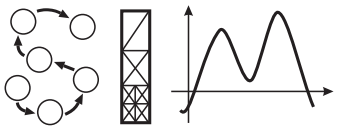
**FRIEDRICH-ALEXANDER-UNIVERSITÄT ERLANGEN-NÜRNBERG**
INSTITUT FÜR INFORMATIK (MATHEMATISCHE MASCHINEN UND DATENVERARBEITUNG)

**Lehrstuhl für Informatik 10 (Systemsimulation)**



# Semiempirische MO-Theorie für Höchstleistungsrechner

Christoph Freundl

Diplomarbeit

# Semiempirische MO-Theorie für Höchstleistungsrechner

Christoph Freundl

Diplomarbeit

| | |
|---|---|
| Aufgabensteller: | Prof. Dr. Ulrich Rüde |
| Betreuer: | PD Dr. Tim Clark |
| Bearbeitungszeitraum: | 15. Januar 2002 – 15. Juli 2002 |

**Erklärung:**

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, den 11. Juli 2002 .....

# Inhaltsverzeichnis

# Vorwort

In der semiempirischen Molekülorbital-Theorie ist der zentrale Bestandteil für Berechnungen an Molekülen die Self-Consistent-Field-Methode, die nicht nur bei semiempirischen Verfahren angewendet wird. Was semiempirische Berechnungen von sog. *ab initio*-Rechnungen unterscheidet, sind im Wesentlichen Vereinfachungen bei der Auswertung von den im SCF-Verfahren zu berechnenden Integralen, z. B. durch Zuhilfenahme empirischer Daten (s. [Cla85] und [Lea96] für eine Übersicht gebräuchlicher semiempirischer Methoden).

Der Löwenanteil an Rechenzeit bei den SCF-Berechnungen wird durch die Lösung von Eigenwertproblemen verbraucht, so dass ein Ziel dieser Arbeit war, die Eigenwert- und Eigenvektorberechnungen zu beschleunigen, und zwar konkret für das am Computer-Chemie-Centrum der Universtät Erlangen entwickelte Programmpaket VAMP. Eine erste Verbesserung kann erzielt werden, indem man anstelle des bisher verwendeten QR-Verfahrens zur Berechnung von Eigenwerten und Eigenvektoren einer reellen symmetrischen Matrix das Divide & Conquer-Verfahren nach Cuppen [Cup81] verwendet. Auch bei serieller Ausführung ist es i. A. schneller als das QR-Verfahren. Zudem bietet es sich an, parallelisiert zu werden. Hier soll ein konkreter Ansatz für die parallele Implementation dieses Algorithmus vorgestellt werden.

Worüber man bei der Implementation sowohl des QR-Verfahrens als auch des Divide & Conquer-Verfahrens zwangsläufig stolpert, ist die (beim QR-Verfahren übliche und beim Divide & Conquer-Verfahren notwendige) Tridiagonalisierung der Matrix, deren Eigenwerte berechnet werden sollen. Auch hierzu wird ein paralleles Rechenverfahren sowie Angaben zu dessen Umsetzung präsentiert.

Es ist anzumerken, dass es mir auf Grund einiger Schwierigkeiten bei der Implementation nicht gelungen ist, über einen Prototypen, mit dem gezeigt werden kann, dass die vorgestellten Ansätze funktionieren, hinauszukommen. Deshalb können in dieser Arbeit leider keine Vergleiche der verschiedenen Rechenverfahren in der Praxis gezogen werden.

Danken möchte ich an dieser Stelle meinen Betreuern Prof. Ulrich Rüde und Marcus Mohr für die für mich sehr hilfreichen und inspirierenden Gespräche zu meinen „persönlichen“ Eigenwertproblemen sowie aus dem Bereich der Chemie meinem Betreuer Dr. Tim Clark und allen Mitarbeitern aus seiner Arbeitsgruppe, die mir einen angenehmen Aufenthalt im Computer-Chemie-Centrum während der Erstellung dieser Arbeit bescherten.

# Kapitel 1

# Die Self-Consistent-Field-Methode

In diesem Kapitel sollen die physikalischen Grundlagen erläutert werden, die den Aufbau von Atomen und Molekülen beschreiben und diese somit überhaupt erst „berechenbar“ machen. Das Ziel wird sein, eine Beschreibung der *Self-Consistent-Field-Methode* (kurz SCF-Methode) zu geben, die einen Algorithmus für eben solche Berechnungen darstellt.

## 1.1 Grundlagen der Quantentheorie

In der klassischen Physik kann die Bewegung von Körpern genau beschrieben werden, d. h. deren Ort und Geschwindigkeit sind zu jedem Zeitpunkt genau bestimmbar. Man sagt auch, die Körper haben Bahnen oder Trajektorien. Alle Eigenschaften eines Körpers, die messbar sind wie eben z. B. dessen Ort oder Geschwindigkeit, heißen auch *Observablen*.

Auf Grund der Heisenbergschen Unschärferelation trifft dies nicht für Teilchen mit geringer Masse zu. Diese haben keine Bahnen mehr, man kann zu einem bestimmten Zeitpunkt nicht mehr genau Ort und Geschwindigkeit angeben. Stattdessen sind nur noch Wahrscheinlichkeitsaussagen dazu möglich.

Man beschreibt daher die Bewegung eines Teilchens durch dessen *Wellenfunktion*. In dieser Wellenfunktion sind alle Informationen über dieses Teilchen enthalten. Jeder Observablen dieses Teilchens entspricht in der Quantenmechanik ein Operator, der, angewandt auf die Wellenfunktion, die entsprechende Information aus dieser „extrahiert“. Die Wellenfunktion ist i. A. komplexwertig.

### 1.1.1 Axiome der Quantentheorie

Bei Schwabl [Sch02] werden die Axiome der Quantentheorie wie folgt angegeben:

1. Der Zustand wird durch die Wellenfunktion $\Psi$ beschrieben
2. Den Observablen entsprechen hermitesche Operatoren, wobei Funktionen von Observablen Funktionen von Operatoren entsprechen
3. Der Mittelwert der Observablen mit zugehörigem Operator $\mathcal{A}$ ist im Zustand $\Psi$ durch

$$\langle \mathcal{A} \rangle = \int \Psi^*(\mathbf{r}) \mathcal{A} \Psi(\mathbf{r}) d\mathbf{r} = \langle \Psi | \mathcal{A} | \Psi \rangle$$

gegeben[1]

4. Die Zeitentwicklung der Zustände wird durch die *Schrödinger-Gleichung* bestimmt
5. Wenn bei Messung von $\mathcal{A}$ der Wert $a_n$ gefunden wurde, geht die Wellenfunktion in die entsprechende Eigenfunktion $\Psi_n$ über

#### 1.1.1.1 Erläuterungen zu den Axiomen

Die Wellenfunktion $\Psi$ an sich ist nicht anschaulich zu interpretieren. Jedoch ist das Quadrat der Wellenfunktion, $|\Psi|^2 = \Psi^*\Psi$, eine Wahrscheinlichkeitsdichte für die Position des betrachteten Systems (s. dazu auch den nächsten Abschnitt).

Die Forderung, dass alle Operatoren hermitesch sind, ist auch einleuchtend, wenn man bedenkt, dass alle Eigenwerte eines Operators genau diejenigen Größen sind, die in dem System gemessen werden können. Messbare Größen müssen aber reell sein, und dies ist nur dann sichergestellt, wenn der Operator hermitesch ist. Eine weitere Folgerung aus der Hermitizität der Operatoren ist, dass die Menge der Eigenfunktionen eines Operators eine Basis bilden, d. h. jede Wellenfunktion ist als Linearkombination dieser Eigenfunktionen darstellbar.

#### 1.1.1.2 Folgerungen aus den Axiomen

Aus den Axiomen ergeben sich u. a. diese Konsequenzen (s. [Sch02]):

- Die möglichen Messwerte einer Observablen sind die Eigenwerte des zugehörigen Operators $\mathcal{A}$
- Seien $a_i$ und $\psi_i$ Eigenwerte und Eigenfunktionen des Operators $\mathcal{A}$ und $\Psi = \sum_i c_i\psi_i$. Dann ist die Wahrscheinlichkeit, dass für die zugehörige Observable der Wert $a_i$ gemessen wird, genau $|c_i|^2$
- $|\Psi|^2$ ist die Wahrscheinlichkeitsdichte für die Position. Da erwünscht ist, dass die Wahrscheinlichkeit, dass das Teilchen irgendwo im Raum zu finden ist, gleich 1 sein soll, muss $\Psi$ normiert werden, so dass $\int |\Psi(\mathbf{r})|d\mathbf{r} = 1$ ist.

### 1.1.2 Die Schrödinger-Gleichung

Die Schrödinger-Gleichung lautet zusammengefasst:

$$\mathcal{H}\Psi = E\Psi \tag{1.1}$$

Sie ist die zentrale Gleichung, die es zu lösen gilt. Der wichtigste Teil der Schrödinger-Gleichung ist der Hamilton-Operator $\mathcal{H}$, der der Operator für die Gesamtenergie ist. Die Gesamtenergie eines Teilchens[2] setzt sich zusammen aus dessen kinetischer und potentieller Energie. Somit lässt sich der Hamilton-Operator schreiben als die Summe des Operators $\mathcal{T}$ für die kinetische Energie und des Operators $\mathcal{V}$ für die potentielle Energie. Die kinetische Energie ist durch die Geschwindigkeit des

[1] Für eine Erläuterung dieser Notation siehe Anhang A
[2] Das kann z. B. ein Elektron, ein Atom oder auch ein großes Molekül sein

Teilchens gegeben, und die potentielle Energie beinhaltet Anziehungs- und Abstoßungskräfte, die auf das Teilchen wirken.

Die Lösung der Schrödinger-Gleichung für ein bestimmtes System erfolgt, indem man den Hamilton-Operator für dieses System aufstellt und anschließend dessen Eigenwerte und Eigenfunktionen berechnet.

Für ein System bestehend aus $n$ Elektronen und $m$ Atomkernen lautet der Hamilton-Operator

$$\mathcal{H} = \mathcal{T} + \mathcal{V} \text{ mit} \tag{1.2}$$

$$\mathcal{T} = -\sum_{i=1}^{n} \frac{1}{2}\nabla_i^2 - \sum_{a=1}^{m} \frac{1}{2m_a}\nabla_a^2 \tag{1.3}$$

$$\mathcal{V} = -\sum_{i=1}^{n}\sum_{a=1}^{m} \frac{Z_a}{r_{ia}} + \sum_{i=1}^{n}\sum_{j>i}^{n} \frac{1}{r_{ij}} + \sum_{a=1}^{m}\sum_{b>a}^{m} \frac{Z_a Z_b}{r_{ab}} \tag{1.4}$$

Der Operator $\mathcal{T}$ beinhaltet den Operator für die kinetische Energie aller Elektronen und den für die kinetische Energie aller Kerne. Die beiden Laplace-Operatoren $\nabla_i^2$ und $\nabla_a^2$ differenzieren nach den Koordinaten des $i$-ten Elektrons bzw. des $a$-ten Kerns.
Im Operator $\mathcal{V}$ für die potentielle Energie sind enthalten: die Anziehung zwischen Elektronen und Kernen im ersten Summanden, die Abstoßung der Elektronen untereinander im zweiten Summanden und schließlich die Abstoßung der Kerne untereinander im letzten Summanden.
$m_a$ ist die Masse des $a$-ten Atomkerns und $Z_a$ ist dessen Kernladungszahl. Für alle Abstände zwischen zwei Teilchen gelte: wenn $\mathbf{r}_i$ und $\mathbf{r}_j$ die Koordinaten des $i$-ten und $j$-ten Elektrons und $\mathbf{s}_a$ und $\mathbf{s}_b$ die Koordinaten des $a$-ten und $b$-ten Kerns sind, dann ist $r_{ia} = |\mathbf{r}_i - \mathbf{s}_a|$ der Abstand zwischen Elektron $i$ und Atomkern $a$, $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$ und $r_{ab} = |\mathbf{s}_a - \mathbf{s}_b|$ entsprechend der Abstand zwischen jeweils zwei Elektronen bzw. Kernen. $r_{12}$ bezeichne grundsätzlich den Abstand zwischen Elektron 1 und 2. Alle Größen sind in atomaren Einheiten anzugeben.

Der Hamilton-Operator und somit die Schrödinger-Gleichung sind kompliziert aufgebaut, so dass, um damit vernünftig umgehen zu können, einige Vereinfachungen unumgänglich sind. Eine erste Vereinfachung betrifft die grundsätzliche Zeitabhängigkeit der Schrödinger-Gleichung. Wir werden im Folgenden nur stationäre Systeme betrachten, so dass sich die Schrödinger-Gleichung auf ihre zeitunabhängige Form reduzieren lässt.
Die weiteren Vereinfachungen und Näherungen werden in den folgenden Abschnitten beschrieben.

### 1.1.3 Die Born-Oppenheimer-Näherung

Die Wellenfunktion ist selbst für ein einfaches Molekül, welches sich aus Atomkernen und Elektronen zusammensetzt, schon relativ komplex, da sie sowohl von den Koordinaten der Elektronen als auch von denen der Kerne abhängt. Ein weiterer Schritt, die Wellenfunktion zu vereinfachen, ist die *Born-Oppenheimer-Näherung*, die besagt, dass man die Bewegung der Elektronen getrennt von der Bewegung der Kerne betrachten kann, da letztere im Vergleich zu ersteren viel größere Massen besitzen und sich somit die Elektronen auf eine Bewegung der Kerne unmittelbar einstellen können. Der Hamilton-Operator (und somit auch die Schrödinger-Gleichung) zerfällt dadurch in einen elektronischen Teil $\mathcal{H}_{\text{elec}}$ sowie in einen nuklearen Teil $\mathcal{H}_{\text{nucl}}$, wobei uns nur der elektronische Teil interessiert, d. h. wir lösen die *elektronische Schrödinger-Gleichung*

$$\mathcal{H}_{\text{elec}}\Psi_{\text{elec}} = E_{\text{elec}}\Psi_{\text{elec}} \text{ mit} \tag{1.5}$$

$$\mathcal{H}_{\text{elec}} = -\sum_{i=1}^{n} \frac{1}{2}\nabla_i^2 - \sum_{i=1}^{n}\sum_{a=1}^{m} \frac{Z_a}{r_{ia}} + \sum_{i=1}^{n}\sum_{j=i+1}^{n} \frac{1}{r_{ij}} \tag{1.6}$$

Die Eigenfunktionen der elektronischen Schrödinger-Gleichung sind die elektronischen Wellenfunktionen mit den zugehörigen elektronischen Energien als Eigenwerten.

Wir betrachten im Folgenden nur noch die elektronische Schrödinger-Gleichung, den elektronischen Hamilton-Operator und die elektronische Energie, so dass wir den Index „elec“ weglassen können.

### 1.1.4 Einelektronensysteme

Die Wellenfunktion eines einzelnen Elektrons bezeichnet man auch als *Orbital*. Ein *Raumorbital* $\psi(\mathbf{r})$ ist eine Funktion der Koordinaten $\mathbf{r}$ eines Elektrons. Es beschreibt die räumliche Verteilung eines Elektrons, so dass $|\psi(\mathbf{r})|^2 d\mathbf{r}$ die Wahrscheinlichkeit ist, das Elektron in einem Volumen $d\mathbf{r}$ um $\mathbf{r}$ zu finden.
Für die Raumorbitale $\{\psi_i\}$ eines Systems nimmt man üblicherweise an, dass sie orthonormal sind.

$$\int \psi_i^*(\mathbf{r})\psi_j(\mathbf{r})d\mathbf{r} = \delta_{ij} \tag{1.7}$$

Die Menge der Raumorbitale ist unendlich, und jede Funktion $f(\mathbf{r})$ kann man als Linearkombination der Basisfunktionen $\{\psi_i\}$ darstellen.

$$f(\mathbf{r}) = \sum_{i=1}^{\infty} a_i \psi_i(\mathbf{r}) \tag{1.8}$$

In der Praxis muss man sich aber auf eine endliche Menge $\{\psi_i, i = 1, 2, \ldots, k\}$ beschränken.

Zur vollständigen Beschreibung eines Elektrons ist zusätzlich noch die Angabe seines Spins notwendig. In die Wellenfunktion muss daher noch die Spinkoordinate $\omega$ des Elektrons eingehen. Das so beschriebene Orbital $\chi(\mathbf{x})$ (mit $\mathbf{x} = (\mathbf{r}, \omega)$), das sowohl von den Raumkoordinaten als auch von der Spinkoordinate des Elektrons abhängt, heißt *Spinorbital*. Elektronen können nur zwei verschiedene Spins annehmen, man bezeichnet sie als Spin “up” bzw. Spin “down” und formuliert zwei entsprechende Spinfunktionen $\alpha(\omega)$ und $\beta(\omega)$, die zueinander orthonormal sind. Aus einem Raumorbital $\psi(\mathbf{r})$ lassen sich somit genau zwei Spinorbitale formulieren, nämlich

$$\chi(\mathbf{x}) = \begin{cases} \psi(\mathbf{r})\alpha(\omega) \\ \psi(\mathbf{r})\beta(\omega) \end{cases} \tag{1.9}$$

Eine Menge von $k$ Raumorbitalen ergibt also eine Menge von $2k$ Spinorbitalen $\{\chi_i\}$ mit

$$\begin{aligned} \chi_{2i-1}(\mathbf{x}) &= \psi_i(\mathbf{r})\alpha(\omega) \\ \chi_{2i}(\mathbf{x}) &= \psi_i(\mathbf{r})\beta(\omega) \end{aligned} \tag{1.10}$$

Sind die Raumorbitale orthonormal, dann sind es auch die daraus gebildeten Spinorbitale.

### 1.1.5 Mehrelektronensysteme

Während die Wellenfunktionen für einzelne Elektronen noch relativ leicht zu bestimmen sind, wird es für Systeme mit mehreren Elektronen schwieriger. Hierzu versucht man, durch geeignete Konstruktion aus einzelnen Spinorbitalen die Gesamtwellenfunktion $\Psi(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ zu formulieren.

#### 1.1.5.1 Das Antisymmetrieprinzip

Die Quantentheorie fordert, dass die Wellenfunktionen eines Systems identischer Teilchen entweder alle symmetrisch oder alle antisymmetrisch sind bezüglich der Vertauschung zweier Teilchen (s. [Rei94]). Im Falle von Elektronen ist die Antisymmetrie der Wellenfunktion notwendig, d. h. es muss gelten:

$$\Psi(\mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_j, \ldots, \mathbf{x}_n) = -\Psi(\mathbf{x}_1, \ldots, \mathbf{x}_j, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_n) \tag{1.11}$$

Das Antisymmetrieprinzip ist eine Verallgemeinerung des eher bekannten *Pauli-Prinzips*, das fordert, dass zwei Elektronen nicht in allen Quantenzahlen übereinstimmen, d. h. dasselbe Spinorbital besetzen. Wäre dies der Fall, dann ergibt sich durch die Erfüllung des Antisymmetrieprinzips, dass die Wellenfunktion überall gleich Null sein müsste.

#### 1.1.5.2 Hartree-Produkte

Angenommen, man hat ein System von zwei Elektronen, wobei Elektron 1 das Orbital $\chi_1$ und Elektron 2 das Orbital $\chi_2$ besetzt, d. h. die Wellenfunktionen der beiden Elektronen sind $\chi_1(\mathbf{x}_1)$ bzw. $\chi_2(\mathbf{x}_2)$. Eine einfache Art, die beiden Wellenfunktionen zu einer Wellenfunktion des Gesamtsystems zu kombinieren, ist die Multiplikation der beiden zu $\Psi(\mathbf{x}_1, \mathbf{x}_2) = \chi_1(\mathbf{x}_1)\chi_2(\mathbf{x}_2)$. Diese Konstruktion bezeichnet man als ein *Hartree-Produkt*.

Die entsprechende Verallgemeinerung auf ein System von $n$ Elektronen lautet[3]:

$$\Psi(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) = \chi_1(\mathbf{x}_1)\chi_2(\mathbf{x}_2)\cdots\chi_n(\mathbf{x}_n) \tag{1.12}$$

Diese Wellenfunktion genügt für Systeme, bei denen die Elektronen nicht miteinander wechselwirken. Jedes Elektron $i$ im Orbital $\chi_j$ besitzt einen eigenen Hamilton-Operator $h(i)$ mit

$$h(i)\chi_j(\mathbf{x}_i) = \varepsilon_j\chi_j(\mathbf{x}_i) \tag{1.13}$$

Die Operatoren $h(i)$ der einzelnen Elektronen addieren sich dann zum Gesamt-Hamilton-Operator

$$\mathcal{H} = \sum_{i=1}^{n} h(i) \tag{1.14}$$

auf. Eine von dessen Eigenfunktionen ist genau das nach Gleichung (1.12) gebildete Hartree-Produkt mit dem zugehörigen Eigenwert

$$E = \varepsilon_i + \varepsilon_j + \ldots + \varepsilon_k \tag{1.15}$$

Nun sind zum einen in Systemen mit mehreren Elektronen diese nicht unabhängig voneinander, d. h. sie wechselwirken miteinander, so dass auch der Hamilton-Operator anders zu formulieren ist. Zum anderen genügt ein Hartree-Produkt nicht dem Antisymmetrieprinzip.

#### 1.1.5.3 Slater-Determinanten

Eine bessere Wahl ist das Heranziehen von *Slater-Determinanten* zur Darstellung der Wellenfunktion. Deren Konstruktion lässt sich, wie in [SO82] dargestellt, wieder am Beispiel von zwei Elektronen (mit den Koordinaten $\mathbf{x}_1$ und $\mathbf{x}_2$) und zwei Spinorbitalen $\chi_i$ und $\chi_j$ leicht nachvollziehen.

[3] Die Indizes der Orbitale sind unabhängig von denen der Elektronen zu betrachten. Es bedeutet einfach nur, dass Elektron $i$ im Orbital $\chi_i$ ist

Man „erzwingt“ die Antisymmetrie, indem man die Hartree-Produkte der beiden möglichen Zustände (Elektron 1 in Orbital $\chi_i$, Elektron 2 in Orbital $\chi_j$ und umgekehrt) geeignet linear kombiniert:

$$\Psi(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{\sqrt{2}}(\chi_i(\mathbf{x}_1)\chi_j(\mathbf{x}_2) - \chi_j(\mathbf{x}_1)\chi_i(\mathbf{x}_2)) \tag{1.16}$$

Die Antisymmetrie, nämlich $\Psi(\mathbf{x}_1, \mathbf{x}_2) = -\Psi(\mathbf{x}_2, \mathbf{x}_1)$, lässt sich leicht verifizieren.

Gleichung (1.16) kann man auch schreiben als

$$\Psi(\mathbf{x}_1, \mathbf{x}_2) = 2^{-\frac{1}{2}} \begin{vmatrix} \chi_i(\mathbf{x}_1) & \chi_j(\mathbf{x}_1) \\ \chi_i(\mathbf{x}_2) & \chi_j(\mathbf{x}_2) \end{vmatrix} \tag{1.17}$$

was die Bezeichnung als Determinante verdeutlicht.

Verallgemeinert man diese Darstellung auf ein System mit $n$ Elektronen, dann ergibt sich als Slater-Determinante

$$\Psi(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) = (n!)^{-\frac{1}{2}} \begin{vmatrix} \chi_i(\mathbf{x}_1) & \chi_j(\mathbf{x}_1) & \cdots & \chi_k(\mathbf{x}_1) \\ \chi_i(\mathbf{x}_2) & \chi_j(\mathbf{x}_2) & \cdots & \chi_k(\mathbf{x}_2) \\ \vdots & \vdots & & \vdots \\ \chi_i(\mathbf{x}_n) & \chi_j(\mathbf{x}_n) & \cdots & \chi_k(\mathbf{x}_n) \end{vmatrix} \tag{1.18}$$

Die Slater-Determinante ist normalisiert durch den Faktor $(n!)^{-\frac{1}{2}}$.
Es bietet sich an, eine Abkürzung für eine Slater-Determinante einzuführen. Wir schreiben für (1.18) auch

$$\Psi(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) = |\chi_i(\mathbf{x}_1)\chi_j(\mathbf{x}_2)\cdots\chi_k(\mathbf{x}_n)\rangle \text{ bzw.} \tag{1.19}$$

$$\Psi(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) = |\chi_i\chi_j\cdots\chi_k\rangle \tag{1.20}$$

wenn die Orbitale in der so angegebenen Reihenfolge die Elektronen $1, 2, \ldots$ enthalten.

### 1.1.6 Ein- und Zweielektronenintegrale

Schreibt man den elektronischen Hamilton-Operator aus (1.6) um in

$$\mathcal{H} = \sum_{i=1}^{n}\left(-\frac{1}{2}\nabla_i^2 - \sum_{a=1}^{m}\frac{Z_a}{r_{ia}}\right) + \sum_{i=1}^{n}\sum_{j=i+1}^{n}\frac{1}{r_{ij}} \tag{1.21}$$

$$= \sum_{i=1}^{n} h(i) + \sum_{\substack{i=1 \\ j>i}}^{n}\frac{1}{r_{ij}} \tag{1.22}$$

$$= \mathcal{O}_1 + \mathcal{O}_2 \tag{1.23}$$

dann hat man die Einelektronenbestandteile von $\mathcal{H}$ in $\mathcal{O}_1$ und die Zweielektronenbestandteile in $\mathcal{O}_2$ aufgeteilt.

#### 1.1.6.1 Einelektronenintegrale

Nun kann man zeigen (siehe [SO82]), dass für den Erwartungswert des Einelektronenoperators $\mathcal{O}_1$ gilt:

$$\langle\Psi_0|\mathcal{O}_1|\Psi_0\rangle = \sum_{i=1}^{n}\int \chi_i^*(\mathbf{x}_1)h(1)\chi_i(\mathbf{x}_1)d\mathbf{x}_1 \tag{1.24}$$

Die Summanden werden über die Koordinaten eines einzelnen Elektrons integriert, man nennt diese daher *Einelektronenintegrale* und führt folgende Abkürzung ein:

$$\langle i|h|j\rangle \equiv \langle \chi_i|h|\chi_j\rangle \equiv \int \chi_i^*(\mathbf{x}_1)h(1)\chi_j(\mathbf{x}_1)d\mathbf{x}_1 \tag{1.25}$$

Damit kann man für (1.24) auch schreiben:

$$\langle \Psi_0|\mathcal{O}_1|\Psi_0\rangle = \sum_{i=1}^{n}\langle i|h|i\rangle \tag{1.26}$$

Formuliert man $\Psi_0$ nicht aus Spinorbitalen sondern aus Raumorbitalen, dann ändert sich obige Schreibweise nur dahingehend, dass man statt der spitzen Klammern runde Klammern benutzt:

$$(i|h|j) \equiv (\psi_i|h|\psi_j) \equiv \int \psi_i^*(\mathbf{r}_1)h(1)\psi_j(\mathbf{r}_1)d\mathbf{r}_1 \tag{1.27}$$

#### 1.1.6.2 Zweielektronenintegrale

Ebenso lässt sich für den Operator $\mathcal{O}_2$ zeigen (s. [SO82]), dass gilt:

$$\begin{aligned}\langle \Psi_0|\mathcal{O}_2|\Psi_0\rangle = \sum_{\substack{i=1\\j>i}}^{n}\Bigg(&\iint \chi_i^*(\mathbf{x}_1)\chi_j^*(\mathbf{x}_2)r_{12}^{-1}\chi_i(\mathbf{x}_1)\chi_j(\mathbf{x}_2)d\mathbf{x}_1d\mathbf{x}_2\\ &-\iint \chi_i^*(\mathbf{x}_1)\chi_j^*(\mathbf{x}_2)r_{12}^{-1}\chi_j(\mathbf{x}_1)\chi_i(\mathbf{x}_2)d\mathbf{x}_1d\mathbf{x}_2\Bigg)\end{aligned} \tag{1.28}$$

Hier sind in den Summanden lauter *Zweielektronenintegrale* enthalten (Integration geht über die Koordinaten zweier Elektronen). Auch hier führt man der Übersichtlichkeit halber eine Abkürzung ein:

$$\langle ij|kl\rangle \equiv \langle \chi_i\chi_j|\chi_k\chi_l\rangle \equiv \iint \chi_i^*(\mathbf{x}_1)\chi_j^*(\mathbf{x}_2)r_{12}^{-1}\chi_k(\mathbf{x}_1)\chi_l(\mathbf{x}_2)d\mathbf{x}_1d\mathbf{x}_2 \tag{1.29}$$

Eine weitere abgekürzte Schreibweise für ein „antisymmetrisiertes“ Zweielektronenintegral lautet schließlich:

$$\langle ij||kl\rangle \equiv \langle ij|kl\rangle - \langle ij|lk\rangle \tag{1.30}$$

(1.28) kürzt sich damit ab zu

$$\langle \Psi_0|\mathcal{O}_2|\Psi_0\rangle = \sum_{\substack{i=1\\j>i}}^{n}\left(\langle ij|ij\rangle - \langle ij|ji\rangle\right) = \sum_{\substack{i=1\\j>i}}^{n}\langle ij||ij\rangle \tag{1.31}$$

Bei der Verwendung von Raumorbitalen an Stelle von Spinorbitalen gibt es folgende Abkürzung für ein Zweielektronenintegral:

$$(ij|kl) \equiv (\psi_i\psi_j|\psi_k\psi_l) \equiv \iint \psi_i^*(\mathbf{r}_1)\psi_j(\mathbf{r}_1)r_{12}^{-1}\psi_k^*(\mathbf{r}_2)\psi_l(\mathbf{r}_2)d\mathbf{r}_1d\mathbf{r}_2 \tag{1.32}$$

## 1.2 Die Hartree-Fock-Näherung

### 1.2.1 Das Variationsprinzip

Die Schrödinger-Gleichung (1.1) kann man nur für die einfachsten Fälle exakt lösen, so dass man für ein gegebenes System i. A. bestenfalls eine gute Näherungslösung berechnen kann.

Nehmen wir an, dass die exakte Lösung für Gleichung (1.1) die Eigenfunktionen $\{\Psi_i\}$ mit den zugehörigen Eigenwerten $\{E_i\}$ sei.

$$\mathcal{H}\,|\Psi_i\rangle = E_i\,|\Psi_i\rangle \qquad i = 0, 1, \ldots \tag{1.33}$$

Die Mengen seien dabei so geordnet, dass $E_0 \leq E_1 \leq \ldots$ gelte.
$\mathcal{H}$ ist ein hermitescher Operator, also sind alle Eigenwerte reell und die Eigenfunktionen $\{\Psi_i\}$ sind orthonormal und bilden eine Basis.

$$\langle\Psi_i|\Psi_j\rangle = \delta_{ij} \tag{1.34}$$

Multipliziert man Gleichung (1.33) von links mit $\langle\Psi_j|$, dann erhält man

$$\langle\Psi_j|\mathcal{H}|\Psi_i\rangle = \langle\Psi_j|E_i|\Psi_i\rangle = E_i\,\langle\Psi_j|\Psi_i\rangle = E_i\delta_{ji} \tag{1.35}$$

Da die Eigenfunktionen $\{\Psi_i\}$ eine Basis bilden, kann man jede Funktion $|\Psi\rangle$, die den gleichen Randbedingungen wie denen der $\{\Psi_i\}$ gehorcht, als Linearkombination derselben schreiben.

$$|\Psi\rangle = \sum_i |\Psi_i\rangle\, a_i = \sum_i |\Psi_i\rangle\,\langle\Psi_i|\Psi\rangle \tag{1.36}$$

Analog gilt

$$\langle\Psi| = \sum_i a_i^*\,\langle\Psi_i| = \sum_i \langle\Psi|\Psi_i\rangle\,\langle\Psi_i| \tag{1.37}$$

Nun gilt

**Satz 1 (Variationsprinzip)** *Ist $\Psi$ eine normalisierte Wellenfunktion mit geeigneten Randbedingungen (üblicherweise, dass sie im Unendlichen verschwindet), dann ist der Erwartungswert des Hamilton-Operators eine obere Schranke für die exakte Energie des Grundzustands. Für*

$$\langle\Psi|\Psi\rangle = 1$$

*gilt*

$$\langle\Psi|\mathcal{H}|\Psi\rangle \geq E_0$$

*Gleichheit gilt nur, wenn $\Psi = \Psi_0$.*

**Beweis:** Betrachtet man zunächst

$$\begin{aligned} 1 = \langle\Psi|\Psi\rangle &= \sum_{i,j} \langle\Psi|\Psi_i\rangle\,\langle\Psi_i|\Psi_j\rangle\,\langle\Psi_j|\Psi\rangle \\ &= \sum_{i,j} \langle\Psi|\Psi_i\rangle\,\delta_{ij}\,\langle\Psi_j|\Psi\rangle \\ &= \sum_i \langle\Psi|\Psi_i\rangle\,\langle\Psi_i|\Psi\rangle \\ &= \sum_i |\langle\Psi_i|\Psi\rangle|^2 \end{aligned} \tag{1.38}$$

und

$$\langle\Psi|\mathcal{H}|\Psi\rangle = \sum_{i,j} \langle\Psi|\Psi_i\rangle\,\langle\Psi_i|\mathcal{H}|\Psi_j\rangle\,\langle\Psi_j|\Psi\rangle = \sum_i E_i\,|\langle\Psi_i|\Psi\rangle|^2 \tag{1.39}$$

dann gilt wegen $E_i \geq E_0$ für alle $i$

$$\langle\Psi|\mathcal{H}|\Psi\rangle \geq \sum_i E_0\,|\langle\Psi_i|\Psi\rangle|^2 = E_0 \sum_i |\langle\Psi_i|\Psi\rangle|^2 = E_0 \tag{1.40}$$

□

Herleitung und Beweis des Variationsprinzips sind [SO82] entnommen.

Das Variationsprinzip besagt also, dass für eine angenäherte Wellenfunktion die daraus berechnete Energie des Grundzustandes immer größer ist als die Grundzustandsenergie der exakten Wellenfunktion. Weiterhin folgt, dass je kleiner die berechnete Grundzustandsenergie einer Nährung ist, diese um so näher am exakten Ergebnis liegt. Man variiert die angenäherte Wellenfunktion also dahingehend, dass die daraus resultierende Energie minimal ist.

### 1.2.2 Übersicht

Der Grundzustand eines Mehrelektronensystems wird durch eine Slater-Determinante angenähert.

$$|\Psi_0\rangle = |\chi_1\chi_2\cdots\chi_n\rangle \tag{1.41}$$

Es gilt nun, die Spinorbitale $\chi_i$ geeignet zu wählen, bis schließlich die aus der Slater-Determinante (1.41) berechnete Energie

$$E_0 = \langle\Psi_0|\mathcal{H}|\Psi_0\rangle \tag{1.42}$$

minimal ist, und nach dem Variationsprinzip $\Psi_0$ somit eine bestmögliche Annäherung an die exakte Wellenfunktion ist.

Dies geschieht, indem man Eigenwertgleichungen der Form

$$f(i)\chi(\mathbf{x}_i) = \varepsilon\chi(\mathbf{x}_i) \tag{1.43}$$

löst. $f(i)$ ist der *Fock-Operator*, er besitzt die Form

$$f(i) = -\frac{1}{2}\nabla_i^2 - \sum_{a=1}^{m}\frac{Z_a}{r_{ia}} + v^{\mathrm{HF}}(i) \tag{1.44}$$

Dieser wirkt effektiv nur auf das Elektron $i$, im Term $v^{\mathrm{HF}}(i)$ sind dabei die Wechselwirkungen aller anderen Elektronen auf dieses spezielle Elektron in Form eines Durchschnittspotentials zusammengefasst. Dies ist auch die wesentliche Eigenschaft der Hartree-Fock-Näherung.

### 1.2.3 Minimierung der Energie

Von der Slater-Determinante $|\Psi_0\rangle$ ausgehend, ist die Energie $E_0 = \langle\Psi_0|\mathcal{H}|\Psi_0\rangle$ ein Funktional der Spinorbitale $\{\chi_i\}$. Wir wollen $E_0[\{\chi_i\}]$ durch geeignete Wahl der Spinorbitale minimieren, unter der Bedingung, dass die so gewählten Spinorbitale orthonormal sind.

$$\langle\chi_i|\chi_j\rangle = \delta_{ij} \tag{1.45}$$

Die elektronische Energie, die minimiert werden soll, lautet

$$\begin{aligned} E_0 = \langle\Psi_0|\mathcal{H}|\Psi_0\rangle = \langle\Psi_0|\mathcal{O}_1 + \mathcal{O}_2|\Psi_0\rangle = \\ \langle\Psi_0|\mathcal{O}_1|\Psi_0\rangle + \langle\Psi_0|\mathcal{O}_2|\Psi_0\rangle = \\ \sum_{i=1}^{n}\langle i|h|i\rangle + \sum_{\substack{i=1\\ j>i}}^{n}\langle ij||ij\rangle \end{aligned} \tag{1.46}$$

Durch die Minimierung dieser Energie unter der Bedingung (1.45) erhält man schließlich eine Gleichung, die diejenigen Spinorbitale definiert, für die $E_0$ minimal wird. Diese Gleichung ist die *Hartree-Fock Integro-Differentialgleichung*:

$$h(1)\chi_i(\mathbf{x}_1) + \sum_{j\neq i}\left(\int |\chi_j(\mathbf{x}_2)|^2 r_{12}^{-1} d\mathbf{x}_2\right)\chi_i(\mathbf{x}_1) - \sum_{j\neq i}\left(\int \chi_j^*(\mathbf{x}_2)\chi_i(\mathbf{x}_2) r_{12}^{-1} d\mathbf{x}_2\right)\chi_j(\mathbf{x}_1) = \varepsilon_i\chi_i(\mathbf{x}_1) \quad (1.47)$$

Für die Herleitung dieser Gleichung sei auf [SO82] verwiesen.

### 1.2.4 Der Fock-Operator

Die beiden Summen in Gleichung (1.47) stehen für alle Wechselwirkungen zwischen den Elektronen. Ohne diese wäre die Gleichung identisch mit (1.13) (bezogen auf ein einzelnes Elektron), wo wir ja Elektron-Elektron-Wechselwirkungen ausgeschlossen haben.

#### 1.2.4.1 Der Coulomb-Operator

Die erste Summe ist der Coulombterm und repräsentiert die durch andere Elektronen verursachten Abstoßungskräfte, die auf das ausgewählte erste Elektron wirken.
Betrachten wir diesen Term genauer: das Gesamtpotential, das auf das erste Elektron wirkt, ist

$$v_i^{\text{coul}}(\mathbf{x}_1) = \sum_{j\neq i}\int |\chi_j(\mathbf{x}_2)|^2 r_{12}^{-1} d\mathbf{x}_2 \quad (1.48)$$

Ein einzelner Summand dieses Terms repräsentiert das Potential, das auf Elektron 1 wirkt, wenn sich ein zweites Elektron im Orbital $\chi_j$ befindet. Die exakte Abstoßung $r_{12}^{-1}$ zwischen den beiden Elektronen wird gemittelt, indem man über alle möglichen Aufenthaltorte $\mathbf{x}_2$ des zweiten Elektrons integriert und diese mit der Wahrscheinlichkeit $|\chi_j(\mathbf{x}_2)|^2$, dass sich Elektron 2 an der Stelle $\mathbf{x}_2$ befindet, gewichtet.
Dies muss natürlich für alle Orbitale $\chi_j$ mit $j \neq i$ geschehen, um die Gesamtwirkung aller $n-1$ Elektronen auf das eine Elektron in $\chi_i$ zu erhalten.
Für diese Wirkung definiert man einen *Coulomb-Operator*

$$\mathcal{J}_j(1) = \int |\chi_j(\mathbf{x}_2)|^2 r_{12}^{-1} d\mathbf{x}_2 \quad (1.49)$$

#### 1.2.4.2 Der Austauschoperator

Man definiert den *Austauschoperator* $\mathcal{K}_j(1)$ so, dass er, auf das Spinorbital $\chi_i(\mathbf{x}_1)$ angewandt, folgendes Ergebnis liefert:

$$\mathcal{K}_j(1)\chi_i(\mathbf{x}_1) = \left(\int \chi_j^*(\mathbf{x}_2) r_{12}^{-1}\chi_i(\mathbf{x}_2) d\mathbf{x}_2\right)\chi_j(\mathbf{x}_1) \quad (1.50)$$

Für diesen Operator gibt es leider keine offensichtliche Erklärung wie für den Coulomb-Operator.

#### 1.2.4.3 Operatorform der Hartree-Fock-Gleichung

Mit den so definierten Operatoren lässt sich die Hartree-Fock-Gleichung (1.47) wie folgt schreiben:

$$\left( h(1) + \sum_j \mathcal{J}_j(1) - \sum_j \mathcal{K}_j(1) \right) \chi_i(\mathbf{x}_1) = \varepsilon_i \chi_i(\mathbf{x}_1) \tag{1.51}$$

Die Beschränkung $j \neq i$ bei den Summen kann man beiseite lassen, da offensichtlich gilt:

$$\left( \mathcal{J}_i(1) - \mathcal{K}_i(1) \right) \chi_i(\mathbf{x}_1) = 0 \tag{1.52}$$

Nun definiert man den *Fock-Operator* durch

$$f(1) = h(1) + \sum_j \left( \mathcal{J}_j(1) - \mathcal{K}_j(1) \right) \tag{1.53}$$

und erhält damit für die Hartree-Fock-Gleichung aus (1.51)

$$f(1)\chi_i(\mathbf{x}_1) = \varepsilon_i \chi_i(\mathbf{x}_i) \tag{1.54}$$

### 1.2.5 Beschränktes Hartree-Fock-Verfahren für Systeme mit abgeschlossenen Schalen

In diesem Abschnitt wollen wir für die Hartree-Fock-Gleichung (1.54) ausführen, wie man diese konkret löst. Wir „beschränken“ uns hier auf Spinorbitale, die die gleiche räumliche Funktion für sowohl $\alpha$- also auch $\beta$-Spin haben. Man bezeichnet dies daher auch als *beschränktes Hartree-Fock-Verfahren* (RHF, von engl. “Restricted Hartree-Fock”). Lässt man diese Einschränkung beiseite, dann läge ein *unbeschränktes Hartree-Fock-Verfahren* (UHF, “Unrestricted Hartree-Fock”) vor.
Desweiteren betrachten wir nur Systeme mit abgeschlossenen Schalen, d. h. in diesen sind alle Raumorbitale doppelt besetzt.

#### 1.2.5.1 Beschreibung des Systems

Unser System habe $n$ Elektronen, die paarweise die Raumorbitale $\psi_i$ mit $i = 1, \ldots, \frac{n}{2}$ besetzen. Die Spinorbitale $\chi_j$ mit $j = 1, \ldots, n$ können aus den Raumorbitalen nach der Vorschrift (1.10) konstruiert werden.
Der Grundzustand unseres Systems lautet dann:

$$|\Psi_0\rangle = |\chi_1 \chi_2 \cdots \chi_n\rangle \tag{1.55}$$

Durch die Beschränkung der Spinorbitale und die Tatsache, dass alle Raumorbitale doppelt besetzt sind, kann man Gleichung (1.54) vereinfachen, indem man alle Spinanteile „wegintegriert“ (für die Vorgehensweise siehe [SO82]). Übrig bleibt die räumliche Hartree-Fock-Gleichung

$$f(1)\psi_i(\mathbf{r}_1) = \varepsilon_i \psi_i(\mathbf{r}_1) \tag{1.56}$$

mit dem Fock-Operator für abgeschlossene Schalen

$$f(1) = h(1) + \sum_{i=1}^{\frac{n}{2}} \left( 2\mathcal{J}_i(1) - \mathcal{K}_i(1) \right) \tag{1.57}$$

und dem entsprechenden Coulomb- und Austauschoperator

$$\mathcal{J}_i(1) = \int \psi_i^*(\mathbf{r}_2) r_{12}^{-1} \psi_i(\mathbf{r}_2) d\mathbf{r}_2 \tag{1.58}$$

$$\mathcal{K}_i(1)\psi_j(\mathbf{r}_1) = \left( \int \psi_i^*(\mathbf{r}_2) r_{12}^{-1} \psi_j(\mathbf{r}_2) d\mathbf{r}_2 \right) \psi_i(\mathbf{r}_1) \tag{1.59}$$

#### 1.2.5.2 Basisfunktionen

Wir führen nun eine Menge $\{\phi_i, i = 1, \ldots, k\}$ von bekannten Basisfunktionen ein und schreiben die Raumorbitale $\psi_i$ als eine Linearkombination derselben:

$$\psi_i = \sum_{b=1}^{k} C_{bi} \phi_b \tag{1.60}$$

Man berechnet die Orbitale $\psi_i$ dadurch, indem man die Koeffizienten $C_{bi}$ berechnet. Setzt man (1.60) in die Hartree-Fock-Gleichung (1.56) ein, so ergibt sich

$$f(1) \sum_{b=1}^{k} C_{bi} \phi_b(\mathbf{r}_1) = \varepsilon_i \sum_b C_{bi} \phi_b(\mathbf{r}_1) \tag{1.61}$$

Durch Multiplikation mit $\phi_a^*(\mathbf{r}_1)$ von links und Integration über $\mathbf{r}_1$ erhält man

$$\sum_b C_{bi} \int \phi_a^*(\mathbf{r}_1) f(1) \phi_b(\mathbf{r}_1) d\mathbf{r}_1 = \varepsilon_i \sum_b C_{bi} \int \phi_a^*(\mathbf{r}_1) \phi_b(\mathbf{r}_1) d\mathbf{r}_1 \tag{1.62}$$

#### 1.2.5.3 Die Roothaan-Gleichungen

Man definiert nun eine *Überlappungsmatrix* $\mathbf{S}$ durch

$$S_{ab} = \int \phi_a^*(\mathbf{r}_1) \phi_b(\mathbf{r}_1) d\mathbf{r}_1 \tag{1.63}$$

und die *Fock-Matrix* $\mathbf{F}$ durch

$$F_{ab} = \int \phi_a^*(\mathbf{r}_1) f(1) \phi_b(\mathbf{r}_1) d\mathbf{r}_1 \tag{1.64}$$

Mit diesen Matrizen kann man Gleichung (1.62) auch als

$$\sum_b F_{ab} C_{bi} = \varepsilon_i \sum_b S_{ab} C_{bi} \qquad i = 1, \ldots, k \tag{1.65}$$

schreiben. Diese sind die *Roothaan-Gleichungen*, die sich auch in eine Matrixgleichung zusammenfassen lassen:

$$\mathbf{FC} = \mathbf{SC}\boldsymbol{\varepsilon} \tag{1.66}$$

Alle Matrizen in dieser Gleichung sind $k \times k$-Matrizen, wobei $\boldsymbol{\varepsilon}$ die Diagonalmatrix mit den Orbitalenergien ist: $\boldsymbol{\varepsilon} = \mathrm{diag}(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_k)$

## 1.3 Formulierung des SCF-Verfahrens

Nun geht es darum, die Gleichung (1.66) zu lösen. Dazu ist es notwendig, die Fock-Matrix $\mathbf{F}$ explizit berechnen zu können.

### 1.3.1 Die Dichtematrix

Da $|\psi_i(\mathbf{r})|^2 d\mathbf{r}$ die Wahrscheinlichkeit angibt, mit der ein Elektron im Orbital $\psi_i$ in einer Umgebung um $\mathbf{r}$ anzutreffen ist, ist $|\psi_i(\mathbf{r})|^2$ gerade die Dichtefunktion dieser Wahrscheinlichkeit. Man bezeichnet diese auch als *Ladungsdichte*.
In unserem System mit abgeschlossenen Schalen (2 Elektronen pro $\psi_i$) ergibt sich somit eine gesamte Ladungsdichte von

$$\rho(\mathbf{r}) = 2\sum_{i=1}^{\frac{n}{2}} |\psi_i(\mathbf{r})|^2 \tag{1.67}$$

Setzt man die Expansion nach den Basisfunktionen in obige Gleichung ein, erhält man

$$\begin{aligned}
\rho(\mathbf{r}) &= 2\sum_{i=1}^{\frac{n}{2}} \psi_i^*(\mathbf{r})\psi_i(\mathbf{r}) \\
&= 2\sum_{i=1}^{\frac{n}{2}} \left( \sum_b C_{bi}^* \phi_b^*(\mathbf{r}) \sum_a C_{ai}\phi_a(\mathbf{r}) \right) \\
&= \sum_{a,b} \left( 2\sum_{i=1}^{\frac{n}{2}} C_{ai} C_{bi}^* \right) \phi_a(\mathbf{r})\phi_b^*(\mathbf{r}) \\
&= \sum_{a,b} P_{ab}\phi_a(\mathbf{r})\phi_b^*(\mathbf{r})
\end{aligned} \tag{1.68}$$

Damit wird also eine *Dichtematrix* $\mathbf{P}$ durch

$$P_{ab} = 2\sum_{i=1}^{\frac{n}{2}} C_{ai} C_{bi}^* \tag{1.69}$$

definiert.

### 1.3.2 Berechnung der Fock-Matrix

Setzt man in die Gleichung (1.64) zur Berechnung der Elemente der Fock-Matrix die Darstellung des Fock-Operators (1.57) ein, so erhält man

$$\begin{aligned}
F_{ab} &= \int \phi_a^*(\mathbf{r}_1) h(1) \phi_b(\mathbf{r}_1) d\mathbf{r}_1 + \sum_{i=1}^{\frac{n}{2}} \int \phi_a^*(\mathbf{r}_1)(2\mathcal{J}_i(1) - \mathcal{K}_i(1))\phi_b(\mathbf{r}_1) d\mathbf{r}_1 \\
&= H_{ab}^{\text{core}} + \sum_{i=1}^{\frac{n}{2}} \iint \big[ 2(\phi_a^*(\mathbf{r}_1)\psi_i^*(\mathbf{r}_2) r_{12}^{-1} \psi_i(\mathbf{r}_2)\phi_b(\mathbf{r}_1)) \\
&\qquad\qquad - \phi_a^*(\mathbf{r}_1)\psi_i^*(\mathbf{r}_2) r_{12}^{-1} \phi_b(\mathbf{r}_2)\psi_i(\mathbf{r}_1) \big] \, d\mathbf{r}_1 d\mathbf{r}_2 \\
&= H_{ab}^{\text{core}} + G_{ab}
\end{aligned} \tag{1.70}$$

Die Elemente $H_{ab}^{\text{core}}$ der *Kern-Hamiltonmatrix* enthalten nur Einelektronenanteile und hängen nur von den verwendeten Basisfunktionen $\phi_a$ ab.

Für die Berechnung der Zweielektronenanteile in $G_{ab}$ in (1.70) setzen wir die Linearexpansion der

noch enthaltenen Raumorbitale $\psi_i$ in den Basisfunktionen ein:

$$\begin{aligned} G_{ab} &= \sum_{i=1}^{\frac{n}{2}} \iint \left[ 2 \left( \phi_a^*(\mathbf{r}_1) \sum_r C_{ri}^* \phi_r^*(\mathbf{r}_2) r_{12}^{-1} \sum_s C_{si} \phi_s(\mathbf{r}_2) \phi_b(\mathbf{r}_1)) \right) \right. \\ &\qquad \left. - \phi_a^*(\mathbf{r}_1) \sum_r C_{ri}^* \phi_r^*(\mathbf{r}_2) r_{12}^{-1} \phi_b(\mathbf{r}_2) \sum_s C_{si} \phi_s(\mathbf{r}_1) \right] d\mathbf{r}_1 d\mathbf{r}_2 \\ &= \sum_{r,s} \left[ 2 \sum_{i=1}^{\frac{n}{2}} C_{ri}^* C_{si} \left( \iint \phi_a^*(\mathbf{r}_1) \phi_r^*(\mathbf{r}_2) \phi_s(\mathbf{r}_2) \phi_b(\mathbf{r}_1) d\mathbf{r}_1 d\mathbf{r}_2 \right. \right. \\ &\qquad \left. \left. - \frac{1}{2} \iint \phi_a^*(\mathbf{r}_1) \phi_r^*(\mathbf{r}_2) \phi_b(\mathbf{r}_2) \phi_s(\mathbf{r}_1) d\mathbf{r}_1 d\mathbf{r}_2 \right) \right] \\ &= \sum_{r,s} P_{rs}((ab|rs) - \frac{1}{2}(as|rb)) \end{aligned} \tag{1.71}$$

Die Abkürzung für das Zweielektronenintegral $(ab|rs)$ ist dabei an Gleichung (1.32) angelehnt, hier ist

$$(ab|rs) \equiv \iint \phi_a^*(\mathbf{r}_1) \phi_b(\mathbf{r}_1) r_{12}^{-1} \phi_r^*(\mathbf{r}_2) \phi_s(\mathbf{r}_2) d\mathbf{r}_1 d\mathbf{r}_2 \tag{1.72}$$

Der Zweielektronenanteil $\mathbf{G}$ hängt von der Dichtematrix $\mathbf{P}$ ab und damit auch die gesamte Fock-Matrix. Die Dichtematrix wiederum berechnet sich in Abhängigkeit von den Koeffizienten $C_{ij}$, die selber aber die Komponenten der Eigenvektoren der Fock-Matrix sind. Damit ist die Eigenwertgleichung (1.66) nichtlinear und muss iterativ gelöst werden, was genau im SCF-Verfahren passiert.

### 1.3.3 Orthogonalisierung der Basis

Der einzig „störende“ Term in der Roothaangleichung (1.66) ist die Überlappungsmatrix $\mathbf{S}$. Wäre diese gleich der Einheitsmatrix, dann würde die Gleichung

$$\mathbf{FC} = \mathbf{C}\boldsymbol{\varepsilon} \tag{1.73}$$

lauten, welche dann einem gewöhnlichen Eigenwertproblem entsprechen würde.

Benutzt man von Anfang an eine orthonormale Basis, dann gilt automatisch $\mathbf{S} = \mathbf{I}$, und man muss sich keine weiteren Gedanken hierzu machen.

Im anderen Fall berechnet sich $\mathbf{S}$ nach Gleichung (1.63). Es gilt $S_{ab} = S_{ba}^*$, d. h. die Matrix $\mathbf{S}$ ist hermitesch ($\mathbf{S} = \mathbf{S}^*$) und kann daher diagonalisiert werden durch eine unitäre Matrix $\mathbf{U}$:

$$\mathbf{U}^*\mathbf{S}\mathbf{U} = \boldsymbol{\sigma} \tag{1.74}$$

$\boldsymbol{\sigma} = \mathrm{diag}(\sigma_1, \ldots, \sigma_k)$ ist die Diagonalmatrix mit den Eigenwerten von $\mathbf{S}$.

Berechnet man nun eine Matrix $\mathbf{X}$ mit

$$\mathbf{X} = \mathbf{U}\boldsymbol{\sigma}^{-\frac{1}{2}}\mathbf{U}^* \tag{1.75}$$

dann ist (mit $\boldsymbol{\sigma}^{-\frac{1}{2}} = \mathrm{diag}(\sigma_1^{-\frac{1}{2}}, \ldots, \sigma_k^{-\frac{1}{2}})$)

$$\begin{aligned} \mathbf{X}^*\mathbf{S}\mathbf{X} &= \mathbf{U}\boldsymbol{\sigma}^{-\frac{1}{2}}\mathbf{U}^*\mathbf{S}\mathbf{U}\boldsymbol{\sigma}^{-\frac{1}{2}}\mathbf{U}^* \\ &= \mathbf{U}\boldsymbol{\sigma}^{-\frac{1}{2}}\boldsymbol{\sigma}\boldsymbol{\sigma}^{-\frac{1}{2}}\mathbf{U}^* \\ &= \mathbf{U}\boldsymbol{\sigma}^{-\frac{1}{2}}\boldsymbol{\sigma}^{\frac{1}{2}}\mathbf{U}^* \\ &= \mathbf{U}\mathbf{U}^* \\ &= \mathbf{I} \end{aligned} \tag{1.76}$$

Diese Transformationsmatrix $\mathbf{X}$ wirke nun folgendermaßen auf die Koeffizientenmatrix:

$$\mathbf{C}' = \mathbf{X}^{-1}\mathbf{C} \qquad \mathbf{C} = \mathbf{X}\mathbf{C}' \tag{1.77}$$

Setzt man nun für $\mathbf{C}$ in Gleichung (1.66) obigen Ausdruck ein und multipliziert beide Seiten von links mit $\mathbf{X}^*$, so ergibt sich

$$\begin{aligned} \mathbf{X}^*\mathbf{F}\mathbf{X}\mathbf{C}' &= \mathbf{X}^*\mathbf{S}\mathbf{X}\mathbf{C}'\boldsymbol{\varepsilon} \\ \mathbf{F}'\mathbf{C}' &= \mathbf{C}'\boldsymbol{\varepsilon} \end{aligned} \tag{1.78}$$

Dieses und weitere Verfahren zur Orthogonalisierung der Basis sind in [SO82] zu finden.

Diese transformierte Roothaan-Gleichung besitzt nun die gewünschte Form wie in (1.73). Man löst diese durch Berechnung der Eigenwerte $\varepsilon_i$ samt zugehöriger Eigenvektoren (die Spalten der Matrix $\mathbf{C}'$) von der transformierten Fock-Matrix $\mathbf{F}'$. Die Eigenvektoren müssen dann nur noch nach (1.77) zu den Eigenvektoren der ursprünglichen Fock-Matrix transformiert werden.

### 1.3.4 Der Algorithmus

Jetzt sind wir endlich so weit, um die einzelnen Arbeitsschritte des SCF-Verfahrens angeben zu können (s. [SO82]):

1. Beschreibe das System (durch die Koordinaten der Kerne $\{\mathbf{s}_a\}$, die Kernladungszahlen $\{Z_a\}$ und die Anzahl der Elektronen $n$) und eine Menge von Basisfunktionen $\{\phi_a\}$
2. Berechne alle benötigten Integrale, nämlich $S_{ab}$, $H_{ab}^{\text{core}}$ und $(ab|rs)$
3. Diagonalisiere die Überlappungsmatrix $\mathbf{S}$ und berechne daraus die Transformationsmatrix $\mathbf{X}$
4. Formuliere eine Schätzung für die Dichtematrix $\mathbf{P}$
5. Berechne den Zweielektronenanteil $\mathbf{G}$ der Fock-Matrix aus der Dichtematrix und den Integralen $(ab|rs)$
6. Berechne daraus zusammen mit dem Kern-Hamiltonanteil die Fock-Matrix $\mathbf{F} = \mathbf{H}^{\text{core}} + \mathbf{G}$
7. Transformiere die Fock-Matrix zu $\mathbf{F}' = \mathbf{X}^T\mathbf{F}\mathbf{X}$
8. Diagonalisiere $\mathbf{F}'$, um $\mathbf{C}'$ und $\boldsymbol{\varepsilon}$ zu erhalten
9. Berechne $\mathbf{C} = \mathbf{X}\mathbf{C}'$
10. Berechne eine neue Dichtematrix $\mathbf{P}$ aus $\mathbf{C}$
11. Prüfe, ob die Abbruchkriterien der Iteration erfüllt sind, also ob sich die Dichtematrix im Vergleich zum vorhergehenden Iterationsschritt wesentlich geändert hat. Ist dies nicht der Fall, beginne die nächste Iteration bei Schritt 5
12. Die Iteration hat konvergiert

### 1.3.5 Das Ergebnis

Hat das SCF-Verfahren konvergiert, dann hat man als Ergebnis mit der Koeffizientenmatrix $\mathbf{C}$ eine Menge von $k$ Raumorbitalen $\{\psi_i\}$ mit den zugehörigen Orbitalenergien $\{\varepsilon_i\}$. Von diesen $k$ Raumorbitalen werden im Grundzustand des Systems diejenigen $\frac{n}{2}$ Orbitale mit den niedrigsten Energien $\varepsilon_i$ mit jeweils zwei Elektronen besetzt. Man nennt diese daher *besetzte* Orbitale. Die restlichen Orbitale, die unbesetzt bleiben, werden auch als *virtuelle* Orbitale bezeichnet. In Abb. 1.1 ist dies veranschaulicht; die Pfeile symbolisieren die Elektronen, die Pfeilrichtung steht dabei für den Spin eines Elektrons ("up" bzw. "down").

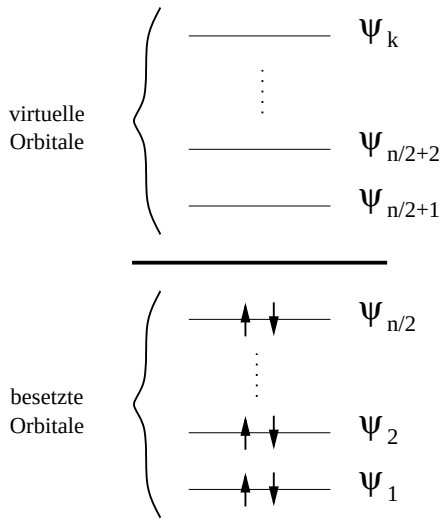



Abbildung 1.1: Besetzte und virtuelle Orbitale

# Kapitel 2

# Eigenwertprobleme

Der entscheidende und rechenintensivste Schritt im SCF-Verfahren ist die Lösung des Eigenwertproblems für die Fock-Matrix. Nach einigen grundlegenden Aussagen zu Eigenwertproblemen im Allgemeinen und einer knappen Darstellung des QR-Verfahrens zur Lösung des Eigenwertproblems wird ein weiteres Rechenverfahren präsentiert, dass einen Divide & Conquer-Ansatz verfolgt.

## 2.1 Klassifikation

### 2.1.1 Definitionen

Für eine Matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$ heißt eine Zahl $\lambda \in \mathbb{C}$ *Eigenwert* von $\mathbf{A}$, wenn es einen Vektor $\mathbf{x} \in \mathbb{C}^n$, $\mathbf{x} \neq \mathbf{0}$ gibt mit

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x} \tag{2.1}$$

Der Vektor $\mathbf{x}$ heißt dann *Eigenvektor* zum Eigenwert $\lambda$; $\lambda$ und $\mathbf{x}$ bilden ein *Eigenpaar* von $\mathbf{A}$.

Für ein Eigenpaar gilt

$$\begin{aligned} \mathbf{A}\mathbf{x} - \lambda\mathbf{x} &= \mathbf{0} \\ (\mathbf{A} - \lambda\mathbf{I})\mathbf{x} &= \mathbf{0} \end{aligned} \tag{2.2}$$

Diese Gleichung besitzt nur dann nichttriviale Lösungen, wenn $\det(\mathbf{A} - \lambda\mathbf{I}) = 0$ ist.

Man definiert ein Polynom $p(\lambda)$ durch

$$p(\lambda) := \det(\mathbf{A} - \lambda\mathbf{I}) = (-1)^n\lambda^n + a_{n-1}\lambda^{n-1} + \ldots + a_1\lambda + a_0 \tag{2.3}$$

Das Polynom $p(\lambda)$ heißt *charakteristisches Polynom* von $\mathbf{A}$. Die Nullstellen des charakteristischen Polynoms sind genau die Eigenwerte von $\mathbf{A}$. Das charakteristische Polynom ist von Grad $n$, also kann $\mathbf{A}$ höchstens $n$ voneinander verschiedene Eigenwerte besitzen.

### 2.1.2 Matrixtransformationen

#### 2.1.2.1 Ähnliche Matrizen

Zwei Matrizen $\mathbf{A}$, $\mathbf{B}$ heißen *ähnlich*, wenn es eine invertierbare Matrix $\mathbf{T}$ gibt, so dass

$$\mathbf{T}^{-1}\mathbf{A}\mathbf{T} = \mathbf{B} \tag{2.4}$$

Man kann leicht zeigen, dass zwei ähnliche Matrizen die gleichen Eigenwerte besitzen. Ist für die Matrizen aus (2.4) $(\lambda, \mathbf{x})$ ein Eigenpaar von $\mathbf{A}$ und setzt man $\mathbf{y} = \mathbf{T}^{-1}\mathbf{x}$, dann ist

$$\mathbf{B}\mathbf{y} = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}\mathbf{y} = \mathbf{T}^{-1}\mathbf{A}\mathbf{x} = \mathbf{T}^{-1}\lambda\mathbf{x} = \lambda\mathbf{T}^{-1}\mathbf{x} = \lambda\mathbf{y} \tag{2.5}$$

Also ist $\lambda$ ebenfalls Eigenwert von $\mathbf{B}$, der zugehörige Eigenvektor ist $\mathbf{T}^{-1}\mathbf{x}$.

#### 2.1.2.2 Diagonalisierbare Matrizen

Für eine Diagonalmatrix $\mathbf{D} = \text{diag}(d_1, \ldots, d_n)$ lassen sich die Eigenwerte leicht bestimmen, es sind nämlich genau die Einträge $d_i$ auf der Diagonalen. Die Eigenvektoren sind genau die Einheitsvektoren. Wir zeigen dies für das Eigenpaar $(d_i, \mathbf{e}_i)$:

$$\mathbf{D}\mathbf{e}_i = (0, \ldots, 0, d_i, 0, \ldots, 0)^T = d_i\mathbf{e}_i \tag{2.6}$$

Eine übliche Vorgehensweise zur Bestimmung der Eigenwerte einer Matrix ist daher, die Matrix durch Ähnlichkeitstransformationen auf Diagonalgestalt zu bringen, da dann die Eigenwerte leicht ablesbar sind. Nun ist dies nicht für jede Matrix möglich; wir bezeichnen Matrizen, die in eine Diagonalmatrix transformiert werden können, als *diagonalisierbar*.

Um diagonalisierbar zu sein, muss eine Matrix $\mathbf{A}$ *normal* sein, i. e. es muss $\mathbf{A}\mathbf{A}^* = \mathbf{A}^*\mathbf{A}$ gelten. Beispiele für normale Matrizen sind

- *Unitäre Matrizen*: es gilt $\mathbf{A}^* = \mathbf{A}^{-1}$. Ist die Matrix $\mathbf{A}$ reell, dann bezeichnet man sie auch als *orthogonal*
- *Hermitesche Matrizen*: für diese gilt $\mathbf{A} = \mathbf{A}^*$. Speziell für reelle Matrizen bedeutet dies, dass diese *symmetrisch* sind

Für eine diagonalisierbare Matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$ gilt insbesondere:

- Sie besitzt $n$ verschiedene Eigenwerte
- Die Eigenvektoren bilden eine Basis des $\mathbb{C}^n$

## 2.2 Das QR-Verfahren

Das QR-Verfahren ist eine der häufig verwendeten Methoden zur Berechnung der Eigenwerte und Eigenvektoren einer Matrix $\mathbf{A}$. Der Ablauf ist recht unkompliziert:
Man setzt $\mathbf{A}_0 := \mathbf{A}$ und berechnet eine Folge von Matrizen $\{\mathbf{A}_i\}$ nach folgender Vorschrift:

$$\mathbf{A}_i = \mathbf{Q}_i\mathbf{R}_i \tag{2.7}$$
$$\mathbf{A}_{i+1} = \mathbf{R}_i\mathbf{Q}_i \tag{2.8}$$

In (2.7) wird die *QR-Zerlegung* der Matrix $\mathbf{A}_i$ berechnet. Dabei ist $\mathbf{Q}_i$ eine unitäre Matrix, und $\mathbf{R}_i$ ist eine rechte obere Dreicksmatrix. Man konstruiert die QR-Zerlegung mittels Householder-Matrizen (s. Abschnitt 3.1.1), die nacheinander spaltenweise alle Einträge von $\mathbf{A}_i$ unterhalb der Diagonalen eliminieren. Diese Konstruktion ist unabhängig von der Matrix $\mathbf{A}_i$ und ist daher für alle Matrizen möglich.

Man kann nun zeigen, dass die Matrizen $\{\mathbf{A}_i\}$ alle zueinander ähnlich sind, und dass die Folge $\{\mathbf{A}_i\}$ gegen eine rechte obere Dreiecksmatrix konvergiert, deren Diagonaleinträge genau die Eigenwerte von $\mathbf{A}$ sind.

Dies ist eine sehr kurze Darstellung des QR-Verfahrens, die aber reichen sollte, um dessen prinzipielle Vorgehensweise zu verstehen. Für eine genauere Beschreibung sei an dieser Stelle auf [SB90] verwiesen.

## 2.3 Ein Divide & Conquer-Verfahren für das Eigenproblem symmetrisch tridiagonaler Matrizen

Dieses Verfahren beruht darauf, dass man eine symmetrische tridiagonale Matrix leicht in kleinere ebenfalls symmetrisch tridiagonale Matrizen zerlegen kann. Aus den berechneten Eigenwerten und Eigenvektoren dieser kleineren Matrizen kann man eine Lösung für die unzerlegte Matrix rekonstruieren.
Die Zerlegung der Matrix geschieht über eine sog. Rang-Eins-Änderung. Bei der Rekonstruktion der Lösung muss man wissen, wie sich die Eigenwerte und Eigenvektoren auf Grund dieser Änderung verhalten. Daher betrachten wir zunächst genau diese Aufgabenstellung, die sich auch auf nicht tridiagonale — aber doch symmetrische — Matrizen anwenden lässt. Anschließend folgt die Vorgehensweise speziell für tridiagonale Matrizen.

### 2.3.1 Rang-Eins-Änderung des symmetrischen Eigenproblems

#### 2.3.1.1 Formulierung

Wie ändern sich die Eigenwerte und Eigenvektoren einer reellen symmetrischen Matrix $\mathbf{A}$, wenn man diese abwandelt in die Form

$$\tilde{\mathbf{A}} = \mathbf{A} + \sigma \mathbf{u}\mathbf{u}^T \tag{2.9}$$

mit $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{u} \in \mathbb{R}^n$ und $\sigma \in \mathbb{R}$? Man nennt dies eine *Rang-Eins-Änderung* der Matrix $\mathbf{A}$. Bunch, Nielsen und Sorensen [BNS78] haben sich mit diesem Problem befasst und geben an, wie Eigenwerte und Eigenvektoren der modifizierten Matrix zu berechnen sind.

Wir gehen davon aus, dass man die Zerlegung $\mathbf{A} = \mathbf{Q}\mathbf{D}\mathbf{Q}^T$ kennt, wobei $\mathbf{D}$ die Diagonalmatrix ist, die genau die Eigenwerte von $\mathbf{A}$ enthält. Dementsprechend sind die Spalten von $\mathbf{Q}$ die zugehörigen normierten Eigenvektoren.

Daraus ergibt sich nun

$$\begin{aligned} \tilde{\mathbf{A}} = \mathbf{A} + \sigma \mathbf{u}\mathbf{u}^T &= \mathbf{Q}\mathbf{D}\mathbf{Q}^T + \sigma \mathbf{u}\mathbf{u}^T \\ &= \mathbf{Q}(\mathbf{D} + \sigma \mathbf{Q}^T\mathbf{u}\mathbf{u}^T\mathbf{Q})\mathbf{Q}^T \\ &= \mathbf{Q}(\mathbf{D} + \sigma \mathbf{z}\mathbf{z}^T)\mathbf{Q}^T \end{aligned} \tag{2.10}$$

mit $\mathbf{z} = \mathbf{Q}^T\mathbf{u}$.

Die Eigenwerte von $\tilde{\mathbf{A}}$ sind demnach genau die Eigenwerte von $\mathbf{D} + \sigma\mathbf{z}\mathbf{z}^T$. Die zugehörigen Eigenvektoren sind von der Form $\mathbf{Q}\mathbf{X}$, wenn $\mathbf{D} + \sigma\mathbf{z}\mathbf{z}^T = \mathbf{X}\mathbf{\Lambda}\mathbf{X}^T$ mit $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$.

Es gilt nun folgender

**Satz 2** *Es sei* $\mathbf{C} = \mathbf{D} + \sigma\mathbf{z}\mathbf{z}^T$*, wobei* $\mathbf{D} \in \mathbb{R}^{n\times n}$ *eine Diagonalmatrix ist,* $\sigma \in \mathbb{R}$ *und* $\mathbf{z} \in \mathbb{R}^n$ *mit* $\|z\|_2 = 1$*. Weiterhin seien* $d_1 \leq d_2 \leq \ldots \leq d_n$ *die Eigenwerte von* $\mathbf{D}$ *und* $\tilde{d}_1 \leq \tilde{d}_2 \leq \ldots \leq \tilde{d}_n$ *die Eigenwerte von* $\mathbf{C}$*.*
*Dann gilt* $\tilde{d}_i = d_i + \sigma\mu_i$ *für* $1 \leq i \leq n$*, wobei* $\sum_{i=1}^n \mu_i = 1$ *und* $0 \leq \mu_i \leq 1$*. Außerdem gilt* $d_1 \leq \tilde{d}_1 \leq d_2 \leq \tilde{d}_2 \leq \ldots d_n \leq \tilde{d}_n$ *falls* $\sigma > 0$ *bzw.* $\tilde{d}_1 \leq d_1 \leq \tilde{d}_2 \leq d_2 \leq \ldots \leq \tilde{d}_n \leq d_n$ *falls* $\sigma < 0$*.*
*Sind zudem alle Eigenwerte* $d_i$ *verschieden und alle Elemente von* $\mathbf{z}$ *ungleich* 0*, dann liegen die Eigenwerte von* $\mathbf{C}$ *echt zwischen denjenigen von* $\mathbf{D}$*, d. h. es gilt* $d_1 < \tilde{d}_1 < d_2 < \tilde{d}_2 < \ldots d_n < \tilde{d}_n$*.*

Für einen Beweis dieses Satzes sei auf [Tho76] bzw. auf [Wil65, S. 95–98] verwiesen.

#### 2.3.1.2 Deflation

Betrachten wir nun wieder $\mathbf{A} + \sigma\mathbf{u}\mathbf{u}^T = \mathbf{Q}(\mathbf{D} + \sigma\mathbf{z}\mathbf{z}^T)\mathbf{Q}^T$. Ohne Beschränkung der Allgemeinheit können wir für $\mathbf{z} = (z_1, z_2, \ldots, z_n)^T$ annehmen, dass $\|z\|_2 = 1$ gilt.

Für einige Fälle vereinfacht sich die Berechnung der Eigenwerte und Eigenvektoren von $\mathbf{D} + \sigma\mathbf{z}\mathbf{z}^T$:

1. Wenn $z_i = 0$ für ein $i$, dann gilt

$$(\mathbf{D} + \sigma\mathbf{z}\mathbf{z}^T)\mathbf{e}_i = \mathbf{D}\mathbf{e}_i + \sigma\mathbf{z}z_i = \mathbf{D}\mathbf{e}_i = d_i\mathbf{e}_i \tag{2.11}$$

   Also ist $\tilde{d}_i = d_i$ Eigenwert von $\mathbf{D}+\sigma\mathbf{z}\mathbf{z}^T$ mit dem Eigenvektor $\mathbf{e}_i$ (i. e. der $i$-te Einheitsvektor). Für die Matrix $\mathbf{A} + \sigma\mathbf{u}\mathbf{u}^T$ ist $d_i$ ebenso Eigenwert zu demselben Eigenvektor $\mathbf{q}_i$, der auch schon Eigenvektor von $\mathbf{D}$ ist.

2. Wenn für ein $i$ eine Komponente $|z_i| = 1$ ist, dann sind alle anderen Komponenten $z_j = 0$ (mit $j \neq i$). Somit ist $\tilde{d}_i = d_i + \sigma$ und für alle $j \neq i$ gilt $\tilde{d}_j = d_j$, für die zugehörigen Eigenvektoren treffen die gleichen Überlegungen wie im vorigen Fall zu. Auch der $i$-te Eigenvektor verändert sich nicht, da $(\mathbf{D} + \sigma\mathbf{z}\mathbf{z}^T)\mathbf{e}_i = (d_i + \sigma)\mathbf{e}_i$ gilt.

3. Wenn ein Eigenwert mehrfach vorkommt, d. h.

$$d_{i+1} = d_{i+2} = \ldots = d_{i+k}$$

   dann gilt für jede beliebige orthonormale Matrix $\mathbf{P}' \in \mathbb{R}^{k\times k}$:

$$\left(\begin{array}{c|c|c} \mathbf{I}_i & & \\ \hline & \mathbf{P}' & \\ \hline & & \mathbf{I}_{n-i-k} \end{array}\right) \mathbf{D} \left(\begin{array}{c|c|c} \mathbf{I}_i & & \\ \hline & \mathbf{P}' & \\ \hline & & \mathbf{I}_{n-i-k} \end{array}\right)^T = \mathbf{D}$$

   Wählt man nun für $\mathbf{P}'$ diejenige Householder-Transformation (siehe dazu auch Abschnitt 3.1.1), die den Vektor $(z_{i+1}, z_{i+2}, \ldots, z_{i+k})$ in einen Vektor $(*, 0, \ldots, 0)$ überführt, dann gilt für

$$\mathbf{P} = \left(\begin{array}{c|c|c} \mathbf{I}_i & & \\ \hline & \mathbf{P}' & \\ \hline & & \mathbf{I}_{n-i-k} \end{array}\right)$$

$$\mathbf{P}(\mathbf{D} + \sigma\mathbf{z}\mathbf{z}^T)\mathbf{P}^T = \mathbf{D} + \sigma\mathbf{P}\mathbf{z}\mathbf{z}^T\mathbf{P}^T = \mathbf{D} + \sigma\tilde{\mathbf{z}}\tilde{\mathbf{z}}^T$$

   Der Vektor $\tilde{\mathbf{z}} = \mathbf{P}\mathbf{z} = (z_1, \ldots, z_i, *, 0, \ldots, 0, z_{i+k+1}, \ldots, z_n)^T$ besitzt $k-1$ Elemente, die gleich Null sind und sich somit auf Fall 1 zurückführen lassen.

Jedes Vorkommen von einem der drei Fälle (Fall 2 ist dabei eigentlich nur ein Spezialfall) verursacht eine *Deflation*. Dadurch lässt sich sich das Problem so weit verkleinern, bis es schließlich von der Form $\mathbf{D} + \sigma\mathbf{z}\mathbf{z}^T$ ist mit $n' \leq n$, $\mathbf{D} = \text{diag}(d_1, \ldots, d_{n'})$, $d_1 < d_2 < \ldots < d_{n'}$ und $\mathbf{z} \in \mathbb{R}^{n'}$, wobei $z_i \neq 0$ für alle $i$. Ohne Beschränkung der Allgemeinheit lässt sich weiterhin annehmen, dass $\sigma > 0$ ist, da der Fall $\sigma = 0$ trivial wäre und man für $\sigma < 0$ nur die Matrix $-\mathbf{D}$ betrachten müsste.

**Satz 3** *Für eine Diagonalmatrix $\mathbf{D} = \text{diag}(d_1, \ldots, d_n)$ mit $d_1 < d_2 < \ldots < d_n$, einen Vektor $\mathbf{z} \in \mathbb{R}^n$ mit $z_i \neq 0$ für alle $i = 1, \ldots, n$ und ein reelles $\sigma > 0$ sind die Eigenwerte der Matrix $\mathbf{D} + \sigma\mathbf{z}\mathbf{z}^T$ genau die $n$ Nullstellen $\lambda_1 < \ldots < \lambda_n$ der rationalen Funktion*

$$\omega(\lambda) = 1 + \sigma\mathbf{z}^T(\mathbf{D} - \lambda\mathbf{I})^{-1}\mathbf{z} = 1 + \sigma\sum_{i=1}^{n}\frac{z_i^2}{d_i - \lambda} \tag{2.12}$$

*Die zugehörigen Eigenvektoren $\mathbf{p}_1, \ldots, \mathbf{p}_n$ erhält man durch*

$$\mathbf{p}_i = \frac{(\mathbf{D} - \lambda_i\mathbf{I})^{-1}\mathbf{z}}{\|(\mathbf{D} - \lambda_i\mathbf{I})^{-1}\mathbf{z}\|_2} \tag{2.13}$$

*Weiterhin gilt*

$$d_1 < \lambda_1 < d_1 < \lambda_2 < \ldots < d_n < \lambda_n < d_n + \sigma\mathbf{z}^T\mathbf{z} \tag{2.14}$$

**Beweis (nach [Cup81]):** Für einen Eigenwert $\lambda$ von $\mathbf{D} + \sigma\mathbf{z}\mathbf{z}^T$ zum Eigenvektor $\mathbf{p}$ gilt

$$(\mathbf{D} + \sigma\mathbf{z}\mathbf{z}^T)\mathbf{p} = \lambda\mathbf{p} \tag{2.15}$$

$$(\mathbf{D} - \lambda\mathbf{I})\mathbf{p} = -\sigma\mathbf{z}^T\mathbf{p}\mathbf{z} \tag{2.16}$$

Als nächstes ist zu zeigen, dass $\mathbf{D} - \lambda\mathbf{I}$ nicht singulär ist. Wenn $\mathbf{D} - \lambda\mathbf{I}$ singulär wäre, dann gilt für ein $i$: $\lambda = d_i$, also ist $0 = ((\mathbf{D} - \lambda\mathbf{I})\mathbf{p})_i = -\sigma\mathbf{z}^T\mathbf{p}z_i$. Das führt wegen $z_i \neq 0$ zu $\mathbf{z}^T\mathbf{p} = 0$ und $(\mathbf{D} - \lambda\mathbf{I})\mathbf{p} = -\sigma\mathbf{z}^T\mathbf{p}\mathbf{z} = \mathbf{0}$, somit ist $(d_j - \lambda)p_j = 0$ für alle $j$ und somit $p_j = 0$ für $j \neq i$, weil ja in diesen Fällen $d_j - \lambda \neq 0$ gilt. Damit ist $0 = \mathbf{z}^T\mathbf{p} = z_ip_i$ und folglich $z_i = 0$, was unserer Annahme widerspricht.
Man kann daher obige Rechnung weiterführen zu

$$\mathbf{p} = -\sigma\mathbf{z}^T\mathbf{p}(\mathbf{D} - \lambda\mathbf{I})^{-1}\mathbf{z} \tag{2.17}$$

$$\mathbf{z}^T\mathbf{p} = -\sigma\mathbf{z}^T\mathbf{p}\mathbf{z}^T(\mathbf{D} - \lambda\mathbf{I})^{-1}\mathbf{z} \tag{2.18}$$

Wegen $\mathbf{z}^T\mathbf{p} \neq 0$ gilt daher

$$1 = -\sigma\mathbf{z}^T(\mathbf{D} - \lambda\mathbf{I})^{-1}\mathbf{z} \tag{2.19}$$

und man erkennt sofort, dass $\lambda$ eine Nullstelle von (2.12) ist. Weiterhin ist auf Grund von (2.17) klar, dass für die normierten Eigenvektoren (2.13) gilt. □

Die Funktion (2.12) wird bei Golub [Gol73] auch als *Säkularfunktion* bezeichnet; dort wird auch darauf hingewiesen, dass die Nullstellen dieser Funktion die Eigenwerte der modifizierten Matrix sind. Die Formel (2.13) für die Eigenwerte geht auf Bunch, Nielsen und Sorensen [BNS78] zurück.

#### 2.3.1.3 Lösung der Säkulargleichung

Der zentrale Punkt bei der Berechnung der Eigenwerte einer rang-eins-modifizierten Matrix ist die Aufgabe, die Nullstellen der Säkularfunktion zu bestimmen. Hierzu bedient man sich üblicherweise einer Vorgehensweise von Bunch, Nielsen und Sorensen [BNS78], die an das Newton-Verfahren angelehnt ist, aber nicht auf lokaler linearer, sondern lokaler rationaler Approximation beruht.

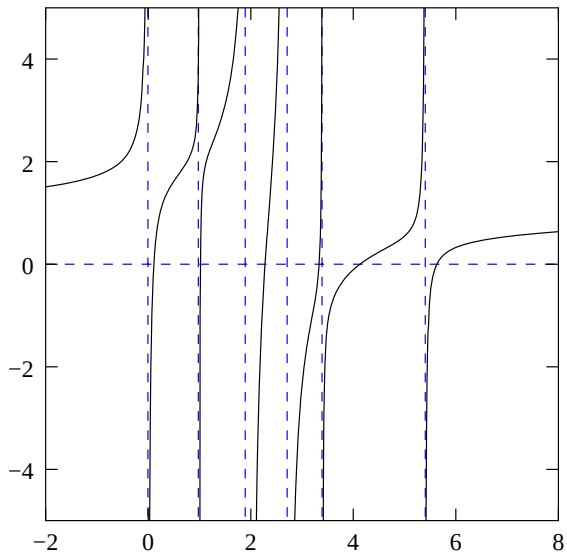



Abbildung 2.1: Beispiel für eine Säkularfunktion

Um die $i$-te Nullstelle $\lambda_i$ zu finden, definiert man zunächst zwei Funktionen

$$\Psi(\lambda) = \sigma \sum_{j=1}^{i} \frac{z_j^2}{d_j - \lambda} \tag{2.20}$$

$$\Phi(\lambda) = \sigma \sum_{j=i+1}^{n} \frac{z_j^2}{d_j - \lambda} \tag{2.21}$$

Damit kann man (2.12) schreiben als

$$\omega(\lambda) = 1 + \Psi(\lambda) + \Phi(\lambda) \tag{2.22}$$

und im Folgenden ist die Gleichung

$$-\Psi(\lambda) = 1 + \Phi(\lambda) \tag{2.23}$$

zu lösen.

Man bedient sich dazu rationaler Interpolanten zur Approximation der Funktionen $\Psi$ und $\Phi$:

$$F(t; p, q) := \frac{p}{q - t} \quad \text{und} \tag{2.24a}$$

$$G(t; d, r, s) := r + \frac{s}{d - t} \tag{2.24b}$$

[BNS78] konstruieren nun ein Folge dieser Interpolanten, beginnend mit:

$$\Psi(t_0) = F(t_0; p, q) = \frac{p}{q - t_0} \qquad \Phi(t_0) = G(t_0; d, r, s) = r + \frac{s}{d - t_0} \tag{2.25a}$$

$$\Psi'(t_0) = F'(t_0; p, q) = \frac{p}{(q - t_0)^2} \qquad \Phi'(t_0) = G'(t_0; d, r, s) = \frac{s}{(d - t_0)^2} \tag{2.25b}$$

Dies sind lokale rationale Annäherungen für $\Psi$ und $\Phi$ an der Stelle $t_0$. Für $d$ setzt man den Wert von $d_{i+1}$ ein. Aus diesen Interpolationsbedingungen lassen sich die Parameter $p$, $q$, $r$ und $s$ berechnen.

Den neuen Näherungswert $t_1$ für $\lambda_i$ erhält man durch Lösung der Gleichung

$$-\frac{p}{q - t_1} = 1 + r + \frac{s}{d - t_1} \tag{2.26}$$

Mit dem neuen Wert $t_1$ berechnet man neue Interpolanten nach (2.25), erhält daraus wieder einen neuen Näherungs wert für $\lambda$ usw.
Insgesamt lässt sich mit diesen Vorschriften eine Folge $\{t_k\}$ von Näherungswerten erzeugen, für die gilt:

**Satz 4** *Wenn bei der Konstruktion nach Gln. (2.25) und (2.26) der Startwert $t_0$ im offenen Intervall $(d_i, \lambda_i)$ liegt, dann ist die Folge der Näherungswerte $\{t_k\}$ wohldefiniert, und es gilt $t_k < t_{k+1} < \lambda_i$ für alle $k \geq 0$. Ferner konvergiert die Folge quadratisch gegen $\lambda_i$.*

Dieser Satz wird in [BNS78] ausführlich bewiesen.

Dongarra und Sorensen [DS87] berechnen für diese Folge jeweils die Korrektur der Näherung $\tau = t_{k+1} - t_k$ und geben als Abbruchkriterien für die Iteration

$$|\omega(t_{k+1})| \leq \eta \max(|d_1|, |d_n|) \quad \text{und} \tag{2.27a}$$

$$|\tau| \leq \eta \min(|d_i - t_{k+1}|, |d_{i+1} - t_{k+1}|) \tag{2.27b}$$

an, wobei $t$ der berechnete Näherungswert der aktuellen Iteration und $\tau$ die Korrektur der vorhergehenden Iteration ist.

Während Bunch, Nielsen und Sorensen [BNS78] oder auch Dongarra und Sorensen [DS87] die Funktionen $\Psi(t)$ durch Interpolanten $F(t; p, q)$ und $\Phi(t)$ durch Interpolanten $G(t; d, r, s)$ annähern, hat sich Li [Li93] mit der Möglichkeit auseinandergesetzt, $\Psi$ durch $G$ und $\Phi$ durch $F$ zu approximieren. Er vergleicht drei Möglichkeiten: zunächst einmal ist dies "Approaching from the Left", was genau der Vorgehensweise in (2.25) und (2.26) entspricht. Im Ansatz "Approaching from the Right" wird $\Psi(t)$ von $G(t; d, r, s)$ und $\Phi(t)$ von $F(t; p, q)$ interpoliert. Die beiden Ansätze unterscheiden sich darin, dass durch die entsprechende Wahl der Interpolanten für die Funktionen $\Psi$ und $\Phi$ die erzeugte Folge der $\{t_i\}$ von „links“ bzw. von „rechts“ monoton gegen den gesuchten Wert $\lambda_i$ konvergiert (s. hierzu Abb. 2.2).
Schließlich schlägt er einen "Middle Way" vor, bei dem sowohl die Funktion $\Psi(t)$ als auch die Funktion $\Phi(t)$ durch rationale Funktionen der Form $G(t; d, r, s)$ interpoliert werden. Dies hat zur Folge, dass sich die $\{t_i\}$ zwar nicht mehr monoton an die gesuchte Nullstelle annähern, die Konvergenz aber schneller ist.

Weiterhin gibt Li ein neues Schema vor, wie die einzelnen Iterationen durchzuführen sind ("Hybrid Scheme"). Dieses Berechnungsverfahren hat sich in Bezug auf seine Geschwindigkeit und Genauigkeit etabliert und wird auch in der LAPACK-Bibliothek in den Routinen zur Eigenwertberechnung mittels des Divide & Conquer-Verfahrens benutzt (siehe auch [Rut94]).

#### 2.3.1.4 Berechnung der Eigenvektoren

Hat man die Eigenwerte $\lambda_i$ der Matrix $\mathbf{D} + \sigma \mathbf{z}\mathbf{z}^T$ berechnet, müssen noch die zugehörigen Eigenvektoren angegeben werden. In Satz 3 wurde hierzu schon die Gleichung (2.13) präsentiert. Es gibt dabei aber noch einiges zu beachten.

Es kommt nämlich nun die Tatsache ins Spiel, dass wir nur mit begrenzter Rechengenauigkeit arbeiten. Die Berechnung von $(\mathbf{D} - \lambda_i \mathbf{I})^{-1}$ kann ungenau werden, wenn andere Eigenwerte sehr nahe an $\lambda_i$ liegen. Dies hat zur Folge, dass die berechneten Eigenvektoren von nahe zusammenliegenden Eigenwerten nicht mehr orthogonal sind, was für die Stabilität des Verfahrens sehr wichtig ist.

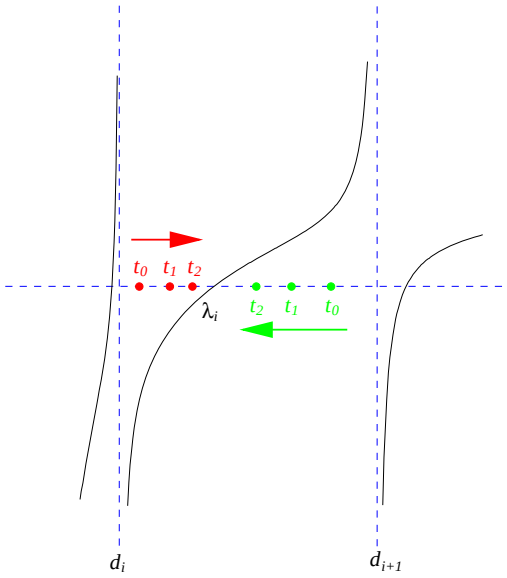



Abbildung 2.2: “Approaching from the Left” (rot) und “Approaching from the Right” (grün)

Cuppen [Cup81] schlägt vor, die Eigenvektoren von nahe beieinander liegenden Eigenwerten zu reorthogonalisieren (mittels eines modifizierten Gram-Schmidt-Verfahrens), und zeigt auch, dass das ohne Probleme möglich ist.

Dongarra und Sorensen [DS87] hingegen entscheiden sich dafür, den Deflationsprozess auszuweiten, indem sie die Frage stellen: „Wann ist ein Eigenpaar von $\mathbf{D}$ eine gute *Annäherung* eines Eigenpaars der modifizierten Matrix?“ Dies ist dann der Fall, wenn — in Anlehnung an die in Abschnitt 2.3.1.2 vorgestellten Kriterien — Komponenten $z_i$ „beinahe Null“ oder Eigenwerte „annähernd gleich“ sind. Aber auch diese Vorgehensweise einer erweiterten Deflation kann eine Orthogonalität der berechneten Eigenvektoren in allen Fällen nicht garantieren.
Weitere Vorschläge, diese Problematik zu umgehen, bestehen in der Benutzung von erweiterter Genauigkeit bei den Rechnungen.

Gu und Eisenstat [GE94] machen schließlich folgende Überlegungen zur Berechnung der Eigenvektoren:

Grundsätzlich können die Eigenvektoren nach Gleichung (2.13) mit hoher relativer Genauigkeit berechnet werden, vorausgesetzt, der berechnete Eigenwert $\lambda_i$ ist exakt.
Tatsächlich wird der berechnete Wert $\tilde{\lambda}_i$ nur eine Annäherung an den exakten Eigenwert $\lambda_i$ sein. Wenn man aber nun einen Vektor $\tilde{\mathbf{z}}$ findet, so dass die $\{\tilde{\lambda}_i\}$ die exakten Eigenwerte von $\mathbf{D} + \sigma\tilde{\mathbf{z}}\tilde{\mathbf{z}}^T$ wären, dann könnten die zugehörigen Eigenvektoren $\tilde{\mathbf{q}}^i$ der modifizierten Matrix auf Grund der vorhergehenden Überlegung mit hoher relativer Genauigkeit berechnet werden, was zur Folge hätte, dass diese numerisch orthogonal wären.
Tatsächlich kann man zeigen, dass ein solcher Vektor $\tilde{\mathbf{z}}$ immer existiert und dass dieser dicht bei $\mathbf{z}$ ist. Das hat wiederum zur Folge, dass die Eigenpaare $(\tilde{\lambda}_i, \tilde{\mathbf{q}}^i)$ der Matrix $\mathbf{D} + \sigma\tilde{\mathbf{z}}\tilde{\mathbf{z}}^T$ auch Eigenpaare der Matrix $\mathbf{D} + \sigma\mathbf{z}\mathbf{z}^T$ sind.

Die Berechnung der Eigenvektoren auf diese Art ergibt schließlich einen stabilen Algorithmus, welcher auch in die LAPACK-Bibliothek aufgenommen wurde.

### 2.3.2 Zerlegung tridiagonaler Matrizen

Basierend auf den vorangegangenen Überlegungen entwickelte Cuppen [Cup81] eine Vorgehensweise, wie man das Eigenproblem speziell für symmetrische tridiagonale Matrizen lösen kann.

Wir betrachten nun eine tridiagonale Matrix $\mathbf{T} \in \mathbb{R}^{n \times n}$:

$$\mathbf{T} = \left(\begin{array}{cccccc|cccccc} \alpha_1 & \beta_1 & & & & & & & & & & 0 \\ \beta_1 & \alpha_2 & \beta_2 & & & & & & & & & \\ & \ddots & \ddots & \ddots & & & & & & & & \\ & & \beta_{k-2} & \alpha_{k-1} & \beta_{k-1} & & & & & & & \\ & & & \beta_{k-1} & \alpha_k & \beta_k & & & & & & \\ \hline & & & & \beta_k & \alpha_{k+1} & \beta_{k+1} & & & & & \\ & & & & & \beta_{k+1} & \alpha_{k+2} & \beta_{k+2} & & & & \\ & & & & & & \ddots & \ddots & \ddots & & & \\ & & & & & & & \beta_{n-2} & \alpha_{n-1} & \beta_{n-1} & \\ 0 & & & & & & & & \beta_{n-1} & \alpha_n & \end{array}\right)$$

Mittels des in (2.9) angegebenen Schemas lässt sich $\mathbf{T}$ zerlegen:

$$\mathbf{T} = \left(\begin{array}{c|c} \mathbf{T}_1 & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{T}_2 \end{array}\right) + \beta_k \left(\begin{array}{c|c} 1 & 1 \\ \hline 1 & 1 \end{array}\right) = \left(\begin{array}{c|c} \mathbf{T}_1 & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{T}_2 \end{array}\right) + \beta_k \mathbf{b}\mathbf{b}^T \tag{2.28}$$

$$\text{mit } b_i = \begin{cases} 1 & \text{wenn } i = k \text{ oder } i = k+1 \\ 0 & \text{sonst} \end{cases}$$

Die Teilmatrizen, deren Eigenwerte und Eigenvektoren nun berechnet werden müssen, haben die Form

$$\mathbf{T}_1 = \begin{pmatrix} \alpha_1 & \beta_1 & & & 0 \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{k-2} & \alpha_{k-1} & \beta_{k-1} \\ 0 & & & \beta_{k-1} & \alpha_k - \beta_k \end{pmatrix}$$

$$\mathbf{T}_2 = \begin{pmatrix} \alpha_{k+1} - \beta_k & \beta_{k+1} & & & 0 \\ \beta_{k+1} & \alpha_{k+2} & \beta_{k+2} & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{n-2} & \alpha_{n-1} & \beta_{n-1} \\ 0 & & & \beta_{n-1} & \alpha_n \end{pmatrix}$$

Nehmen wir nun an, dass die Lösung der beiden Eigenprobleme gegeben sei mit

$$\begin{aligned} \mathbf{T}_1 &= \mathbf{Q}_1 \mathbf{D}_1 {\mathbf{Q}_1}^T \\ \mathbf{T}_2 &= \mathbf{Q}_2 \mathbf{D}_2 {\mathbf{Q}_2}^T \end{aligned} \tag{2.29}$$

Nun lässt sich die gleiche Rechnung wie in (2.10) durchführen und man erhält:

$$\begin{aligned} \mathbf{T} &= \left(\begin{array}{c|c} \mathbf{T}_1 & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{T}_2 \end{array}\right) + \beta_k \mathbf{b}\mathbf{b}^T \\ &= \left(\begin{array}{c|c} \mathbf{Q}_1 & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{Q}_2 \end{array}\right) \left( \left(\begin{array}{c|c} \mathbf{D}_1 & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{D}_2 \end{array}\right) + 2\beta_k \mathbf{z}\mathbf{z}^T \right) \left(\begin{array}{c|c} \mathbf{Q}_1 & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{Q}_2 \end{array}\right)^T \end{aligned} \tag{2.30}$$

**z** ist hierbei normiert (daher der Faktor 2) und berechnet sich zu

$$\mathbf{z} = \frac{1}{\sqrt{2}} \left( \begin{array}{c|c} \mathbf{Q}_1^T & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{Q}_2^T \end{array} \right) \mathbf{b} = \frac{1}{\sqrt{2}} \left( \begin{array}{c|c} \mathbf{Q}_1^T & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{Q}_2^T \end{array} \right) \begin{pmatrix} 0 \\ \vdots \\ 1 \\ 1 \\ \vdots \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{l} \\ \hline \mathbf{f} \end{pmatrix} \tag{2.31}$$

Der Vektor **l** ist dabei die letzte Spalte von ${\mathbf{Q}_1}^T$ und **f** ist die erste Spalte von ${\mathbf{Q}_2}^T$.

Die Matrix **T** lässt sich also als eine Rang-Eins-Änderung der zerlegten Matrix $\tilde{\mathbf{T}}$ formulieren. Eigenwerte und Eigenvektoren sind somit nach den in Abschnitt 2.3.1 erläuterten Vorgehensweisen zu berechnen.

# Kapitel 3

# Tridiagonalisierung symmetrischer Matrizen

Wie sich herausstellt, ist der notwendige Tridiagonalisierungsschritt vor der Anwendung des Divide & Conquer-Verfahrens ein möglicher Flaschenhals im Lauf der Berechnung (siehe hierzu auch [BSL94]), wenn dieser seriell ausgeführt wird, wie es z. B. im Householder-Verfahren der Fall ist. Jedoch birgt auch die Tridiagonalisierung das Potential, durch eine Parallelisierung dieses Vorgangs beschleunigt ausgeführt werden zu können.

## 3.1 Tridiagonalisierung mit dem Householder-Verfahren

### 3.1.1 Householder-Matrizen

Wir stellen folgende Aufgabe: es sei ein Vektor $\mathbf{x} \in \mathbb{R}^n$ gegeben. Gesucht ist eine Matrix $\mathbf{P}$, die den Vektor $\mathbf{x}$ in ein Vielfaches des ersten Einheitsvektors transformiert, d. h.

$$\mathbf{P}\mathbf{x} = k\mathbf{e}_1 \tag{3.1}$$

Man kann diese Aufgabe auch so auffassen, dass man eine Matrix sucht, die, angewandt auf $\mathbf{x}$, bis auf die erste Komponente alle weiteren Einträge eliminiert.

Eine Lösung dieser Aufgabe besteht darin, für $\mathbf{P}$ eine speziell konstruierte *Householder-Matrix* zu verwenden. Householder-Matrizen besitzen die Form

$$\mathbf{P} = \mathbf{I} - 2\mathbf{w}\mathbf{w}^T \text{ mit } \|\mathbf{w}\|_2 = 1 \tag{3.2}$$

Man kann zeigen, dass Householder-Matrizen sowohl unitär als auch hermitesch sind.

Für die Aufgabe aus (3.1) wählen wir nun

$$\mathbf{w} = \frac{\mathbf{x} - k\mathbf{e}_1}{\|\mathbf{x} - k\mathbf{e}_1\|_2} \quad \text{wobei } k = \begin{cases} -\text{sign}(x_1) \cdot \|x\|_2 & \text{wenn } x_1 \neq 0 \\ -\|x\|_2 & \text{wenn } x_1 = 0 \end{cases} \tag{3.3}$$

dann leistet die nach (3.2) gebildete Matrix $\mathbf{P}$ das Verlangte.

### 3.1.2 Das Householder-Verfahren

Wir wollen nun eine symmetrische Matrix auf Tridiagonalgestalt bringen. Dazu nutzen wir Householder-Matrizen, um in der Matrix spalten- bzw. zeilenweise alle Einträge bis auf diejenigen in der Diagonalen und Nebendiagonalen zu eliminieren.

Nehmen wir an, die Matrix $\mathbf{A}_{i-1}$ besitze folgende Form:

$$\mathbf{A}_{i-1} = \left(\begin{array}{c|c|c} \mathbf{T}_{i-1} & \mathbf{b} & \mathbf{0} \\ \hline \mathbf{b}^T & d_i & \mathbf{x}_i{}^T \\ \hline \mathbf{0} & \mathbf{x}_i & \tilde{\mathbf{A}}_{i-1} \end{array}\right) \tag{3.4}$$

Hierbei ist $\mathbf{T}_{i-1}$ eine tridiagonale $(i-1)\times(i-1)$-Matrix und der Vektor $\mathbf{b} \in \mathbb{R}^{i-1}$ hat außer der letzten Komponente nur Nulleinträge: $\mathbf{b} = (0,\ldots,0,b_{i-1})^T$.
Wir wählen nun als Matrix $\tilde{\mathbf{P}}_i$ diejenige Householder-Matrix, für die $\tilde{\mathbf{P}}_i\mathbf{x}_i = b_i\mathbf{e}_1$ ist und definieren dann die Matrix $\mathbf{P}_i$ als

$$\mathbf{P}_i = \left(\begin{array}{c|c|c} \mathbf{I}_{i-1} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & 1 & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \tilde{\mathbf{P}}_i \end{array}\right) \tag{3.5}$$

Da $\tilde{\mathbf{P}}_i$ unitär und hermitesch ist, gilt dies auch für $\mathbf{P}_i$. Nun ist

$$\begin{aligned} \mathbf{P}_i^{-1}\mathbf{A}_{i-1}\mathbf{P}_i = \mathbf{P}_i\mathbf{A}_{i-1}\mathbf{P}_i &= \left(\begin{array}{c|c|c} \mathbf{T}_{i-1} & \mathbf{b} & \mathbf{0} \\ \hline \mathbf{b}^T & d_i & \mathbf{x}_i{}^T\tilde{\mathbf{P}}_i \\ \hline \mathbf{0} & \tilde{\mathbf{P}}_i\mathbf{x}_i & \tilde{\mathbf{P}}_i\tilde{\mathbf{A}}_{i-1}\tilde{\mathbf{P}}_i \end{array}\right) \\ &= \left(\begin{array}{ccc|c|c|c} & & & 0 & & \\ & \mathbf{T}_{i-1} & & \vdots & & \mathbf{0} \\ & & & 0 & & \\ & & & b_{i-1} & & \\ \hline 0 & \cdots & 0\ b_{i-1} & d_i & b_i & \mathbf{0} \\ \hline & & & b_i & & \\ \hline & \mathbf{0} & & \mathbf{0} & & \tilde{\mathbf{P}}_i\tilde{\mathbf{A}}_{i-1}\tilde{\mathbf{P}}_i \end{array}\right) \\ &= \left(\begin{array}{c|c|c} \mathbf{T}_i & \mathbf{b}' & \mathbf{0} \\ \hline \mathbf{b}'^T & d_{i+1} & \mathbf{x}_{i+1}{}^T \\ \hline \mathbf{0} & \mathbf{x}_{i+1} & \tilde{\mathbf{A}}_i \end{array}\right) =: \mathbf{A}_i \end{aligned} \tag{3.6}$$

und man hat somit eine neue Matrix $\mathbf{A}_i$ erhalten, die in (3.4) beschriebene Form besitzt.

Zur Tridiagonalisierung einer $n\times n$-Matrix $\mathbf{A}$ setzt man $\mathbf{A}_0 := \mathbf{A}$ und berechnet nach obiger Bildungsvorschrift nacheinander die Matrizen $\mathbf{A}_i = \mathbf{P}_i^{-1}\mathbf{A}_{i-1}\mathbf{P}_i$ für $i = 1,\ldots,n-2$. Die Matrix $\mathbf{A}_{n-2}$ besitzt dann die gewünschte Tridiagonalgestalt.
Die Gesamttransformation $\mathbf{Q}$, die die Matrix $\mathbf{A}$ tridiagonalisiert, lautet also $\mathbf{Q} = \mathbf{P}_1\mathbf{P}_2\ldots\mathbf{P}_{n-2}$, und es gilt $\mathbf{Q}^{-1}\mathbf{A}\mathbf{Q} = \mathbf{T}$, wobei in $\mathbf{T}$ auf der Diagonalen die Werte $d_1, d_2, \ldots, d_n$ und auf den Nebendiagonalen die Werte $b_1, b_2, \ldots, b_{n-1}$ stehen.

## 3.2 Tridiagonalisierung durch Bandreduktion

Im Gegensatz zum Householder-Verfahren, wo in einer Spalte alle Subdiagonalen auf einen Schlag eliminiert werden, wird hier die Bandbreite der Matrix schrittweise verringert, bis schließlich nur

noch eine tridiagonale Matrix übrig bleibt (s. Abb. 3.1). Die Bandbreite einer Matrix gibt an, wieviele Diagonalen der Matrix Elemente ungleich Null besitzen. So hat eine Matrix $\mathbf{A}$ mit der Bandbreite $b$ nur von Null verschiedene Einträge in der Hauptdiagonalen und den ersten $b-1$ Nebendiagonalen oder anders ausgedrückt: $A_{ij} = 0$ für $|i-j| \geq b$. Eine voll besetzte $n \times n$-Matrix besitzt somit die Bandbreite $n$.

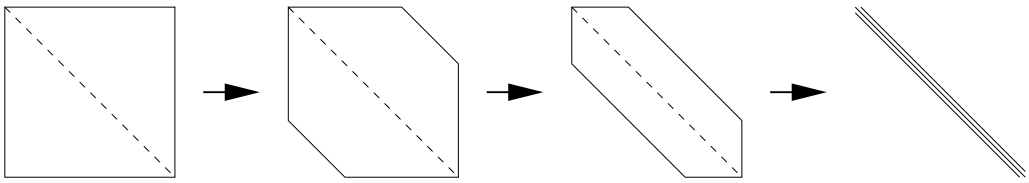


Abbildung 3.1: Verkleinerung der Bandbreite bis zur Tridiagonalform

Die Vorgehensweise der Tridiagonalisierung durch Bandreduktion wurde ursprünglich für Bandmatrizen von Lang [Lan91] entwickelt und später von Bischof et al. [BMS93, BSL94] für voll besetzte Matrizen erweitert. Sie soll im Rest dieses Kapitels erläutert werden.

### 3.2.1 Eine Bandreduktion

Angenommen, wir haben eine Matrix $\mathbf{A}$ der Bandbreite $b$ und wollen $d$ Nebendiagonalen entfernen, so dass die reduzierte Matrix die Bandbreite $b-d$ besitzt. Die Nebendiagonalen von der $b-d+1$-ten bis zur $b$-ten Nebendiagonale bilden zusammen das Band, das eliminiert werden soll.

Die Reduktion dieses Bandes beinhaltet die Elimination aller seiner Spalten. Diese werden der Reihe nach durch Householder-Transformationen eliminiert. In der $j$-ten Spalte stehen die Elemente des hier betrachteten Bandes in den Zeilen $j+b-d$ bis $j+b-1$. Wir wählen daher den Vektor $\mathbf{x} \in \mathbb{R}^{d+1}$ so, dass $x_i = A_{i+j+b-d-2,j}$ mit $i = 1, \ldots, d+1$ ist und konstruieren daraus eine Householder-Matrix nach den Gln. (3.1), (3.2) und (3.3).

Durch die so konstruierte Transformation wird die betreffende Spalte zwar eliminiert, aber an einer anderen Stelle in der Matrix wird i. A. ein „Bulge“, also eine Ausbuchtung erzeugt, die die Bandbreite der Matrix sogar vergrößert (s. Abb. 3.2). Der Grund, warum wir hier nur die Elemente unterhalb der Hauptdiagonalen betrachten, wird in Abschnitt 3.2.2 deutlich werden.

Der nächste Schritt besteht nun darin, die erste Spalte des entstandenen Bulges entfernen, und zwar wie oben auch durch eine Householder-Transformation[1], wodurch jedoch erneut ein weiterer Bulge entstehen kann (s. Abb. 3.3). Von diesem wird ebenso die erste Spalte eliminiert, und der Vorgang setzt sich fort, bis kein neuer Bulge mehr entsteht[2].

Ist dieser Punkt erreicht, sieht die Matrix aus wie in Abb. 3.4 dargestellt. Man erkennt die entstandenen Bulges, die über die ursprüngliche Bandbreite der Matrix hinausragen. Nun kann man mit der Elimination der zweiten Spalte fortfahren, die wie die Elimination der ersten Spalte erneut einen Bulge erzeugt. Die Entfernung der ersten Spalte dieses Bulges hat einen weiteren entstehenden Bulge zur Folge usw.

[1] Wie man berechnen kann, welche Einträge in der Matrix eliminiert werden müssen, wird in Abschnitt 4.2.1.2 beschrieben

[2] Ein passender Vergleich hierzu ist eine Teppichfalte, die man bis zum Rand des Teppichs schieben muss, bis sie endlich verschwindet

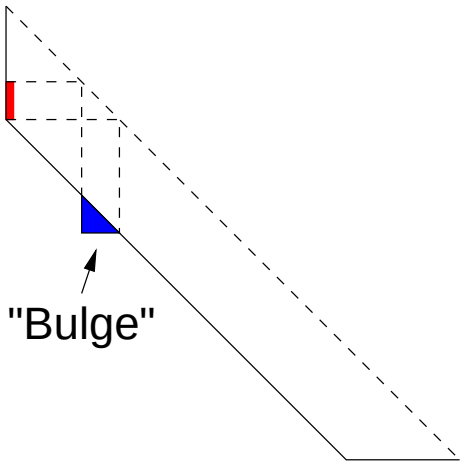



Abbildung 3.2: Elimination der ersten Spalte des Bandes

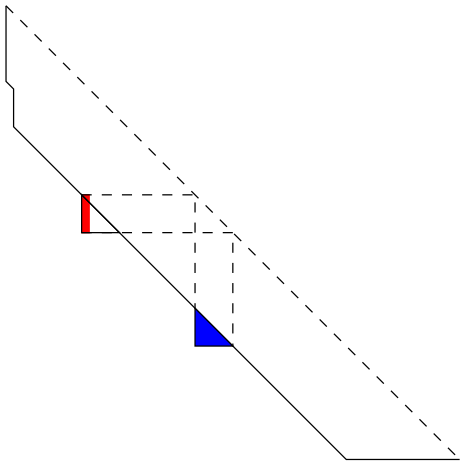


Abbildung 3.3: Elimination der ersten Spalte des Bulges

Jetzt wird auch deutlich, warum man von jedem Bulge nur die erste Spalte entfernt und nicht etwa den ganzen Bulge vollständig. Bei einer nachfolgenden Spaltenelimination würde nämlich der Bereich des gerade mühsam entfernten Bulges ohnehin wieder mit Einträgen gefüllt, so dass eine vollständige Entfernung der Bulges völlig unnötig ist.

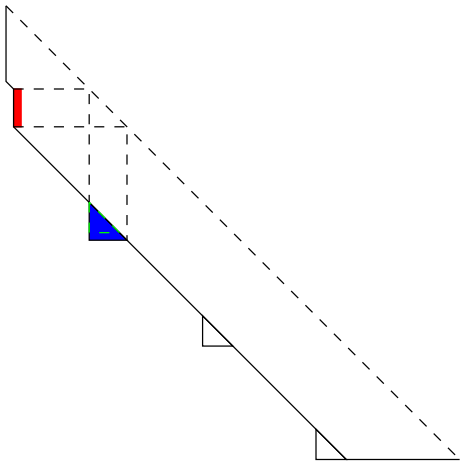


Abbildung 3.4: Elimination der nächsten Spalte

Im Folgenden wollen wir gelegentlich genau unterscheiden, ob sich eine zu eliminierende Spalte im zu reduzierenden Band oder in einem Bulge befindet. Im ersten Fall bezeichnen wir die Spalte als *Bandspalte*, im letzten Fall als *Bulgespalte*. Der Begriff Bandspalte wird auch später dazu benötigt werden, um einzelne Arbeitsschritte bei der Bandreduktion zu unterscheiden (s. Abschnitt 4.2.1.1).

### 3.2.2 Die verwendete Rechenprimitive

Sowohl die Elimination einer Bandspalte als auch die Entfernung einer Bulgespalte erfordern die gleiche Rechenoperation, die wir mit *Zeroout* bezeichnen. Diese beinhaltet folgende Schritte

1. Die Elimination der Spalte mittels einer Householder-Transformation
2. Die Anwendung der Transformation auf die Matrix von links
3. Die Anwendung der Transformation auf die Matrix von links und rechts
4. Die Anwendung der Transformation auf die Matrix von rechts

Die Unterscheidungen in den Schritten 2 bis 4 rühren daher, dass wir mit einer symmetrischen Matrix arbeiten und diese nach einer Householder-Elimination symmetrisch bleibt, so dass wir hier nur das untere Dreieck der Matrix betrachten und bearbeiten müssen. Diese Householder-Transformation verändert weiterhin bei der Multiplikation von links an die zu tridiagonalisierende Matrix nur wenige Zeilen bzw. bei Multiplikation von rechts nur wenige Spalten der Matrix. Die Bereiche der Matrix, die in den einzelnen Schritte geändert werden, sind in Abb. 3.5 dargestellt.

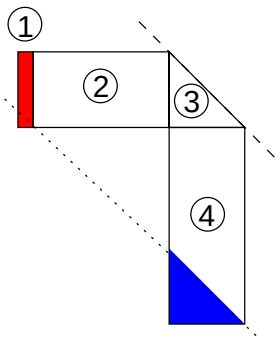



Abbildung 3.5: Die Rechenprimitive Zeroout

### 3.2.3 Aufeinander folgende Bandreduktionen

Um eine symmetrische $n \times n$-Matrix mittels Bandreduktionen zu tridiagonalisieren, ist der Vorschlag von Bischof, Marques und Sun [BMS93], durch aufeinander folgende Bandreduktionen die Bandbreite der Matrix schrittweise so lange zu verringern, bis sie schließlich nur noch 2 beträgt. Wir müssen also eine Folge von $k$ Bandreduktionen durchführen, so dass, wenn bei der $i$-ten Bandreduktion $d_i$ weitere Subdiagonalen eliminiert werden, die Gesamtanzahl eliminierter Subdiagonalen $\sum_{i=1}^{k} d_i = n - 2$ ergibt.

### 3.2.4 Parallelität

Im Laufe dieses Rechenverfahrens bieten sich Möglichkeiten, mehrere der Zeroout-Operationen gleichzeitig durchzuführen, nämlich dann, wenn sich die jeweils bearbeiteten Teile der Matrix nicht überlappen. Dies kann schon der Fall sein, wenn man eine einzelne Bandreduktion für sich betrachtet oder, wenn man den „scheibchenweisen“ Ansatz aus Abschnitt 3.2.3 verfolgt, bei der gleichzeitigen Reduktion benachbarter Bänder.

Wir wollen nun diese zwei Fälle unterscheiden:

1. Parallelität innerhalb einer Bandreduktion: Wenn die Bulge-Elimination bei der Entfernung einer Spalte weit genug fortgeschritten ist, dann kann man bereits mit der Elimination der nächsten Bandspalte beginnen (s. Abb. 3.6). Ebenso kann die Elimination einer Bulgespalte durchgeführt werden, wenn der Bulge der vorhergehenden Spalte genügend weit entfernt ist.

2. Parallelität bei benachbarten Bändern: Wenn genügend Bandspalten eines Bandes eliminiert worden sind, kann man schon mit der Reduktion des nächsten Bandes beginnen (s. Abb. 3.7).

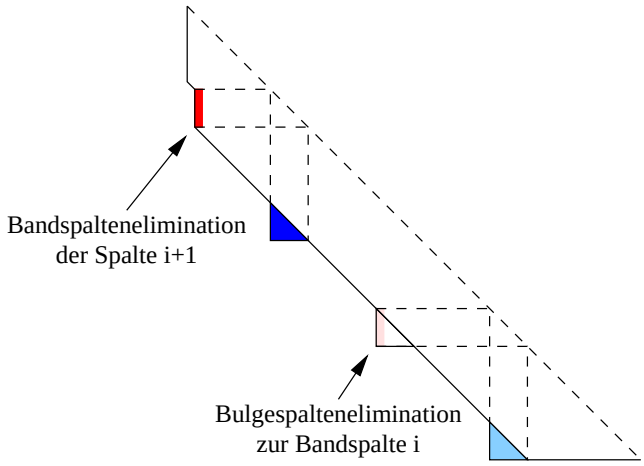



Abbildung 3.6: Parallelität innerhalb einer Bandreduktion

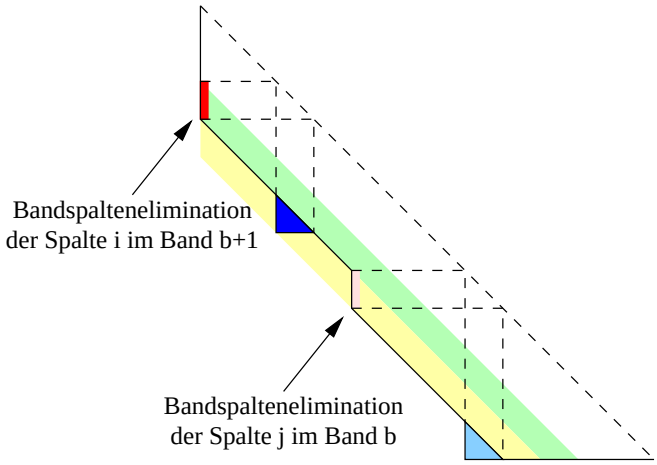



Abbildung 3.7: Parallelität bei benachbarten Bändern

# Kapitel 4

# Implementation

## 4.1 Das Divide & Conquer-Verfahren

Fassen wir noch einmal kurz die schon in Abschnitt 2.3.2 erläuterten Arbeitsschritte eines Divide & Conquer-Schrittes zur Berechnung der Eigenwerte und Eigenvektoren einer symmetrischen tridiagonalen Matrix zusammen:

- Zerlege die Matrix $\mathbf{T}$ in $\tilde{\mathbf{T}}$
- Löse das Eigenproblem der Matrizen $\mathbf{T}_1$ und $\mathbf{T}_2$
- Berechne die korrigierten Eigenwerte und Eigenvektoren von $\mathbf{T}$ durch Lösung der Säkulargleichung

Dabei ist nicht zu vergessen, dass die Fock-Matrizen, deren Eigenwerte und Eigenvektoren wir berechnen wollen, nicht tridiagonal sind, d. h. es ist vorher noch ein Tridiagonalisierungsschritt durchzuführen. Die Tridiagonalisierung soll hierbei nach dem Householder-Verfahren ablaufen, eine Implementation der Tridiagonalisierung durch parallele Bandreduktion wird in Abschnitt 4.2 präsentiert.

### 4.1.1 Zerlegung der Matrix

Die Zerlegung der Matrix kann, wie für ein Divide & Conquer-Verfahren üblich, rekursiv erfolgen. Durch die fortgesetzte Aufteilung der Matrix mittels Halbierung wird ein Binärbaum aufgespannt (s. Abb. 4.1), an dessen Blättern sich schließlich entweder $1 \times 1$- oder $2 \times 2$-Matrizen befinden, deren Eigenwerte und Eigenvektoren trivial zu berechnen sind, oder man beendet die Zerlegung bei kleinen $m \times m$-Matrizen, deren Eigenwerte und Eigenvektoren „herkömmlich“, z. B. mit dem QR-Verfahren berechnet werden (vgl. [Cup81]). Diese Matrizen wollen wir im Folgenden *Blattmatrizen* nennen.

Wir wollen nun eine $n \times n$-Matrix zerlegen, so dass die entstehenden Blattmatrizen maximal die Größe $m \times m$ besitzen. Angenommen, es seien dazu $h$ Schritte notwendig, dann erhält man $2^h$ Blattmatrizen, wobei $d_i^{(h)}$ die Dimension der $i$-ten Blattmatrix sei.
Man beginnt daher mit $d_1^{(0)} = n$. Falls nach dem $h$-ten Zerlegungsschritt $d_i^{(h)} \leq m$ für alle $i =$

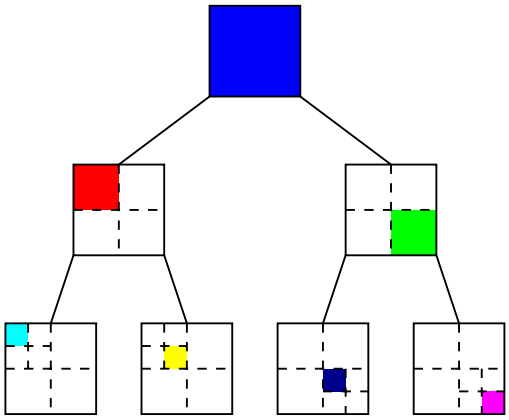


Abbildung 4.1: Aufgespannter Binärbaum beim Zerlegen der Matrix

$1, \ldots, 2^h$ gilt, ist man mit der Zerlegung fertig. Ansonsten werden die Matrizen folgendermaßen weiter aufgeteilt:

$$\begin{aligned} d_{2i-1}^{(h+1)} &= \left\lfloor \frac{d_i^{(h)}}{2} \right\rfloor \\ d_{2i}^{(h+1)} &= \left\lfloor \frac{d_i^{(h)} + 1}{2} \right\rfloor \end{aligned} \qquad \text{für } i = 1, \ldots, 2^h \tag{4.1}$$

### 4.1.2 Verschmelzung von Tridiagonalisierung und Lösung der Teilprobleme

Nachdem die Größe und Lage der Teilprobleme auf Blattebene festgelegt ist, geht es nun daran, die Tridiagonalisierung und die Lösung der Eigenwertprobleme der Blattmatrizen durchzuführen. Die Tridiagonalisierung soll, wie schon erwähnt, nach dem Householder-Verfahren ablaufen. Es werden daher von einem Matrixende beginnend nacheinander alle Spalten (und damit auch alle Zeilen) eliminiert. Nun muss die Tridiagonalisierung nicht vollständig durchgeführt worden sein, bevor man mit der Bearbeitung der Blattmatrizen beginnen kann. Ist nämlich eine Blattmatrix fertig tridiagonalisiert, dann ändern sich deren Einträge bei den nachfolgenden Householder-Schritten nicht mehr, so dass mit der Lösung des Eigenwertproblems der betreffenden Blattmatrix unmittelbar begonnen werden kann.
In Abb. 4.2 ist dies illustriert, wobei dort die Elimination der Spalten bei der höchsten Spalte beginnt.

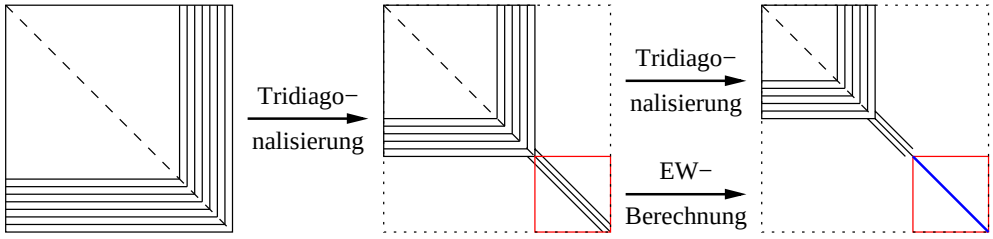



Abbildung 4.2: Parallele Tridiagonalisierung und Lösung der Eigenwertprobleme

Im weiteren Verlauf können zwei benachbarte Blattmatrizen, deren Eigenprobleme gelöst wurden, sofort miteinander verschmolzen werden, während die Eigenwertprobleme für andere Blattmatrizen noch gelöst werden und sogar, während die Tridiagonalisierung noch nicht abgeschlossen ist (s. auch Abb. 4.3).

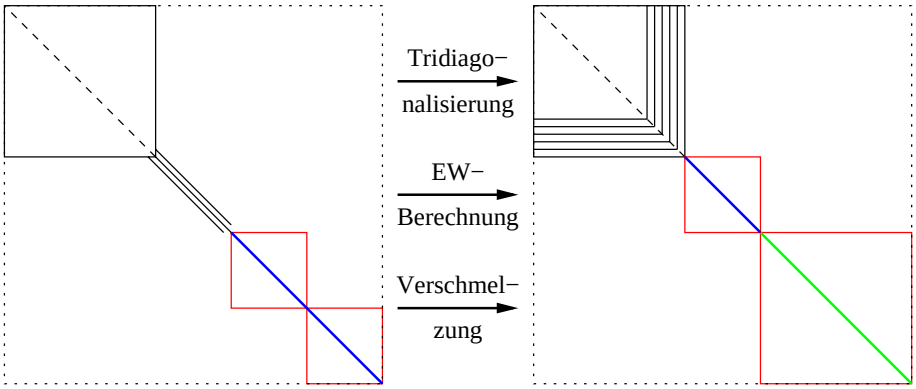



Abbildung 4.3: Verschmelzung zweier Blattmatrizen

### 4.1.3 Ein threadbasierter Ansatz

Die Zerlegung der Matrix liefert $2^h$ Blattmatrizen. Die grundsätzliche Idee besteht nun darin, dass jede Blattmatrix von einem Thread bearbeitet wird. Dieser ist zunächst für die Tridiagonalisierung und die Eigenwertberechnung seines Teilstücks zuständig. Jeder Thread erhält als eindeutige Thread-Id die Nummer der von ihm bearbeiteten Teilmatrix auf Blattebene, wobei die Blattmatrizen von 1 bis $2^h$ durchnummeriert werden.
Das Zusammenfassen der Ergebnisse der Teilstücke geschieht wie folgt: wir nummerieren die einzelnen Ebenen des Binärbaums aus Abb. 4.1 fortlaufend, wobei die Ebene mit den Blattmatrizen die Nummer 1 erhält und die Ebene des Wurzelknotens die Nummer $h+1$. Die Conquer-Schritte, die im Baum auf Ebene $i$ ausgeführt werden müssen, werden von den Threads erledigt, deren Thread-Id genau 1 modulo $2^i$ ist.
Der Ablauf ist für eine Zerlegung mit $h = 3$ in Abb. 4.4 dargestellt.

Dieser Ansatz lässt sich sehr leicht z. B. mittels POSIX-Threads implementieren. So kann Thread $i$, nachdem dieser seinen Tridiagonalisierungsschritt abgeschlossen hat, gezielt den Thread $i + 1$ starten. Die Synchronisation bei den Conquer-Schritten findet statt, indem der Thread, der die Verschmelzung ausführen soll, einfach die Beendigung des Nachbarthreads abwartet. Durch den Abschluss von Thread 1 ist gleichzeitig auch die gesamte Eigenwert- und Eigenvektorberechnung beendet.

### 4.1.4 Ein taskbasierter Ansatz

Im Gegensatz zu dem threadbasierten Ansatz, wo für jeden Thread von Anfang an genau festgelegt ist, welche Operationen dieser nacheinander auszuführen hat, legen wir nun fest, welche Aufgaben oder auch *Tasks* insgesamt durchzuführen sind und in welcher Reihenfolge dies geschehen muss. Die Ausführung der Tasks geschieht i. A. wieder durch Threads, allerdings ist für einen bestimmten Thread nicht von vornherein bestimmt, welche Tasks dieser im Einzelnen nacheinander abarbeitet.

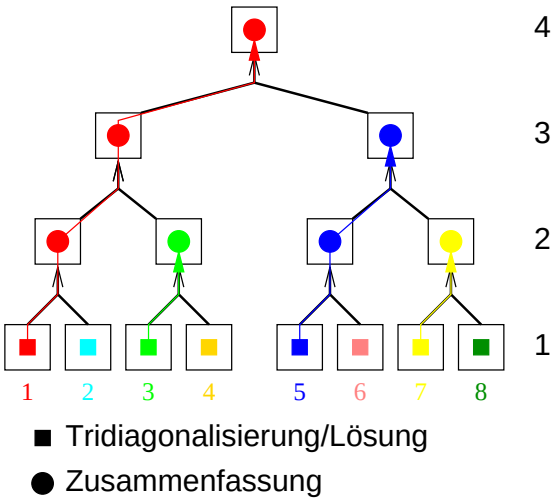



Abbildung 4.4: Ablauf bei Bearbeitung durch Threads

Für die Ausführung des Divide & Conquer-Verfahrens definieren wir folgende Tasks:

**TRD**$(i)$ Die Tridiagonalisierung des Teilstücks $i$ auf Blattebene (Die Blattmatrizen werden wie im threadbasierten Ansatz von 1 bis $2^h$ durchnummeriert).

**SOLVE**$(i)$ Die Berechnung der Eigenwerte und Eigenvektoren der Blattmatrix $i$.

**MERGE**$(l, i)$ Die Zusammenfassung der Ergebnisse auf Baumhöhe $l$. Hierbei entspricht, anders als im vorigen Abschnitt, die Blattebene des Baumes der Höhe 0 und die Ebene des Wurzelknotens der Höhe $h$, wenn $h$ die Anzahl der Zerlegungen angibt.
Wenn $l = 1$ ist, dann werden die Blattmatrizen $2i - 1$ und $2i$ zusammengefasst, andernfalls die durch die Tasks MERGE$(l-1, 2i-1)$ und MERGE$(l-1, 2i)$ zusammengefassten Matrizen.

Die Abhängigkeiten der Tasks untereinander sind dann wie folgt gegeben:

- Task TRD$(i+1)$ kann erst nach Beendung des Tasks TRD$(i)$ ausgeführt werden (für $i = 1, \ldots, 2^h - 1$)
- Task SOLVE$(i)$ kann erst nach Beendung des Tasks TRD$(i)$ ausgeführt werden (für $i = 1, \ldots, 2^h$)
- Task MERGE$(1, (i+1)/2)$ kann erst nach Beendung der Tasks SOLVE$(i)$ und SOLVE$(i+1)$ ausgeführt werden (für $i = 1, 3, \ldots, 2^h - 1$)
- Task MERGE$(l+1, (i+1)/2)$ kann erst nach Beendung der Tasks MERGE$(l, i)$ und MERGE$(l, i+1)$ ausgeführt werden (für $i = 1, 3, \ldots, 2^{h-l+1} - 1$)

Die Abhängigkeiten sind für eine Zerlegung mit $h = 3$ in Abb. 4.5 dargestellt.

#### 4.1.4.1 Verwaltung der Tasks

Die Verwaltung der Tasks geschieht zweckmäßig über Warteschlangen, in der die zur Ausführung bereiten Tasks stehen. Die Warteschlangen besitzen dabei die FIFO-Eigenschaft.

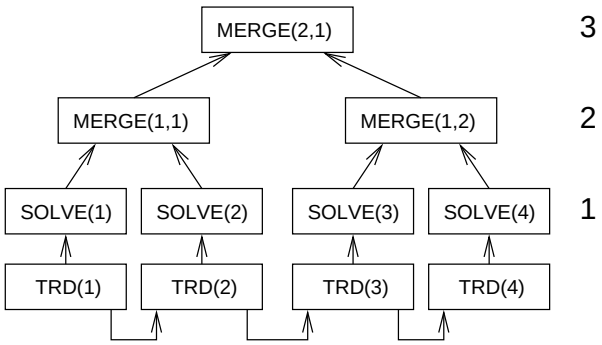



Abbildung 4.5: Abhängigkeiten der Tasks untereinander

Für die SOLVE- und MERGE-Tasks benutzt man dabei zwei verschiedene Warteschlangen. Der Grund dafür ist, dass den SOLVE-Tasks eine höhere Priorität gegenüber den MERGE-Tasks eingeräumt werden soll.

Eine Ausnahme bilden die TRD-Tasks. Da diese ohnehin seriell abgearbeitet werden müssen und die höchste Priorität bei der Bearbeitung besitzen sollen, werden diese durch einen eigenen Thread exklusiv abgearbeitet.

#### 4.1.4.2 Ablauf

Der Thread, der für die Tridiagonalisierung der Teilstücke zuständig ist, arbeitet nacheinander alle TRD-Tasks ab. Sobald er den Task TRD($i$) ausgeführt hat, kann er den entsprechenden Task SOLVE($i$) für die Lösung des Eigenwertproblems des gerade tridiagonalisierten Teilstücks in die SOLVE-Warteschlange stellen. Sind alle TRD-Tasks beendet, arbeitet der Thread nach dem nachfolgend beschriebenen Schema weiter.

Alle anderen Threads verfahren von Anfang an nach dem folgenden Schema:

1. Wenn sich ein zur Ausführung bereiter Task in der SOLVE-Warteschlange befindet, enferne diesen aus der Warteschlange und führe ihn — dies sei nun der Task SOLVE($i$) — aus.
   Prüfe anschließend, ob die zugehörige benachbarte Blattmatrix $i+1$ bzw. $i-1$, je nachdem ob $i$ ungerade bzw. gerade ist, bereits „gelöst“ wurde. Falls dies so ist, dann stelle den Task MERGE$(1,(i+1)/2)$ bzw. MERGE$(1,i/2)$ in die MERGE-Warteschlange. Beginne dann wieder bei Punkt 1.

2. Enthält die MERGE-Warteschlange einen zur Ausführung bereiten Task MERGE$(l,i)$, dann entferne diesen aus der Warteschlange und führe ihn aus.
   Wenn für den gerade bearbeitete Task $l = h$ gilt (wobei $h$ die Anzahl der Zelegungen ist), dann ist der Algorithmus für alle Threads beendet.
   Sonst, wenn $i$ ungerade bzw. gerade ist und der Task MERGE$(l,i+1)$ bzw. der Task MERGE$(l,i-1)$ schon ausgeführt wurde, dann stelle den Task MERGE$(l+1,(i+1)/2)$ bzw. MERGE$(l+1,i/2)$ in die MERGE-Warteschlange. Fahre anschließend wieder bei Punkt 1 fort.

3. Ansonst stehen für den Thread gerade keine Tasks zur Ausführung zur Verfügung. Setze die Arbeit wieder bei Punkt 1 fort.

#### 4.1.4.3 Anmerkungen

Im Gegensatz zu dem threadbasierten Ansatz ist der Ansatz über Tasks mittels OpenMP-Direktiven umsetzbar, denn im Wesentlichen ist nur die in 4.1.4.2 beschriebene Schleife zu parallelisieren.

Unbedingt notwendig ist die Synchronisation der Threads bei Zugriffen auf die Warteschlangen. Sobald ein Thread eine Warteschlange verändern will, benötigt er exklusiven Zugriff auf diese. Glücklicherweise stellt OpenMP Lock-Variablen zur Verfügung, so dass man jede Warteschlange einzeln sperren kann.

Was OpenMP im Gegensatz zu den im threadbasierten Ansatz verwendeten POSIX-Threads nicht bieten kann, sind Condition-Variablen. Diese wären hier sehr hilfreich für den Fall, dass für einen Thread keine Tasks zur Ausführung bereit stehen. Durch Abfrage einer Condition-Variable könnte er dann solange blockieren, bis ein Task ausführbereit wird. Abgesehen davon, dass man mit den von OpenMP zur verfügung gestellten Synchronisationsmitteln, i. e. im Wesentlichen die Lock-Variablen, Condition-Variablen nachbilden könnte, bleibt die einzige Möglichkeit ein “busy wait”.

### 4.1.5 Bewertung

Ob man nun den threadbasierten oder den taskbasierten Ansatz verfolgt — was einer effizienten Parallelisierung entgegenwirkt, ist die Tridiagonalisierung nach dem Householder-Verfahren, die nur seriell ablaufen kann. Wünschenswert wäre es, wenn alle Threads gleichzeitig mit der Eigenwertberechnung der Blattmatrizen starten könnten. Dies ist auch der Grund, warum wir uns im nächsten Abschnitt genauer mit der Implementation der Tridiagonalisierung auseinander setzen werden.

Ein weiteres Potential zur Parallelisierung des Verfahrens wurde bisher noch gar nicht erwähnt. Es handelt sich dabei um die Lösung der Säkulargleichung, die in jedem Conquer-Schritt berechnet werden muss. Die Nullstellen der Säkularfunktion können nämlich alle unabhängig voneinander und damit parallel berechnet werden. Dies ist um so wichtiger, da gegen Ende des Algorithmus nur noch wenige, aber große Teilmatrizen verschmolzen werden müssen, während die Anzahl der zu berechnenden Nullstellen immer größer wird.

## 4.2 Die Tridiagonalisierung

Auch hier bietet sich ein taskbasierter Ansatz an. Es gibt nur einen einzigen Tasktyp ZEROOUT, der genau eine Rechenprimitive Zeroout, wie in Abschnitt 3.2.2 beschrieben, ausführt.

### 4.2.1 Tasks

#### 4.2.1.1 Unterscheidung der Tasks

Die Tasks unterscheiden sich voneinander in drei Punkten:

- Jeder Task ist für die Reduktion eines bestimmten Bandes notwendig, daher besitzt er eine bestimmte *Bandnummer*.
- In seinem Band ist der Task für die Elimination einer bestimmten Bandspalte zuständig. Der Index dieser Bandspalte ist die *Spaltennummer* dieses Tasks.

- Des Weiteren hat jeder Task eine spezifische *Wellennummer*. Für die Elimination einer Spalte in einem Band sind mehrere Wellen notwendig, das sind die Elimination der Bandspalte selbst (Welle 0), die Elimination der ersten Bulgespalte (Welle 1), die Elimination der zweiten Bulgespalte (i. e. die erste Spalte des zweiten Bulges) (Welle 2) usw.

Man kann daher jeden ZEROOUT-Task durch ein Zahlentripel bestehend aus Bandnummer, Spaltennummer und Wellennummer eindeutig beschreiben. Im Folgenden werden ZEROOUT-Tasks daher auch durch dessen Angabe in Form von (band, spalte, welle) unterschieden.

#### 4.2.1.2 Genauere Betrachtung der Tasks

Jeder Task eliminiert in der Matrix den Teil einer Spalte, sei es eine Band- oder eine Bulgespalte. Diese Spalte wird als *Arbeitsspalte* des Tasks bezeichnet. Von dieser Arbeitsspalte entfernt er einen bestimmten Ausschnitt, genauer gesagt alle Elemente von einer bestimmten Zeile bis zu einer bestimmten Zeile. Alle diese Größen sind aus den Kennziffern Band-, Spalten- und Wellennummer eindeutig berechenbar. Sie geben zudem eine genaue Beschreibung davon, welcher Teil der Matrix bei der Ausführung der Zeroout-Operation modifiziert wird. Dies ist wichtig, um von zwei verschiedenen Tasks genau sagen zu können, ob sie kollisionsfrei und damit unabhängig voneinander ausgeführt werden können (siehe dazu Abschnitt 4.2.3).

Konkret werden zwei Typen definiert, einer, der die Information über das zu bearbeitende Band, Spalte und Welle enthält, und ein Typ, der den Index der Arbeitsspalte mit den zugehörigen Zeilenindizes enthält. Der Grund für die Trennung dieser Information wird in Abschnitt 4.2.2 deutlich.

In den Abbildungen 4.6 und 4.7 sind die Implementationen der Datentypen in Fortran 90 angegeben. In der Definition fuer den Task-Datentyp sind zusätzlich zwei boolesche Variablen enthalten, die für die Prüfung der Kollisionsfreiheit genutzt werden, sowie ein Zeiger auf einen Spalten-Datentyp, der die ergänzende Information zu diesem Task enthalten soll, und außerdem ein Zeiger auf einen weiteren Task, so dass Warteschlange in Form einer verketteten Liste aufgebaut werden kann.
Im Spalten-Datentyp ist zusätzlich der Index der Spalte gespeichert, der der Spaltennummer des zugeordneten Tasks entspricht. Auch hier ermöglicht ein Zeiger auf den eigenen Datentyp die Erstellung einer verketteten Liste.

```
type zerooutask_t
   integer :: column, band, wave
   type(column_t), pointer :: column_info
   logical :: samebandok, prevbandok
   type(zerooutask_t), pointer :: next_task
end type zerooutask_t
```

Abbildung 4.6: Datentyp für einen Task

Sind Band-, Spalten- und Wellennummer für einen Task task bekannt, dann lassen sich die Größen

```
type column_t
   integer :: index
   integer :: workcol
   integer :: from, to
   type(column_t), pointer :: next_column
end type column_t
```

Abbildung 4.7: Datentyp für eine zu eliminierende Spalte

des zugehörigen Spaltentyps wie folgt berechnen:

$$\mathsf{task\%column\_info\%workcol} = \begin{cases} \mathsf{task\%column} & \text{wenn } \mathsf{task\%wave} = 0 \\ b - 1 - \mathsf{task\%band} \cdot d + \mathsf{task\%column} + & \\ \quad (\mathsf{task\%wave} - 1) \cdot (b - d \cdot (\mathsf{task\%band} - 1)) & \text{sonst} \end{cases} \tag{4.2}$$

$$\mathsf{task\%column\_info\%from} = b - 1 - \mathsf{task\%band} \cdot d + \mathsf{task\%column} + \mathsf{task\%wave} \cdot (b - d \cdot (\mathsf{task\%band} - 1)) \tag{4.3}$$

$$\mathsf{task\%column\_info\%to} = \min(b - 1 - (\mathsf{task\%band} - 1) \cdot d + \mathsf{task\%column} + \mathsf{task\%wave} \cdot (b - d \cdot (\mathsf{task\%band} - 1)), n) \tag{4.4}$$

Dabei ist $b$ die ursprüngliche Bandbreite der Matrix und $d$ die Breite eines Bandes. Wir gehen davon aus, dass die Breite aller zu eliminierenden Bänder gleich groß ist. Diese Annahme führt aber im Allgemeinen dazu, dass das letzte Band eine Breite kleiner als $d$ besitzt, was in den Berechnungsvorschriften (4.2), (4.3) und (4.4) nicht berücksichtigt wurde.

Die obigen Formeln geben an, wie man die Größen workcol, from und to direkt aus Band-, Spalten- und Wellennummer berechnen kann. Einfacher ist es jedoch, diese Werte inkrementell zu berechnen. Hierzu wird der allererste Task first für Band 1, Spalte 1, Welle 0 mit den folgenden Werten initialisiert

$$\begin{aligned} \mathsf{first\%column\_info\%workcol} &= 1 \\ \mathsf{first\%column\_info\%from} &= \max(2, b - d) \\ \mathsf{first\%column\_info\%to} &= b \end{aligned}$$

Ausgehend davon lassen sich für alle Tasks beim Übergang zum nächsten Band, zur nächsten Spalte bzw. zur nächsten Welle wie folgt berechnen:

- Wenn $\mathsf{task2\%band} = \mathsf{task1\%band} + 1$, $\mathsf{task2\%column} = \mathsf{task1\%column} = 1$ und $\mathsf{task2\%wave} = \mathsf{task1\%wave} = 0$, dann ist

$$\begin{aligned} \mathsf{task2\%column\_info\%workcol} &= 1 \\ \mathsf{task2\%column\_info\%from} &= \max(2, \mathsf{task1\%column\_info\%from} - d) \\ \mathsf{task2\%column\_info\%to} &= \min(\mathsf{task1\%column\_info\%to} - d, n) \end{aligned}$$

- Wenn task2%band = task1%band, task2%column = task1%column + 1 und task2%wave = task1%wave = 0, dann ist

$$\begin{aligned} \text{task2\%column\_info\%workcol} &= \text{task1\%column} + 1 \\ \text{task2\%column\_info\%from} &= \text{task1\%column\_info\%from} + 1 \\ \text{task2\%column\_info\%to} &= \min(\text{task1\%column\_info\%to} + 1, n) \end{aligned}$$

- Wenn task2%band = task1%band, task2%column = task1%column
  und task2%wave = task1%wave + 1, dann ist

$$\begin{aligned} \text{task2\%column\_info\%workcol} &= \text{task1\%column\_info\%from} \\ \text{task2\%column\_info\%from} &= \text{task1\%column\_info\%from} + b - 1 - d \cdot (\text{task1\%band} - 1) \\ \text{task2\%column\_info\%to} &= \min(\text{task1\%column\_info\%to} + b - 1 - d \cdot (\text{task1\%band} - 1), n) \end{aligned}$$

### 4.2.2 Verwaltung der Bänder

Im Hinblick auf eine leichte Überprüfung der Kollisionsfreiheit der Tasks werden für jedes zu eliminierende Band folgende Informationen separat gespeichert:

- Eine Warteschlange, die alle dieses Band betreffenden wartenden Tasks enthält. Das sind diejenigen Tasks, die auf Grund der Reihenfolge, in der die Tasks nacheinander auszuführen sind (mehr dazu in Abschnitt 4.2.3), zwar schon mit ihrer Abarbeitung beginnen könnten, dieses aber nicht erlaubt ist, da sie sonst mit einem aktuell ausgeführten Task kollidieren würden.
- Eine Warteschlange, die alle dieses Band betreffenden bereiten Tasks enthält, die sofort ausgeführt werden können. Für diese Tasks ist also sowohl die richtige Ausführungsreihenfolge als auch die Kollisionsfreiheit sichergestellt.
- Eine Liste, in der alle Bandspalten dieses Bandes aufgelistet sind, für die sich gerade ein Task in der Abarbeitung befindet bzw. bereite Tasks vorhanden sind.

Die Umsetzung dieser Anforderungen in Fortran 90 ist in Abb. 4.8 dargestellt. Die zusätzlich gespeicherte Matrix H enthält nach vollendeter Elimination des entsprechenden Bandes die orthogonale Transformation, die hierzu insgesamt notwendig war. Dies ist erforderlich für die spätere Berechnung der genau den Eigenvektoren entsprechenden Gesamttransformation zur Diagonalmatrix.

```
type bandtasks_t
   type(column_t), pointer :: active_columns
   type(zerooutask_t), pointer :: ready, waiting
   real(double), dimension(:,:), pointer :: H
end type bandtasks_t
```

Abbildung 4.8: Datentyp für ein Band

Die „Verzeigerung“ der Datenstrukturen ist in Abb. 4.9 beispielhaft illustriert. Man erkennt die beiden Warteschlangen ready und waiting für die bereiten und wartenden Tasks. Jeder Task hat einen

Zeiger auf seine zugehörige Spalteninformation, und für alle bereiten Tasks sind diese zudem in der Liste der aktiven Spalten aufgeführt, die wie die Warteschlangen für die Tasks nach dem Spaltenindex sortiert ist. Betrachtet man nur die Warteschlange der bereiten Tasks und die Liste der aktiven Spalten, so sieht man, dass für jeden bereiten Task in der Warteschlange eine Spaltenstruktur in der Liste steht, aber nicht umgekehrt zu jedem Eintrag in der Liste ein Task bereit steht. Tritt dieser Fall auf, dann bedeutet das nichts anderes, als dass zu der bestimmten Spaltenstruktur sich gerade ein Task in der Ausführung und daher nicht mehr in der Warteschlange befindet. Die Information darüber, welchen Bereich der Matrix dieser Task bearbeitet — diese befindet sich ja genau in der Spaltenstruktur — ist notwendig, um für neu ankommende Tasks dieses und des nachfolgenden Bandes die Kollisionsfreiheit sicherstellen zu können.

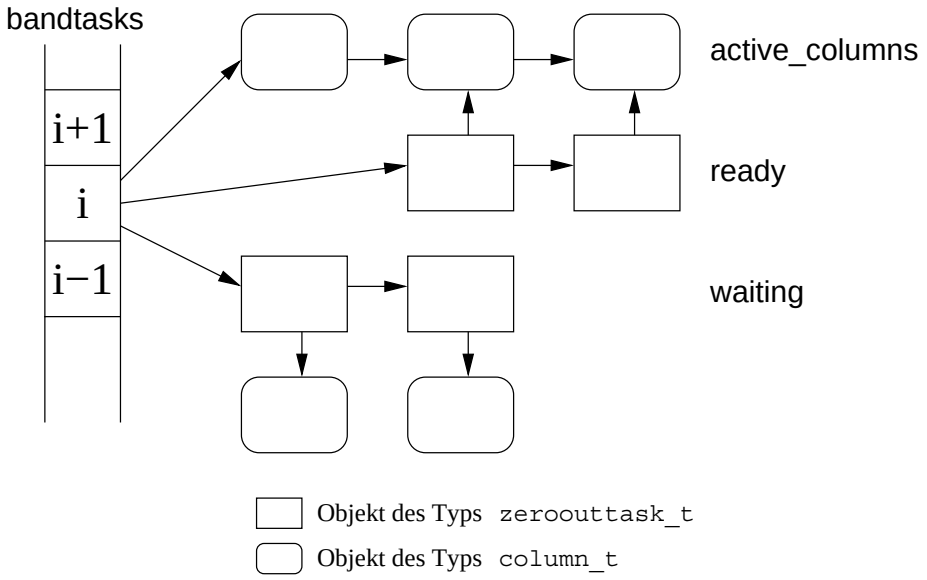



Abbildung 4.9: Bandverwaltung

Es wird, wie in Abb. 4.9 bereits angedeutet, ein Array bestehend aus dem so definierten Datentyp angelegt. Die Dimension dieses Arrays ist die Anzahl der zu eliminierenden Bänder.

### 4.2.3 Abhängigkeiten der Tasks untereinander

Wir unterscheiden zwei Arten der Abhängigkeiten: zum einen die grundsätzlichen Abhängigkeiten, die die richtige Reihenfolge bei der Ausführung der Tasks garantieren und zum anderen die Abhängigkeiten, die die Kollisionsfreiheit bei gleichzeitiger Ausführung mehrerer Tasks sicherstellen.

#### 4.2.3.1 Grundsätzliche Abhängigkeiten

Um bei einer parallelen Ausführung das gleiche Ergebnis wie bei der seriellen Ausführung des Algorithmus zu gewährleisten, sind einige triviale Bedingungen erforderlich:

- Bei der Elimination einer Spalte in einem bestimmten Band kann die Ausführung einer Welle erst dann begonnen werden, wenn die vorhergehende Welle beendet ist.

$$(\mathsf{b},\mathsf{s},\mathsf{w}) \rightarrow (\mathsf{b},\mathsf{s},\mathsf{w}{+}1) \tag{4.5}$$

- Die Elimination der Spalte in einem bestimmten Band kann mit ihrer ersten Welle (mit Index 0) — das ist also die Elimination der Bandspalte selbst — erst dann beginnen, wenn die erste Welle der Elimination der vorhergehenden Spalte im gleichen Band vollendet wurde.

$$(\mathsf{b},\mathsf{s},0) \rightarrow (\mathsf{b},\mathsf{s}{+}1,0) \tag{4.6}$$

- Die Elimination der ersten Spalte in einem Band kann erst dann begonnen werden, wenn die erste Spalte in darunterliegenden Band eliminiert wurde.

$$(\mathsf{b},1,0) \rightarrow (\mathsf{b}{+}1,1,0) \tag{4.7}$$

#### 4.2.3.2 Kollisionsfreiheit

Die Kollisionsfreiheit muss für alle in Ausführung befindlichen und zur Ausführung bereiten Tasks paarweise gegeben sein. Es genügt zu zeigen, dass dies der Fall ist, wenn alle Paare von Tasks, die entweder im gleichen Band oder in benachbarten Bändern arbeiten, keine Kollision aufweisen. Wir unterscheiden daher die folgenden beiden Bedingungen:

**Gleiches Band** Dies enspricht dem Fall 1 in Abschnitt 3.2.4. Er tritt nur bei zwei aufeinander folgenden Spalteneliminationen auf. Kollisionsfreiheit ist für einen bestimmten Task in diesem Fall genau dann gegeben, wenn der Index der Arbeitsspalte des die vorhergehende Spalte bearbeitenden Tasks größer ist als der Index der letzten Spalte, die vom betrachteten Task verändert wird.
Konkret kann man für die Implementation diesen Sachverhalt für einen bestimmten Task task so formulieren: es sei previnfo dasjenige Element in der Liste active_columns des Bandes task%band, das den Index task%column − 1 besitzt. Kollisionsfreiheit liegt genau dann vor, wenn previnfo%workcol > task%column_info%to gilt bzw. ein solches Element previnfo in der Liste nicht vorkommt.

**Benachbarte Bänder** Hier betrachten wir Fall 2 aus Abschnitt 3.2.4. Ein bestimmter Task kann kollisionsfrei ausgeführt werden, wenn im darunterliegenden Band genügend viel Spalten eliminiert worden sind.
Für die Implementation heißt dies, betrachtet man wiederum einen festgelegten Task task: es sei previnfo das erste Element in der Liste active_columns des Bandes task%band − 1. Kollisionsfreiheit liegt genau dann vor, wenn task%column_info%to < previnfo%index.
Es sei angemerkt, dass diese Bedingung eigentlich zu hart ist, da sich der zu previnfo gehörige Task schon in einer Bulge-Elimination befinden könnte, so dass die kleinste noch veränderliche Spalte in diesem Band previnfo%index + 1 wäre.

Es ist nun noch zu zeigen, dass diese Bedingungen zusammen mit den Bedingungen aus 4.2.3.1 genügen, um die Kollisionsfreiheit für alle Taskpaare zu gewähren.

1. Die Wellen einer Spalte müssen in der richtigen Reihenfolge ausgeführt werden.
   Dies ist trivial durch die Bedingung (4.5) erfüllt.

2. Die Spalten eines Bandes müssen in der richtigen Reihenfolge eliminiert werden. Außerdem dürfen bei der Elimination einer Spalte nur Teile der Matrix verändert werden, die von die kleineren Spalten desselben Bandes bearbeitenden Tasks nicht mehr modifiziert werden.

Die Beachtung von (4.6) stellt sicher, dass die Elimination der Bandspalten in der richtigen Reihenfolge stattfinden. Die richtige Abfolge der Eliminationen der Bulgespalten ist dadurch gesichert, dass die Kollisionsfreiheitsbedingung für Tasks im gleichen Band transitiv ist. Wenn also ein Task kollisionsfrei mit der Elimination der direkt vorhergehenden Spalte ablaufen kann, dann ist automatisch die Kollisionsfreiheit mit allen vorhergehenden Spalten sichergestellt.

3. Bei der Elimination einer Spalte eines Bandes dürfen nur Teile der Matrix verändert werden, die durch Eliminationen von Spalten aus niedrigeren Bändern nicht mehr angetastet werden. Bedingung (4.7) garantiert, dass die Reduktion eines Bandes erst beginnen kann, wenn die Reduktion des darunter liegenden Bandes begonnen hat. Die ebenfalls transitive Bedingung für die Kollisionsfreiheit bei benachbarten Bändern gewährleistet die Kollisionsfreiheit mit allen Eliminationen in den darunter liegenden Bändern.

### 4.2.4 Ablauf

Zu Beginn ist der einzig bereite Task der Task (1,1,0), der die Bandspalte der ersten Spalte des ersten Bandes eliminiert. Er wird in die ready-Warteschlange des ersten Bandes eingefügt.

Alle Threads arbeiten nach dem folgenden Schema:

1. Finde das Band mit dem kleinsten Index, das einen Task in der ready-Warteschlange enthält. Entferne diesen Task task aus der Warteschlange und bearbeite ihn. Falls kein Task bereit ist, warte darauf, bis einer bereit wird.

2. Prüfe, ob die Elimination der aktuellen Spalte beendet ist oder ob noch Bulgeeliminationen stattfinden müssen. In letzterem Fall teste für den Task (task%band,task%column,task%wave+1) die Kollisionsfreiheit und stelle ihn entsprechend in die ready- oder waiting-Warteschlange des Bandes task%band.
   In ersterem Fall prüfe für die wartenden Tasks im gleichen Band und in Band task%band+1, ob für diese jetzt Kollisionsfreiheit gegeben ist und stelle sie ggf. in die jeweilige ready-Warteschlange.

3. Wenn task%wave = 0 ist, d. h. man hat gerade eine Bandspalte eliminiert, dann prüfe zunächst, ob die Elimination des Bandes task%band abgeschlossen ist.

   - Ist dies Fall, dann prüfe für alle wartenden Tasks aus dem Band task%band+1, ob sie kollisionsfrei ausgeführt werden können und bewege sie entsprechend von der waiting- in die ready-Warteschlange. War das Band task%band aber schon das letzte zu eliminierende Band, dann ist die Tridiagonalisierung abgeschlossen.
   - Ist die Elimination des Bandes hingegen noch nicht beendet, dann prüfe für den Task (task%band,task%column+1,0) die Kollisionsfreiheit und stelle ihn entsprechend in die ready- oder waiting-Warteschlange des Bandes task%band.

   Gilt zusätzlich noch task%column = 1, dann prüfe für den Task (task%band+1,1,0) die Kollisionsfreiheit und stelle ihn entsprechend in die ready- oder waiting-Warteschlange des Bandes task%band+1.

4. Falls die Tridiagonalisierung noch nicht abgeschlossen ist, beginne wieder mit Schritt 1.

### 4.2.5 Grenzen der Parallelität

Wie gleich gezeigt werden soll, ist die Tridiagonalisierung durch Bandreduktion, so man sie parallel ausführen will, für Matrizen mit voller Bandbreite nicht rentabel. Dies liegt daran, dass bei der Elimination der ersten Bänder die Bedingungen für die Kollisionsfreiheit für Paare von Tasks niemals erfüllt werden können. Erst wenn die Bandbreite der Matrix klein genug ist, können verschiedene Tasks gleichzeitig ausgeführt werden. Es stellt sich daher die Frage, was für Anforderungen an die Bandbreite gestellt werden müssen.

Betrachten wir eine $n \times n$-Matrix, legen $d$ für die Breite der einzelnen Bänder fest und suchen Kriterien für die maximale Gröse der Bandbreite $b$.

Nun überlegen wir, welche Bedingungen erfüllt sein müssen, damit innerhalb der Reduktion des ersten Bandes zwei Tasks gleichzeitig ausgeführt werden können, d. h. die Bedingung „Gleiches Band“ aus Abschnitt 4.2.3.2 erfüllt ist. Die Elimination der ersten Bandspalte erzeugt einen Bulge in den Spalten $b-d$ bis $b$. Die Elimination der ersten Bulgespalte (also in der Spalte $b-d$) erfoldert die Auslöschung aller Elemente in den Zeilen $2b-d$ bis $2b$, was wiederum einen zweiten Bulge in den Spalten $2b-d$ bis $2b$ erzeugt. Die Elimination der zweiten Bulgespalte (ab Zeile $3b-d$) kann nun parallel zur Elimination der zweiten Bandspalte stattfinden. Dies setzt voraus, dass der zweite erzeugte Bulge tatsächlich innerhalb der Matrix liegt, d. h. es muss gelten

$$\begin{aligned} 3b - d &< n \\ 3b &< n + d \\ b &< \frac{n+d}{3} \end{aligned} \tag{4.8}$$

Für die Bedingung „Benachbarte Bänder“ stellen wir ähnlich die Überlegung an, welche Bedingungen erfüllt sein müssen, damit eine Elimination im Band 2 gleichzeitig mit einer Elimination im ersten Band stattfinden kann. Betrachten wir hierzu die Elimination der ersten Bandspalte im zweiten Band, welche sich über die Zeilen $b-2d$ bis $b-d$ erstreckt. Diese erzeugt einen Bulge in den Spalten $b-2d$ bis $b-d$. Diese Elimination kann somit parallel zur Reduktion des ersten Bandes geschehen, wenn im ersten Band schon mindestens $b-d$ Spalten eliminiert wurden. Die zu entfernenden Bandspalten des ersten Bandes gehen bis zur Spalte $n-b+d$ (exklusive), also muss gelten

$$\begin{aligned} b - d &< n - b + d \\ 2b &< n + 2d \\ b &< \frac{n}{2} + d \end{aligned} \tag{4.9}$$

Es ist also angebracht, anfangs die Bandbreite der Matrix so zu reduzieren (z. B. mittels des Householder-Verfahrens), dass wenigstens eine dieser beiden Bedingungen erfüllt ist, und erst anschließend mit der parallelen Bearbeitung zu beginnen.

### 4.2.6 Anmerkungen

Auf die Synchronisierung der Zugriffe auf die Warteschlange trifft hier ebenfalls das schon in Abschnitt 4.1.4.3 Erwähnte zu.

Dieses Verfahrens könnte man zusätzlich noch dahingehend modifizieren, dass man in einem Eliminationsschritt nicht eine einzelne Spalte sondern mehrere Spalten auf einmal entfernt. Dies ergibt

dann eine geblockte Variante des Algorithmus und wird so von Bischof, Sun und Lang [BSL94] vorgeschlagen.

# Anhang A

# Bracket-Notation

Die Bracket-Notation ist eine in der Quantenmechanik verwendete abgekürzte Schreibweise für Funktionen, komplex konjugierte Funktionen und Skalarprodukte bzw. Integrale.

Man setzt für eine Funktion $a(x)$

$$a(x) \equiv |a\rangle$$

und

$$a^*(x) \equiv \langle a|$$

Einen Ausdruck $\langle a|$ bezeichnet man auch als „Bra“ und einen Ausdruck $|a\rangle$ auch als „Ket“.

Für das Skalarprodukt zweier Funktionen $a(x)$ und $b(x)$ schreibt man dann

$$\int a^*(x)b(x)dx = \langle a|b\rangle$$

Hier folgt auf ein „Bra“ ein „Ket“, man erhält ein „Bra-Ket“ oder auch „Bracket“, daher die Bezeichnung der Notation. Solange Bras und Kets einzeln vorkommen, entsprechen diese nur den oben angegebenen Abkürzungen. Wird ein Bracket gebildet, dann impliziert dies automatisch die Integration.

Für eine normierte Funktion $a(x)$ gilt dann

$$\int a^*(x)a(x)dx = \langle a|a\rangle = 1$$

Mit einem Operator $\mathcal{O}$ kann man für

$$\mathcal{O}a(x) = b(x)$$

auch

$$\mathcal{O}\,|a\rangle = |b\rangle$$

schreiben.

Weiterhin gelte die Notation

$$\int a^*(x)\mathcal{O}b(x)dx = \langle a|\mathcal{O}|b\rangle$$

# Abbildungsverzeichnis

# Literaturverzeichnis


[BMS93] Christian Bischof, Mercedes Marques und Xiaobai Sun. Parallel bandreduction and tridiagonalization. In R. Sincovec, Hrsg., *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing*, S. 383–390. SIAM, 1993.

[BNS78] J. R. Bunch, C. P. Nielsen und D. C. Sorensen. Rank-one modification of the symmetric eigenproblem. *Numer. Math.*, 31: S. 31–48, 1978.

[BS92] Christian H. Bischof und Xiaobai Sun. A framework for symmetric band reduction and tridiagonalization. Technical Report MCS–P298–0392, Mathematics and Computer Science Division, Argonne National Laboratory, 1992.

[BSL94] Christian Bischof, Xiaobai Sun und Bruno Lang. Parallel tridiagonalization through two-step band reduction. In *Proceedings of Scalable High Performance Computing Conference*, S. 23–27. IEEE Computer Society Press, Mai 1994.

[Cla85] Tim Clark. *A Handbook of Computational Chemistry*. John Wiley & Sons, Inc., 1985.

[Cup81] J. J. M. Cuppen. A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numer. Math.*, 36: S. 177–195, 1981.

[DS87] J. J. Dongarra und D. C. Sorensen. A fully parallel algorithm for the symmetric eigenvalue problem. *SIAM J. Sci. Stat. Comput.*, 8(2): S. 139–154, März 1987.

[GE94] Ming Gu und Stanley C. Eisenstat. A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem. *SIAM J. Matrix Anal. Appl.*, 15(4): S. 1266–1276, Oktober 1994.

[GE95] Ming Gu und Stanley C. Eisenstat. A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem. *SIAM J. Matrix Anal. Appl.*, 16(1): S. 172–191, Januar 1995.

[Gol73] G. H. Golub. Some modified matrix eigenvalue problems. *SIAM Rev.*, 15: S. 318–334, 1973.

[Lan91] Bruno Lang. *Parallele Reduktion symmetrischer Bandmatrizen auf Tridiagonalgestalt*. PhD thesis, Universität Karlsruhe (TH), 1991.

[Lea96] Andrew R. Leach. *Molecular Modelling: Principles and Applications*. Addison Wesley Longman Limited, 1996.

[Li93] Ren-Cang Li. Solving secular equations stably and efficiently. Technical Report UCB//CSD-94-851, Department of Mathematics, University of California, Berkeley, April 1993. auch: LAPACK Working Note 89.

[Rei94] Joachim Reinhold. *Quantentheorie der Moleküle*. Teubner, Stuttgart, 1994.

[Rut94] Jeffery D. Rutter. A serial implementation of cuppen's divide and conquer algorithm for the symmetric eigenvalue problem. Technical report, Department of Computer Science, University of Tennessee, Knoxville, 1994.

[SB90] Josef Stoer and Roland Bulirsch. *Numerische Mathematk 2*. Springer Verlag, dritte Auflage, 1990.

[Sch02] Franz Schwabl. *Quantenmechanik*. Springer Verlag, sechste Auflage, 2002.

[SCP82] J. J. P. Stewart, P. Császár und P. Pulay. Fast semiempirical calculations. *Journal of Computational Chemistry*, Vol. 3(2): S. 227–228, 1982.

[SO82] Attila Szabo und Neil S. Ostlund. *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory*. Macmillan Publishing Co., Inc., 1982.

[SS90] Gilbert W. Stewart und Ji-guang Sun. *Matrix Perturbation Theory*. Academic Press, Inc., 1990.

[Tho76] R. C. Thompson. The behavior of eigenvalues and singular values under perturbations of restricted rank. *Linear Algebra and Appl.*, 13: S. 69–78, 1976.

[Wil65] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford, 1965.