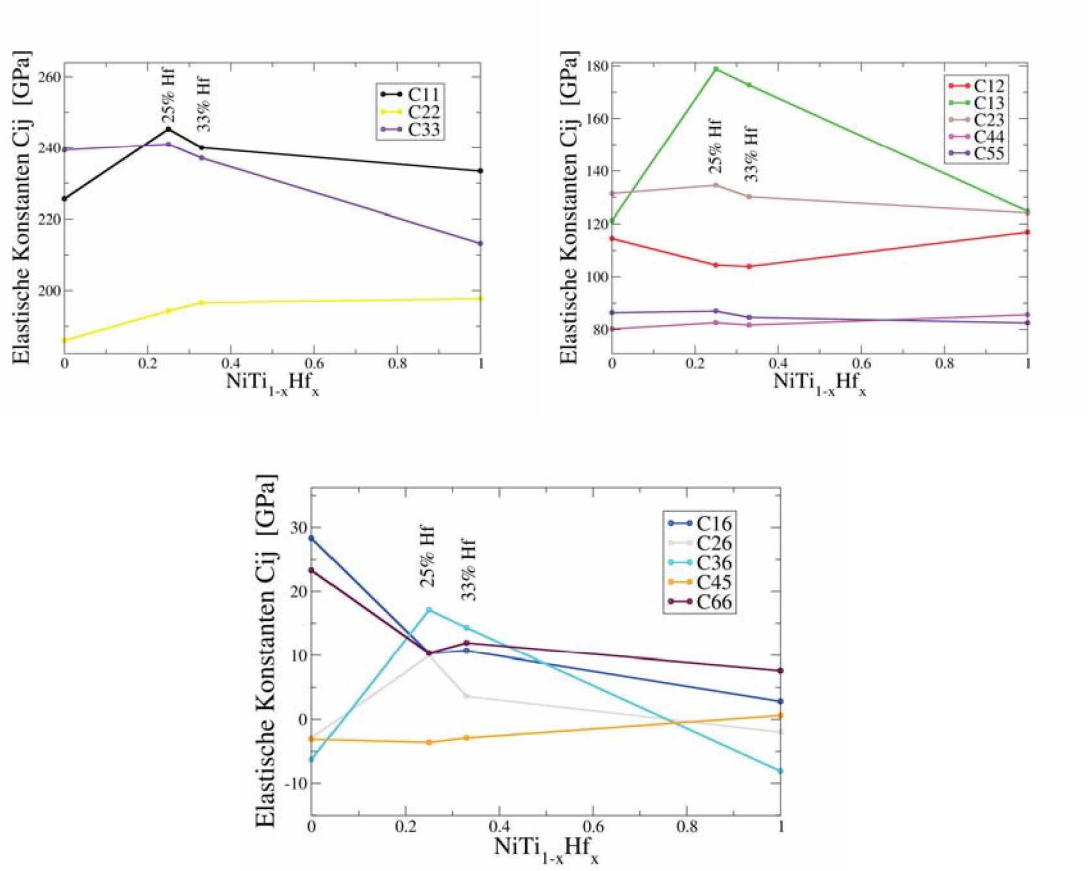

Abbildung 5.1: Einfluss der Hafniumkonzentration auf die elastischen Konstanten. Die NiHf-Daten stammen von Herrn Rostam Golesorkhtabar.

punkten pro Deformation einer elastischen Konstantenberechnung erhöht werden. Es ist jedoch zu beachten, dass pro Dehnung in etwa 200 Berechnungsstunden auf einem Prozessor vonnöten sind. So könnte ein exakteres Bild des Einflusses des Hafniumgehalts ermittelt werden.

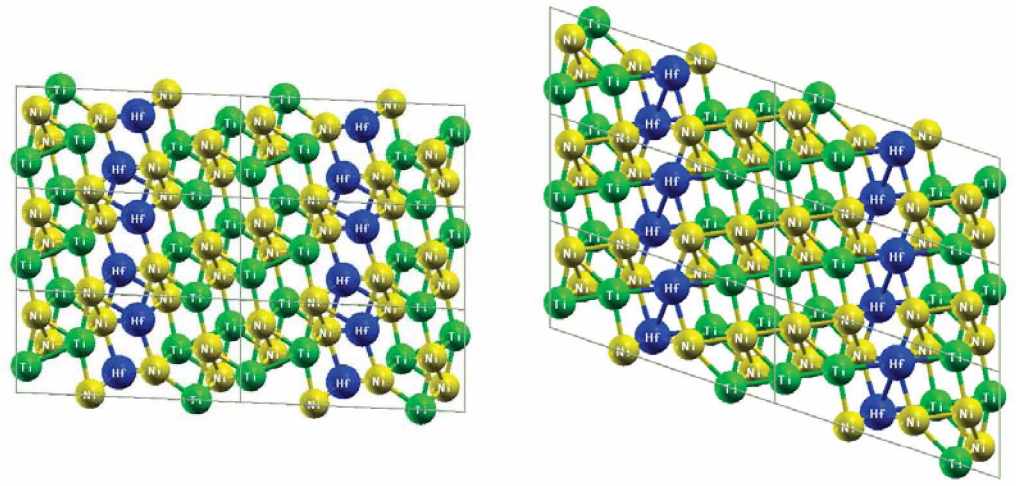

Abbildung 5.2: Darstellung von zwei unterschiedlich stark deformierten Kristallen nach Derformationsart δ_{13} für eine monokline Einheitszelle.

Literaturverzeichnis

[1] Melton Duerig. Applications of shape memory in the usa. *New Materials and Processes for the Future (eds.) N. Igata, I. Kimpara et al.*, pages 195–200, 1989.

[2] Michael Kaack. *Elastische Eigenschaften von NiTi-Formgedächtnis-Legierungen.* Ph. D. thesis, Ruhr-Universität Bochum, Bochum, February 2002.

[3] C. Ambrosch-Draxl, S. Sagmeister, Ch. Meisenbichler, and J. Spitaler. Exciting code, 2009.

[4] J. K. Dewhurst, S. Sharma, and C. Ambrosch-Draxl. Code development under the RT network EXCITING funded by the EU, contract HPRN-CT-2002-00317, 2005.

[5] P. Blaha, K. Schwarz, G. K. H. Madsen, D. Kvasnicka, and J. Luitz. WIEN2k, An Augmented Plane Wave + Local Orbital Program for Calculating Crystal Properties. Technical report, Vienna University of Technology, Vienna, 2001.

[6] S Cottenier. Density functional theory and the family of (l)apw-methods: a step-by-step introduction. *Computer Physics Reports*, 29(L):2005, 2004.

[7] M. Born and R. Oppenheimer. Zur Quantentheorie der Molekeln. *Ann. Physik*, 84:457, 1927.

[8] D. R. Hartree. *Phys. Rev.*, 24:89, 1928.

[9] P. Hohenberg and W. Kohn. Inhomogeneous electron gas. *Phys. Rev.*, 136:B864, 1964.

[10] W. Kohn and L. J. Sham. Quantum density oscillations in an inhomogeneous electron gas. *Phys. Rev.*, 137(6A):A1697–A1705, Mar 1965.

[11] M. Methfessel and A. T. Paxton. High-precision sampling for Brillouin-zone integration in metals. *Phys. Rev. B*, 40:3616, 1989.

[12] W. Kohn and L. J. Sham. Self-consistent equations including exchange and correlation effects. *Phys. Rev.*, 140(4A):A1133–A1138, Nov 1965.

[13] N. Argaman and G. Makov. Density functional theory: An introduction. *American Journal of Physics*, 68:69–79, January 2000.

[14] J. P. Perdew, K. Burke, and M. Ernzerhof. Generalized Gradient Approximation Made Simple. *Phys. Rev. Lett.*, 77:3865, 1996.

[15] O. K. Andersen. Linear methods in band theory. *Phys. Rev. B*, 12:3060, 1975.

[16] Peter Schwerdtfeger. The pseudopotential approximation in electronic structure theory. *ChemPhysChem*, 2011.

[17] J. C. Slater. Wave functions in a periodic potential. *Phys. Rev.*, 51:846, 1937.

[18] F. Bloch. Ueber die quantenmechanik der elektronen in kristallgittern. *Z. Phys.*, 12:47–88, 1928.

[19] C Ambrosch-Draxl. Augmented planewave methods. *Phys. Scr.*, T109:48–53, 2004.

[20] D. Singh and H. Krakauer. H-point phonon in molybdenum: Superlinearized augmented-plane-wave calculations. *Phys. Rev. B*, 43(2):1441–1445, Jan 1991.

[21] D. Singh. *Phys. Rev. B*, 43:6388, 1991.

[22] E Sjöstedt, L Nordström, and D.J Singh. An alternative way of linearizing the augmented plane-wave method. *Solid State Communications*, 114(1):15 – 20, 2000.

[23] Stöckel. Metalle erinnern sich. *Bild der Wissenschaft*, 2/1990:15–20, 1990.

[24] Stöckel. Formgedaechtniseffekt von nickel-titan legierungen. *Feinwerktechnik Messtechnik 95*, 5:332–334, 1987.

[25] E. Unterweger A. C. Kneissl, M. Bruncko, G. Lojen, K. Mehrabi, and H. Scherngell. Microstructure and properties of niti and cualni shape memory alloys. *Metalurgija - Journal of Metallurgy*, 14:89–100, 2011.

[26] P. Nash. *Phase Diagram of Binary Nickel Alloys*. ASM International, 1991.

[27] Rostam Golesorkhtabar, Pasquale Pavone, Jürgen Spitaler, Peter Puschnig, and Claudia Ambrosch-Draxl. Elastic: A universal tool for calculating the second-order elastic constants from first principles. *Computer Physics Communications*, to be submitted 2011.

[28] Hao Wang and Mo Li. *Ab initio* calculations of second-, third-, and fourth-order elastic constants for single crystals. *Phys. Rev. B*, 79:224102, Jun 2009.

[29] W. F. Perger. First-principles calculation of second-order elastic constants and equations of state for lithium azide, lin3, and lead azide, pb(n3)2. *International Journal of Quantum Chemistry*, 110(10):1916–1922, 2010.

[30] D.A. Porter and K.E. Easterling. *Phase transformations in metals and alloys.* Nelson Thornes, 1992.

[31] F. D. Murnaghan. *Proc. Nat. Acad. Sci. U.S.A.*, 3:244, 1944.

[32] Francis Birch. Finite elastic strain of cubic crystals. *Phys. Rev.*, 71(11):809–824, Jun 1947.

[33] W. Voigt. *Lehrbuch der Kristallphysik.* Teubner, 1928.

[34] A Taylor. *J. Inst. Metals*, 77:585, 1950.

[35] A. Kokalj. Computer graphics and graphical user interfaces as tools in simulations of matter at the atomic scale. *Comput. Mater. Sci.*, 28:155, 2003. Code available from http://www.xcrysden.org/.

[36] R. R. Pawar and V. T. Deshpande. The anisotropy of the thermal expansion of α-titanium. *Acta Crystallographica Section A*, 24(2):316–317, Mar 1968.

[37] R. B. Russell. On the zr-hf system. *Journal of Applied Physics*, 24(2):232–233, 1953.

[38] John P. Perdew, Adrienn Ruzsinszky, Gábor I. Csonka, Oleg A. Vydrov, Gustavo E. Scuseria, Lucian A. Constantin, Xiaolan Zhou, and Kieron Burke. Restoring the density-gradient expansion for exchange in solids and surfaces. *Phys. Rev. Lett.*, 100:136406, Apr 2008.

[39] R. Gaudoin and W. M. C. Foulkes. *Ab initio* calculations of bulk moduli and comparison with experiment. *Phys. Rev. B*, 66:052104, Aug 2002.

[40] Mehrdad Zarinejad, Yong Liu, and Timothy J. White. The crystal chemistry of martensite in nitihf shape memory alloys. *Intermetallics*, 16(7):876 – 883, 2008.

[41] J. M. Sanchez, F. Ducastelle, and D. Gratias. Generalized cluster description of multicomponent systems. *Physica A: Statistical and Theoretical Physics*, 128(1-2):334 – 350, 1984.

[42] Jürgen Spitaler. (to be published).

[43] Chris G. Van de Walle, D. B. Laks, G. F. Neumark, and S. T. Pantelides. First-principles calculations of solubilities and doping limits: Li, na, and n in znse. *Phys. Rev. B*, 47(15):9425–9434, Apr 1993.

[44] Monodeep Chakraborty, Jürgen Spitaler, Peter Puschnig, and Claudia Ambrosch-Draxl. Atat@wien2k: An interface for cluster expansion based on the linearized augmented planewave method. *Comp. Phys. Comm.*, 181:913–920, 2010.

[45] L. Vegard. Die konstitution der mischkristalle und die raumfüllung der atome. *Zeitschrift für Physik A Hadrons and Nuclei*, 5:17–26, 1921. 10.1007/BF01349680.

[46] P.P Singh and Munish Kumar. Temperature dependence of bulk modulus and second-order elastic constants. *Physica B: Condensed Matter*, 344(1-4):41 – 51, 2004.

[47] C. Kittel. *Introduction to solid state physics*. Wiley, 2005.

[48] Zhao-Yi Zeng, Cui-E Hu, Ling-Cang Cai, Xiang-Rong Chen, and Fu-Qian Jing. First-principles determination of the structure, elastic constant, phase diagram and thermodynamics of niti alloy. *Physica B: Condensed Matter*, 405(17):3665 – 3672, 2010.

[49] S.L. Shang, A. Saengdeejing, Z.G. Mei, D.E. Kim, H. Zhang, S. Ganeshan, Y. Wang, and Z.K. Liu. First-principles calculations of pure elements: Equations of state and elastic stiffness constants. *Computational Materials Science*, 48(4):813 – 826, 2010.

[50] E. S. Fisher and C. J. Renken. Single-crystal elastic moduli and the hcp $\rightarrow$ bcc transformation in ti, zr, and hf. *Phys. Rev.*, 135:A482–A494, Jul 1964.

[51] YanJun Hao, Jun Zhu, Lin Zhang, HaiSheng Ren, and JianYing Qu. Structure phase transition and elastic properties of hafnium: First-principles study. *Philosophical Magazine Letters*, 91(1):61–69, 2011.

[52] O. Mercier, K. N. Melton, G. Gremaud, and J. Hagi. Single-crystal elastic constants of the equiatomic NiTi alloy near the martensitic transformation. *Journal of Applied Physics*, 51(3):1833–1834, 1980.

[53] N. Hatcher, O. Yu. Kontsevoi, and A. J. Freeman. Role of elastic and shear stabilities in the martensitic transformation path of niti. *Phys. Rev. B*, 80:144203, Oct 2009.

[54] M.F.-X. Wagner and W. Windl. Lattice stability, elastic constants and macroscopic moduli of niti martensites from first principles. *Acta Materialia*, 56(20):6232 – 6245, 2008.

Anhang A

Programm-Code des Optimierungsprogramms

A.1 Setup

```
#!/usr/bin/python
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%#
#%%%——— Cell_Optimization_Setup_exciting ————%%%#
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%#
#
# AUTHORS:
# Daniel Lueftner and Rostam Golesorkhtabar
# daniel.lueftner@stud.unileoben.ac.at
#
# DATE:
# Mon Feb 21 16:06:41 CET 2011
#
# SYNTAX:
# python Optimization_Setup_exciting.py
#        Optimization_Setup_exciting
#--------------------------------------------------------

from lxml  import etree as ET
from sys   import stdin
from numpy import *
import numpy as np
import subprocess
import os.path
import shutil
```

```
import math
import sys
import os

#%%%%%%%%%%%%%%— Dictionaries —%%%%%%%%%%%%%%%%%%%%%%%#
Def_matrix = {
"VOL":                        "[ 1+eps   0        0       ]\
\n                             [ 0       1+eps    0       ]\
\n                             [ 0       0        1+eps   ]\n"}

StructDic = {'1':'Cubic','2':'Hexagonal','3':'Rhombohedral
    ','4':'Tetragonal','5':'Orthorhombic','6':'Monoclinic',
    '7':'Triclinic'}

#%%%%%%%%%%%%%%%— Check input file exist —%%%%%%%%%%%%%#
INF = raw_input("\n>>>> Please enter the exciting input
    file name: ")
while (os.path.exists(INF)==False):
     print("\n     ... Oops ERROR: There is NO file named "+
          INF +" !?!?!?     \n")
     INF = raw_input("\n>>>> Please enter the correct input
          file name or press Ctrl + C to leave programm: ")
#———————————————————————————————————————————————————————————————

#%%— Calculate Space-group number and classify it —%%#
os.system('xsltproc ~/bin/exciting2sgroup.xsl '+ INF +' >
    sgroup.in ; sgroup sgroup.in > sgroup.out ; rm -f
    sgroup.in')
for line in open("sgroup.out", "r"):
     if line.find("Number and name of space group:") >= 0 :
          SGN=int(float(line.split()[6])); break

if   (  1 <= SGN and SGN <=   2): n = 7 ; m = 6  #
    Triclinic
elif (  3 <= SGN and SGN <=  15): n = 6 ; m = 4  #
    Monoclinic
elif ( 16 <= SGN and SGN <=  74): n = 5 ; m = 3  #
    Orthorhombic
elif ( 75 <= SGN and SGN <= 142): n = 4 ; m = 2  #
    Tetragonal
elif (143 <= SGN and SGN <= 167): n = 3 ; m = 2  #
    Rhombohedral
elif (168 <= SGN and SGN <= 194): n = 2 ; m = 2  #
```

```
   Hexagonal
elif (195 <= SGN and SGN <= 230): n = 1 ; m = 1  # Cubic
else: sys.exit("\n      ... Oops ERROR: WRONG Space Group
   Number !?!?!?     \n")
print '\n      '+line.strip()
print '      '+ StructDic[str(n)] +' structure in
   crystallography classification.'
nloop = 0
#%%%--- Read optimization type ---%%%%%%%%%#
if (n==1):
    dirn = 'VOL'
if (n==2 or n==3 or n==4):
    print "\n      Which optimization you like to apply?"
    print    "      1 ... Change in volume  "
    print    "      2 ... Change of c/a ratio with constant
       volume  "
    print    "      3 ... Complete Optimization: volume and
       c/a  "
    num = input(">>>> Please choose '1' , '2' or '3': ")
    while (num != 1 and num != 2 and num != 3):
        print("\n     ... Oops ERROR: Please choose between
            '1' or '2' \n")
        print "\n      Which optimization you like to apply
           ?"
        print    "      1 ... Change in volume  "
        print    "      2 ... Change of c/a ratio with
           constant volume  "
        print    "      3 ... Complete Optimization: volume
           and c/a  "
        num = input(">>>> Please choose '1' , '2' or '3':
           ")
    if (num == 1 ): dirn = 'VOL'
    if (num == 2 ): dirn = 'COA'
    if (num == 3 ):
        dirn = 'COMPLETE_' + StructDic[str(n)]
        dirnames = ['VOL','COA']
        nloop = input(">>>> Please enter the number of
           loops ")
if (n==5):
    print "\n      Which optimization you like to apply?"
    print    "      1 ... Change in volume  "
    print    "      2 ... Change of b/a ratio with constant
       volume  "
```

```
    print "      3 ... Change of c/a ratio with constant
       volume "
    print "      4 ... Complete Optimization: volume , b/
       a and c/a "
    num = input(">>>> Please choose '1' , '2' , '3' or
       '4': ")
    while (num != 1 and num != 2 and num != 3 and num !=
       4):
        print("\n     ... Oops ERROR: Please choose between
            '1' , '2' or '3' \n")
        print "\n     Which optimization you like to apply
           ?"
        print "      1 ... Change in volume "
        print "      2 ... Change of b/a ratio with
           constant volume "
        print "      3 ... Change of c/a ratio with
           constant volume "
        print "      4 ... Complete Optimization: volume
           , b/a and c/a "
        num = input(">>>> Please choose '1' , '2' , '3'
           or '4': ")
    if (num == 1 ): dirn = 'VOL'
    if (num == 2 ): dirn = 'BOA'
    if (num == 3 ): dirn = 'COA'
    if (num == 4 ):
        dirn = 'COMPLETE_' + StructDic[str(n)]
        dirnames = ['VOL','BOA','COA']
        nloop = input(">>>> Please enter the number of
           loops ")
if (n==6):
    print "\n     Which optimization you like to apply?"
    print "      1 ... Change in volume    "
    print "      2 ... Change of b/a ratio with constant
       volume    "
    print "      3 ... Change of c/a ratio with constant
       volume    "
    print "      4 ... Change of the Gamma angle   "
    print "      5 ... Complete Optimization: volume , b/
       a , c/a and GAMMA "
    num = input(">>>> Please choose '1' , '2' , '3' , '4'
       or '5': ")
    while (num != 1 and num != 2 and num != 3 and num != 4
        and num != 5):
```

```
        print("\n     ... Oops ERROR: Please choose between
            '1' , '2' , '3' or '4' \n")
        print "\n     Which optimization you like to apply
           ?"
        print    "     1 ... Change in volume  "
        print    "     2 ... Change of b/a ratio with
           constant volume  "
        print    "     3 ... Change of c/a ratio with
           constant volume  "
        print    "     4 ... Change of the Gamma angle  "
        print    "     5 ... Complete Optimization: volume
           , b/a , c/a and GAMMA "
        num = input(">>>> Please choose '1' , '2' , '3' ,
           '4' or '5': ")
    if (num == 1 ): dirn = 'VOL'
    if (num == 2 ): dirn = 'BOA'
    if (num == 3 ): dirn = 'COA'
    if (num == 4 ): dirn = 'GAMMA'
    if (num == 5 ):
        dirn = 'COMPLETE_' + StructDic[str(n)]
        dirnames = ['VOL','BOA','COA','GAMMA']
        nloop = input(">>>> Please enter the number of
           loops ")
if (n==7):
    print "\n     Which optimization you like to apply?"
    print    "     1 ... Change in volume    "
    print    "     2 ... Change of b/a ratio with constant
       volume    "
    print    "     3 ... Change of c/a ratio with constant
       volume    "
    print    "     4 ... Change of the Alpha angle  "
    print    "     5 ... Change of the Beta angle   "
    print    "     6 ... Change of the Gamma angle  "
    print    "     7 ... Complete Optimization: volume , b/
       a , c/a and GAMMA "
    num = input(">>>> Please choose '1' , '2' , '3' , '4'
       , '5' , '6' or '7': ")
    while (num != 1 and num != 2 and num != 3 and num != 4
        and num!= 5 and num != 6 and num != 7):
        print("\n     ... Oops ERROR: Please choose between
            '1' , '2' , '3' , '4' , '5' or '6' \n")
        print "\n     Which optimization you like to apply
           ?"
```

```
        print "      1 ... Change in volume "
        print "      2 ... Change of b/a ratio with
           constant volume "
        print "      3 ... Change of c/a ratio with
           constant volume "
        print "      4 ... Change of the Alpha angle "
        print "      5 ... Change of the Beta angle "
        print "      6 ... Change of the Gamma angle "
        print "      7 ... Complete Optimization: volume
           , b/a , c/a and GAMMA "
        num = input(">>>> Please choose '1' , '2' , '3' ,
           '4' , '5' , '6' or '7': ")
    if (num == 1 ): dirn = 'VOL'
    if (num == 2 ): dirn = 'BOA'
    if (num == 3 ): dirn = 'COA'
    if (num == 4 ): dirn = 'ALPHA'
    if (num == 5 ): dirn = 'BETA'
    if (num == 6 ): dirn = 'GAMMA'
    if (num == 7 ):
        dirn = 'COMPLETE_' + StructDic[str(n)]
        dirnames = ['VOL','BOA','COA','ALPHA','BETA','
           GAMMA']
        nloop = input(">>>> Please enter the number of
           loops ")
#%%%%%%%%-- Read maximum Physical strain --%%%%%%%%%%#
if (dirn == 'VOL' or dirn == 'BOA' or dirn == 'COA' or
   dirn == 'COMPLETE_'+ StructDic[str(n)]):
    mdr = input("\n>>>> Please enter maximum physical
       strain: ")
    if (1 < mdr or mdr < 0): sys.exit("\n      ... Oops
       ERROR: Maximum Physical strain is out of range
       !!!!!!      \n")
    mdr = round(mdr, 4)
    print "      Maximum Physical strain is " +str(mdr) +"\
       n"

if (dirn == 'ALPHA' or dirn == 'BETA' or dirn == 'GAMMA'
   or dirn == 'COMPLETE_'+ StructDic[str(n)] and n == 6 or
    n == 7 ):
    angle = input ("\n>>>> Please enter the maximum change
       of the angle in degrees: ")
    if (90 < angle or angle < 0): sys.exit("\n      ...
       Oops ERROR: Maximum change of angle is out of range
```

```
        !!!!!!      \n")
    angle = round(angle, 1)
    print "      Maximum change of angle is " +str(angle) +
       " degrees \n"
#-------------------------------------------------------------

#%%%%--- Read Number of distorted structures ---%%%%%#
point_number=input(">>>> Please enter the number of
   distorted structures for each deformation type [odd
   number > 4]: ")
point_number=int(abs(point_number))
if (point_number < 5):  sys.exit("\n      ... Oops ERROR:
   Number of distorted structures < 5 !!!!!!     \n")
if (99 < point_number): sys.exit("\n      ... Oops ERROR:
   Number of distorted structures > 99 !!!!!!    \n")
if (mod(point_number,2)==0): point_number=point_number+1
print "      Number of distorted structures is "+ str(
   point_number) +"\n"
ptn = int((point_number-1)/2)
if (dirn == 'VOL' or dirn == 'BOA' or dirn == 'COA' or
   dirn == 'COMPLETE_'+ StructDic[str(n)]):
    if (mdr/ptn <= 0.0001):
        sys.exit("\n      ... Oops ERROR: Strain distances
           are less than 0.0001 \
                \n                          Choose larger
                   maximum Physical strain or less
                   number of distorted structures.\n")
#-------------------------------------------------------------

#%%%%%%%%%%%%%--- Read input file ---%%%%%%%%%%%%%%%%%%%%#
inobj = open(INF,"r"); doc = ET.parse(inobj); root = doc.
   getroot()
#-------------------------------------------------------------

#%%%--- Read scale, stretch, and base vectors from input
   file ---%#
scale=map(float,doc.xpath('/input/structure/crystal/@scale
   '))
if (scale==[]): scale=[1]

stretchstr=doc.xpath('/input/structure/crystal/@stretch')
if (stretchstr==[]): stretch=[1.,1.,1.]
else: stretch=np.array(map(float,stretchstr[0].split()))
```

```
basevectsn = doc.xpath('//basevect/text()'); bv=[]
for basevect in basevectsn:
    bv.append(map(float,basevect.split()))
M_old = np.array(bv); D = np.linalg.det(M_old)
V0= abs(stretch[0]*stretch[1]*stretch[2]*scale[0]**3*D)
#-----------------------------------------------------------

#%%%%%%%%%-- Directory management --%%%%%%%%%%%%%%%%%%%%%#
if (dirn == 'VOL'):
    if (os.path.exists('VOL_old')): shutil.rmtree('VOL_old
        ')
    if (os.path.exists('VOL')): os.rename('VOL', 'VOL_old'
        )
if (dirn == 'COA'):
    if (os.path.exists('COA_old')): shutil.rmtree('COA_old
        ')
    if (os.path.exists('COA')): os.rename('COA', 'COA_old'
        )
if (dirn == 'BOA'):
    if (os.path.exists('BOA_old')): shutil.rmtree('BOA_old
        ')
    if (os.path.exists('BOA')): os.rename('BOA', 'BOA_old'
        )
if (dirn == 'ALPHA'):
    if (os.path.exists('ALPHA_old')): shutil.rmtree('
       ALPHA_old')
    if (os.path.exists('ALPHA')): os.rename('ALPHA', '
       ALPHA_old')
if (dirn == 'BETA'):
    if (os.path.exists('BETA_old')): shutil.rmtree('
       BETA_old')
    if (os.path.exists('BETA')): os.rename('BETA', '
       BETA_old')
if (dirn == 'GAMMA'):
    if (os.path.exists('GAMMA_old')): shutil.rmtree('
       GAMMA_old')
    if (os.path.exists('GAMMA')): os.rename('GAMMA', '
       GAMMA_old')
if (dirn == 'COMPLETE_'+ StructDic[str(n)]):
    if (os.path.exists('COMPLETE_'+ StructDic[str(n)] +'
       _old')): shutil.rmtree('COMPLETE_'+ StructDic[str(n
       )] +'_old')
```

```
    if (os.path.exists('COMPLETE_'+ StructDic[str(n)] )):
       os.rename('COMPLETE_'+ StructDic[str(n)] , '
       COMPLETE_'+ StructDic[str(n)] +'_old')
os.mkdir(dirn); os.chdir(dirn)
#-------------------------------------------------------------
#%%%--- Make INFO_ElaStic file and write into it ---%%%#
if (dirn == 'VOL' or dirn == 'BOA' or dirn == 'COA' or
   dirn == 'COMPLETE_'+ StructDic[str(n)] and n>=1 or n<=5
    ):
    INFO=open('INFO_FILE' ,'w')
    print >>INFO, "Space-group number                =", SGN
       ,\
                  "\nStructure type                    =",
                     StructDic[str(n)],\
                  "\nMaximum Physical strain          =",
                     mdr,\
                  "\nNumber of distorted structures   =",
                     point_number,\
                  "\nMode of optimization             =",
                     str(dirn) ,\
                  "\nNumber of loops                  =",
                     nloop
    INFO.close
if(dirn == 'ALPHA' or dirn == 'BETA' or dirn == 'GAMMA'):
    INFO=open('INFO_FILE' ,'w')
    print >>INFO, "Space-group number                =", SGN
       ,\
                  "\nStructure type                    =",
                     StructDic[str(n)],\
                  "\nMaximum change of angle          =",
                     angle,\
                  "\nNumber of distorted structures   =",
                     point_number,\
                  "\nMode of optimization             =",
                     str(dirn) ,\
                  "\nNumber of loops                  =",
                     nloop
    INFO.close
if (dirn == 'COMPLETE_'+ StructDic[str(n)] and n>=6):
    INFO=open('INFO_FILE' ,'w')
    print >>INFO, "Space-group number                =", SGN
       ,\
                  "\nStructure type                    =",
```

```
                StructDic[str(n)],\
              "\nMaximum Physical strain          =",
                mdr,\
              "\nMaximum change of angle          =",
                angle,\
              "\nNumber of distorted structures   =",
                point_number,\
              "\nMode of optimization             =",
                str(dirn) ,\
              "\nNumber of loops                  =",
                nloop
    INFO.close
#-----------------------------------------------------------------------

#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%#
###------------------------ Structures maker -------------------------###
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%#
if (dirn !=  'COMPLETE_'+ StructDic[str(n)]):
    fdis=open('Cell_Parameters','w')
    cont1=0
    for s in range(-ptn, ptn+1):
        if(dirn == 'VOL' or dirn == 'BOA' or dirn == 'COA'
          ):
            r = mdr*s/ptn
            if (s==0): r=0.0001
            eps = r
            DtDic={\
            'VOL':[ (1.+eps, 0., 0.), (0., 1.+eps, 0.)
                ,(0., 0., 1+eps)],\
            'BOA':[(1./(1+eps)**(1./2), 0., 0.), (0., 1.+
               eps, 0.),(0., 0., 1./(1+eps)**(1./2))],\
            'COA':[(1./(1+eps)**(1./2), 0., 0.), (0.,
               1./(1+eps)**(1./2), 0.),(0., 0., 1+eps)]}
            if(dirn == 'VOL'):
                M_eps = np.array(DtDic[dirn])
                M_new = dot(M_old,M_eps)
            if(dirn == 'BOA'):
                x = math.cos
                y = math.sin
                aold = (M_old[0][0]**2.+M_old[0][1]**2.+
                   M_old[0][2]**2.)**(1./2)
                bold = (M_old[1][0]**2.+M_old[1][1]**2.+
                   M_old[1][2]**2.)**(1./2)
```

```
        cold = (M_old[2][0]**2.+M_old[2][1]**2.+
            M_old[2][2]**2.)**(1./2)
        ao = math.acos((M_old[2][0]*M_old[1][0]+
            M_old[2][1]*M_old[1][1]+M_old[2][2]*
            M_old[1][2])/(cold*bold))
        bo = math.acos((M_old[0][0]*M_old[2][0]+
            M_old[0][1]*M_old[2][1]+M_old[0][2]*
            M_old[2][2])/(aold*cold))
        go = math.acos((M_old[0][0]*M_old[1][0]+
            M_old[0][1]*M_old[1][1]+M_old[0][2]*
            M_old[1][2])/(aold*bold))
        boaold = bold/aold
        boanew = boaold * (1+eps)
        anew = (aold**3.*(boaold/boanew))**(1./3)
        bnew = anew * boanew
        cnew = cold * (anew/aold)
        an = ao
        bn = bo
        gn = go
        M_new = [[anew*((1+2.*x(an)*x(bn)*x(gn)-(x
            (an))**2.-(x(bn))**2.-(x(gn))**2.)
            **0.5/(y(an))),anew*((x(gn)-x(bn)*x(an)
            )/(y(an))),anew*x(bn)],\
              [0.,bnew*y(an),bnew*x(an)],\
              [0.,0.,cnew]]
        M_new = np.array(M_new)
    if(dirn == 'COA'):
        x = math.cos
        y = math.sin
        aold = (M_old[0][0]**2.+M_old[0][1]**2.+
            M_old[0][2]**2.)**(1./2)
        bold = (M_old[1][0]**2.+M_old[1][1]**2.+
            M_old[1][2]**2.)**(1./2)
        cold = (M_old[2][0]**2.+M_old[2][1]**2.+
            M_old[2][2]**2.)**(1./2)
        ao = math.acos((M_old[2][0]*M_old[1][0]+
            M_old[2][1]*M_old[1][1]+M_old[2][2]*
            M_old[1][2])/(cold*bold))
        bo = math.acos((M_old[0][0]*M_old[2][0]+
            M_old[0][1]*M_old[2][1]+M_old[0][2]*
            M_old[2][2])/(aold*cold))
        go = math.acos((M_old[0][0]*M_old[1][0]+
            M_old[0][1]*M_old[1][1]+M_old[0][2]*
```

```
                M_old[1][2])/(aold*bold))
            coaold = cold/aold
            coanew = coaold * (1+eps)
            anew = (aold**3.*(coaold/coanew))**(1./3)
            bnew = bold * (anew/aold)
            cnew = anew * coanew
            an = ao
            bn = bo
            gn = go
            M_new = [[anew*((1+2.*x(an)*x(bn)*x(gn)-(x
                (an))**2.-(x(bn))**2.-(x(gn))**2.)
                **0.5/(y(an))),anew*((x(gn)-x(bn)*x(an)
                )/(y(an))),anew*x(bn)],\
                  [0.,bnew*y(an),bnew*x(an)],\
                  [0.,0.,cnew]]
            M_new = np.array(M_new)
    if(dirn == 'ALPHA' or dirn == 'BETA' or dirn == '
        GAMMA'):
        eps = math.radians(angle)*s/ptn
        if (s==0): eps=0.0001
        aold = (M_old[0][0]**2.+M_old[0][1]**2.+M_old
            [0][2]**2.)**(1./2)
        bold = (M_old[1][0]**2.+M_old[1][1]**2.+M_old
            [1][2]**2.)**(1./2)
        cold = (M_old[2][0]**2.+M_old[2][1]**2.+M_old
            [2][2]**2.)**(1./2)
        ao = math.acos((M_old[2][0]*M_old[1][0]+M_old
            [2][1]*M_old[1][1]+M_old[2][2]*M_old[1][2])
            /(cold*bold))
        bo = math.acos((M_old[0][0]*M_old[2][0]+M_old
            [0][1]*M_old[2][1]+M_old[0][2]*M_old[2][2])
            /(aold*cold))
        go = math.acos((M_old[0][0]*M_old[1][0]+M_old
            [0][1]*M_old[1][1]+M_old[0][2]*M_old[1][2])
            /(aold*bold))
        if(dirn == 'ALPHA'): an = ao+eps
        else: an = ao
        if(dirn == 'BETA'): bn = bo+eps
        else: bn = bo
        if(dirn == 'GAMMA'): gn = go+eps
        else: gn = go
        x = math.cos
        y = math.sin
```

```
            anew = (aold**6*((1+2*x(ao)*x(bo)*x(go)-(x(ao)
                )**2-(x(bo))**2-(x(go))**2)/\
                    (1+2*x(an)*x(bn)*x(gn)-(x(an))**2-(x(bn
                        ))**2-(x(gn))**2)))**(1./6)
            bnew = bold * (anew/aold)
            cnew = cold * (anew/aold)
            M_new = [[anew*((1+2.*x(an)*x(bn)*x(gn)-(x(an)
                )**2.-(x(bn))**2.-(x(gn))**2.)**0.5/(y(an))
                ),anew*((x(gn)-x(bn)*x(an))/(y(an))),anew*x
                (bn)],\
                    [0.,bnew*y(an),bnew*x(an)],\
                    [0.,0.,cnew]]
            for k in range(3):
                for l in range(3):
                    if (abs(M_new[k-1][l-1]) <= 10**(-10))
                        : M_new[k-1][l-1] = 0.
            M_new = np.array(M_new)
        cont1 += 1
        if (cont1<10): dirn_cont1=str(dirn.lower())+"_0"+
            str(cont1)
        else: dirn_cont1=str(dirn.lower())+"_"+str(cont1)
        bsvct=doc.xpath('//crystal/basevect')
        fmt = '%16.10f'
        for j in range(3):
            bdummy = str(fmt%M_new[j,0])+str(fmt%M_new[j
                ,1])+str(fmt%M_new[j,2])+' '
            print >>fdis,'V'+str(j+1)+' ---> '+bdummy
            bsvct[j].text=bdummy
        print >>fdis
        os.mkdir(dirn_cont1); os.chdir(dirn_cont1)
#-------Write structure file-------------------------------
        fileName= dirn_cont1+'.xml'
        outobj=open(fileName,"w")
        outobj.write(ET.tostring(root, method='xml',
                                       pretty_print=True,
                                       xml_declaration=True
                                           ,
                                       encoding='UTF-8'))
        outobj.close()
#----------------------------------------------------------
        os.chdir('../')
    os.system("cp ../" + INF + " old_input.xml")
    os.chdir('../')
```

```
    fdis.close()
#-------------------------------------------------------------------

if (dirn ==  'COMPLETE_'+ StructDic[str(n)]):
    for x in dirnames:
        os.mkdir(x)
        os.chdir(x)
        if (x == 'VOL' or x == 'BOA' or x == 'COA'):
            INFO=open('INFO_FILE' ,'w')
            print >>INFO, "Space-group number
                          =", SGN,\
                          "\nStructure type
                                          =",
                             StructDic[str(n)],\
                          "\nMaximum Physical strain
                                    =", mdr,\
                          "\nNumber of distorted
                             structures  =", point_number
                             ,\
                          "\nMode of optimization
                                       =", str(x)
            INFO.close
        else:
            INFO=open('INFO_FILE' ,'w')
            print >>INFO, "Space-group number
                          =", SGN,\
                          "\nStructure type
                                          =",
                             StructDic[str(n)],\
                          "\nMaximum change of angle
                                    =", angle,\
                          "\nNumber of distorted
                             structures  =", point_number
                             ,\
                          "\nMode of optimization
                                       =", str(x)
            INFO.close

#-------Write structure file-------------------------------------
        os.chdir('../')
    outobj=open('new_input.xml',"w")
    outobj.write(ET.tostring(root, method='xml',
                             pretty_print=True,
```

```
                                    xml_declaration=True,
                                    encoding='UTF-8'))
    outobj.close()
#-------------------------------------------------------------------------
```

A.2 Auswertung

```
#!/usr/bin/python
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%#
#%%%------ Cell_Optimization_Result_exciting ------%%%#
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%#
#
# AUTHORS:
# Daniel Lueftner and Rostam Golesorkhtabar
# daniel.lueftner@stud.unileoben.ac.at
#
# DATE:
# Mon Feb 21 16:06:41 CET 2011
#
# SYNTAX:
# python Optimization_Result_exciting.py
#        Optimization_Result_exciting
#
#-------------------------------------------------------------------------
import os
import sys
import math
import numpy
from pylab import *
from scipy import *
from lxml  import etree as ET
import matplotlib.pyplot as plt
from scipy.optimize import leastsq
```

```
#%%%%%%%%%%%%%%-- Check INFO_Case exists --%%%%%%%%%%%%%%#
bolean=str(os.path.exists('INFO_FILE'))
if(bolean=='False'): sys.exit("\n      ...Oops ERROR: Where
    is INFO_FILE !?!?!?      \n")
#-------------------------------------------------------------

#%%%%%%%%%%%%%%-- Read the INFO_Case file --%%%%%%%%%%%%%%#
INFO=open('INFO_FILE','r')
l1 = INFO.readline(); l2 = INFO.readline(); l3 = INFO.
    readline(); l4 = INFO.readline(); l5 = INFO.readline()
SGN= l1.split()[3]
mdr= l3.split()[4]
point_number = l4.split()[5]
dirn = l5.split()[-1]
if (dirn == 'VOL' or dirn == 'BOA' or dirn == 'COA'):
     mdr= l3.split()[-1]
     mdr=float(mdr)
else:
     angle =  l3.split()[-1]
     angle = float(angle)
SGN=int(float(SGN))
point_number=int(float(point_number))
ptn=int((point_number-1)/2)
INFO.close()
#-------------------------------------------------------------

#%%%%%%%%-- Classify the Space group number --%%%%%%%%#

if    (  1 <= SGN and SGN <=   2): n = 7 ; m = 6  #
    Triclinic
elif (  3 <= SGN and SGN <=  15): n = 6 ; m = 4  #
    Monoclinic
elif ( 16 <= SGN and SGN <=  74): n = 5 ; m = 3  #
    Orthorhombic
elif ( 75 <= SGN and SGN <= 142): n = 4 ; m = 2  #
    Tetragonal
elif (143 <= SGN and SGN <= 167): n = 3 ; m = 2  #
    Rhombohedral
elif (168 <= SGN and SGN <= 194): n = 2 ; m = 2  #
    Hexagonal
elif (195 <= SGN and SGN <= 230): n = 1 ; m = 1  # Cubic
else: sys.exit("\n      ... Oops ERROR: WRONG Space Group
    Number !?!?!?      \n")
```

```
#-------------------------------------------------------------------

#%%%-- Read energies, Volume and calculate strain --%%%#

Ene = []; Vol = []; strain = []
for i in range(1,point_number+1):
    if(i<10): dirn_i=str(dirn.lower())+"_0"+str(i)
    else: dirn_i=str(dirn.lower())+'_' + str(i)
    os.chdir(dirn_i)
    if( os.path.exists('TOTENERGY.OUT') ):
        energyfile = open('TOTENERGY.OUT','r')
        energies = energyfile.readlines()
        energyfile.close()
        energy = float(energies[-1])
    Ene.append(energy)
    if (dirn == 'VOL'):
        volfile = open('INFO.OUT','r')
        for line in volfile:
            if line.find("Unit cell volume") >= 0: volume
              = float(line.split()[-1])
        volfile.close()
        Vol.append(volume)
    s = i-ptn-1
    if (dirn == 'VOL' or dirn == 'BOA' or dirn == 'COA'):
       r = mdr*s/ptn
    else: r = math.radians(angle)*s/ptn
    if(s==0): r=0.0001
    strain.append(r)
    os.chdir('../')
#-------------------------------------------------------------------
inobj = open('old_input.xml',"r"); doc = ET.parse(inobj);
   root = doc.getroot()
scale=map(float,doc.xpath('/input/structure/crystal/@scale
   '))
if (scale==[]): sys.exit("\n      ... Oops ERROR: There is
   NO scale in "+ INF +" file !?!?!?     \n")

stretchstr=doc.xpath('/input/structure/crystal/@stretch')
if (stretchstr==[]): stretch=[1.,1.,1.]
else: stretch=np.array(map(float,stretchstr[0].split()))
basevectsn = doc.xpath('//basevect/text()'); bv=[]
for basevect in basevectsn:
    bv.append(map(float,basevect.split()))
```

```
M_old = np.array(bv); D = np.linalg.det(M_old)
#-------------------------------------------------------------------

#%%--- Make fit (BM) and calculate cell parameters ---%%#
if(dirn == 'VOL'):
    v = np.array(Vol)
    e = np.array(Ene)
    a,b,c = polyfit(v,e,2)
    v0 = -b/(2*a)
    e0 = a*v0**2 + b*v0 + c
    b0 = (2*a*v0)*29407.2
    bP = 1.
    def Birch(parameters,vol):
        E0 = parameters[0]
        B0 = parameters[1]
        BP = parameters[2]
        V0 = parameters[3]
        E = E0+(9./16)*(B0/29407.2)*V0*(((((V0/vol)
            **(2./3))-1.)**3.)*BP+((((V0/vol)**(2/3.))-1.)
            **2.)*(6.-4.*((V0/vol)**(2./3))))
        return E
    def res(p,y,x):
        err = y - Birch(p,x)
        return err
    p0 = [e0, b0, bP, v0]
    p1 = leastsq(res, p0, args=(e, v))
    print "The Bulk modulus is " + str(p1[0][1]) + " GPa"
    print "E_min:" + str(p1[0][0])
    volu = linspace(Vol[0], Vol[-1], 1000)
    plt.plot( volu, Birch(p1[0], volu), '-' , Vol, Ene ,'o
        ')
    plt.title('BM-Fit-test')
    plt.legend(['BM-Fit', 'Data'])
    plt.xlabel('Volume [a.u.^3]')
    plt.ylabel('Energy [Hartree]')
    ax = gca()
    plt.text(0.3,0.8, 'equi_volume    = %1.2f [a.u.^3]' %
       p1[0][3], transform = ax.transAxes)
    plt.text(0.3,0.75,'Bulk modulus =  %1.2f GPa' % p1
       [0][1],  transform = ax.transAxes)
    savefig('BM_FIT.png')
    new_scale = (p1[0][3]/(abs(D)*stretch[0]*stretch[1]*
       stretch[2]))**(1./3)
```

```
    xsc = doc.xpath('//crystal')
    fmt = '%16.10f'
    xsc[0].set("scale",str(fmt%new_scale))

#-------------------------------------------------------------------

#%- Make fit (Polynomial) and calculate cell parameters-%#

else:
    coeffs = numpy.polyfit(strain, Ene, 4)
    strainu = linspace(strain[0], strain[-1], 1000)
    plt.plot(strainu , coeffs[0]*strainu**4 + coeffs[1]*
       strainu**3 + coeffs[2]*strainu**2 + coeffs[3]*
       strainu + coeffs[4] , '-')
    plt.plot(strain , Ene , 'o')
    plt.xlabel('strain ')
    plt.ylabel('Energy [Hartree]')
    savefig('Polynomial_Fit.png')
    le = zeros(6)
    if (dirn == 'BOA'):
        eps = coeffs[3]*(-1)
        print eps
        aold = (M_old[0][0]**2.+M_old[0][1]**2.+M_old
           [0][2]**2.)**(1./2)
        bold = (M_old[1][0]**2.+M_old[1][1]**2.+M_old
           [1][2]**2.)**(1./2)
        cold = (M_old[2][0]**2.+M_old[2][1]**2.+M_old
           [2][2]**2.)**(1./2)
        ao = math.acos((M_old[2][0]*M_old[1][0]+M_old
           [2][1]*M_old[1][1]+M_old[2][2]*M_old[1][2])/(
           cold*bold))
        bo = math.acos((M_old[0][0]*M_old[2][0]+M_old
           [0][1]*M_old[2][1]+M_old[0][2]*M_old[2][2])/(
           aold*cold))
        go = math.acos((M_old[0][0]*M_old[1][0]+M_old
           [0][1]*M_old[1][1]+M_old[0][2]*M_old[1][2])/(
           aold*bold))
        boaold = bold/aold
        boanew = boaold * (1+eps)
        anew = (aold**3.*(boaold/boanew))**(1./3)
        bnew = anew * boanew
        cnew = cold * (anew/aold)
        an = ao
```

```
        bn = bo
        gn = go
        x = math.cos
        y = math.sin
        M_new = [[anew*((1+2.*x(an)*x(bn)*x(gn)-(x(an))
            **2.-(x(bn))**2.-(x(gn))**2.)**0.5/(y(an))),
            anew*((x(gn)-x(bn)*x(an))/(y(an))),anew*x(bn)
            ],\
              [0.,bnew*y(an),bnew*x(an)],\
              [0.,0.,cnew]]
        M_new = numpy.array(M_new)
    elif (dirn == 'COA'):
        eps = coeffs[3]*(-1)
        aold = (M_old[0][0]**2.+M_old[0][1]**2.+M_old
            [0][2]**2.)**(1./2)
        bold = (M_old[1][0]**2.+M_old[1][1]**2.+M_old
            [1][2]**2.)**(1./2)
        cold = (M_old[2][0]**2.+M_old[2][1]**2.+M_old
            [2][2]**2.)**(1./2)
        ao = math.acos((M_old[2][0]*M_old[1][0]+M_old
            [2][1]*M_old[1][1]+M_old[2][2]*M_old[1][2])/(
            cold*bold))
        bo = math.acos((M_old[0][0]*M_old[2][0]+M_old
            [0][1]*M_old[2][1]+M_old[0][2]*M_old[2][2])/(
            aold*cold))
        go = math.acos((M_old[0][0]*M_old[1][0]+M_old
            [0][1]*M_old[1][1]+M_old[0][2]*M_old[1][2])/(
            aold*bold))
        coaold = cold/aold
        coanew = coaold * (1+eps)
        anew = (aold**3.*(coaold/coanew))**(1./3)
        bnew = bold * (anew/aold)
        cnew = anew * coanew
        an = ao
        bn = bo
        gn = go
        x = math.cos
        y = math.sin
        M_new = [[anew*((1+2.*x(an)*x(bn)*x(gn)-(x(an))
            **2.-(x(bn))**2.-(x(gn))**2.)**0.5/(y(an))),
            anew*((x(gn)-x(bn)*x(an))/(y(an))),anew*x(bn)
            ],\
              [0.,bnew*y(an),bnew*x(an)],\
```

```
            [0.,0.,cnew]]
        M_new = numpy.array(M_new)
    else:
        aold = (M_old[0][0]**2.+M_old[0][1]**2.+M_old
            [0][2]**2.)**(1./2)
        bold = (M_old[1][0]**2.+M_old[1][1]**2.+M_old
            [1][2]**2.)**(1./2)
        cold = (M_old[2][0]**2.+M_old[2][1]**2.+M_old
            [2][2]**2.)**(1./2)
        ao = math.acos((M_old[2][0]*M_old[1][0]+M_old
            [2][1]*M_old[1][1]+M_old[2][2]*M_old[1][2])/(
            cold*bold))
        bo = math.acos((M_old[0][0]*M_old[2][0]+M_old
            [0][1]*M_old[2][1]+M_old[0][2]*M_old[2][2])/(
            aold*cold))
        go = math.acos((M_old[0][0]*M_old[1][0]+M_old
            [0][1]*M_old[1][1]+M_old[0][2]*M_old[1][2])/(
            aold*bold))
        if(dirn == 'ALPHA'): an = ao+coeffs[3]*(-1)
        else: an = ao
        if(dirn == 'BETA'): bn = bo+coeffs[3]*(-1)
        else: bn = bo
        if(dirn == 'GAMMA'): gn = go+coeffs[3]*(-1)
        else: gn = go
        x = math.cos
        y = math.sin
        anew = (aold**6*((1+2*x(ao)*x(bo)*x(go)-(x(ao))
            **2-(x(bo))**2-(x(go))**2)/\
                (1+2*x(an)*x(bn)*x(gn)-(x(an))**2-(x(bn))
                    **2-(x(gn))**2)))**(1./6)
        bnew = bold * (anew/aold)
        cnew = cold * (anew/aold)
        M_new = [[anew*((1+2.*x(an)*x(bn)*x(gn)-(x(an))
            **2.-(x(bn))**2.-(x(gn))**2.)**0.5/(y(an))),
            anew*((x(gn)-x(bn)*x(an))/(y(an))),anew*x(bn)
            ],\
                [0.,bnew*y(an),bnew*x(an)],\
                [0.,0.,cnew]]
        for k in range(3):
            for l in range(3):
                if (abs(M_new[k-1][l-1]) <= 10**(-10)):
                    M_new[k-1][l-1] = 0.
        M_new = numpy.array(M_new)
```

```
    bsvct=doc.xpath('//crystal/basevect')
    fmt = '%16.10f'
    for j in range(3):
        bdummy = str(fmt%M_new[j,0])+str(fmt%M_new[j,1])+
            str(fmt%M_new[j,2])+' '
        bsvct[j].text=bdummy
#-------------------------------------------------------------------------

#%%%%%%%%%%%%%%--- Write new input file ---%%%%%%%%%%%%%%%%#

newinp=open('new_input.xml','w')
newinp.write(ET.tostring(root, method='xml',
                                    pretty_print=True,
                                    xml_declaration=True,
                                    encoding='UTF-8'))
newinp.close()
#-------------------------------------------------------------------------
```