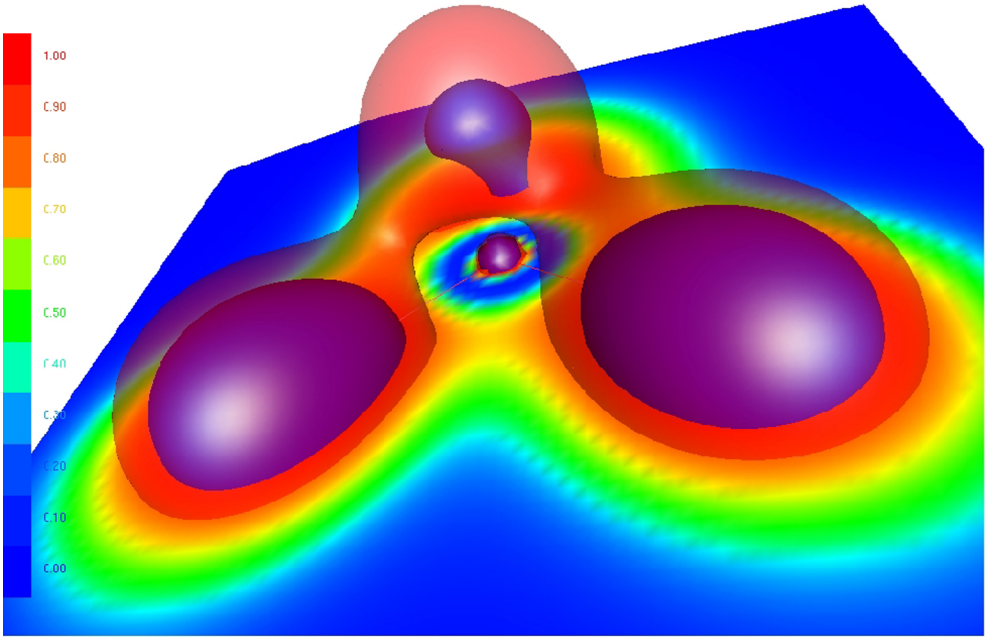
Institut für
Theoretische Chemie
Heinrich-Heine-Universität Düsseldorf

Diplomarbeit
über

Quantitative Charakterisierung von Bindungspolaritäten mittels der Elektronenlokalisationsfunktion und der Theorie von Atomen in Molekülen



vorgelegt von Stephan Raub
aus Düsseldorf

Düsseldorf, den 06. November 2001

Ich versichere hiermit, die vorliegende Arbeit selbstständig und unter ausschließlicher Zuhilfename der im Literaturverzeichnis angegebenen Hilfsmittel erstellt zu haben.

Düsseldorf, den 06. November 2001

Stephan Raub

Danksagung

Ich möchte diese Stelle nutzen um meinen Dank all denen auszusprechen, die in irgendeiner Form dazu beigetragen haben, dass diese Arbeit entstehen konnte.

An erster Stelle möchte ich PD Dr. Georg Jansen für die gründliche und intensive Betreuung, sowohl während des Spezialisierungsstudiums als auch während der Diplomarbeit, danken. Allein seine Anwesenheit trieb mich so manches mal zu Höchstleistungen an.

Dr. Volker Schmid half mir mit den Tücken, die LaTeX für einen bereithält. Und auch das eine oder andere LinuX-Problem hätte ich ohne ihn nicht so schnell gelöst bekommen.

Rolf Linder danke ich für das Lektorat, sowie die Beratung in stilistischen Fragen.

Darüber hinaus gilt mein Dank auch meiner Familie, die mir dieses Studium überhaupt erst ermöglichte. Und natürlich danke ich meiner Verlobten Sandra, die so viele Stunden auf mich verzichten musste, für ihr Verständnis und ihre Geduld. Auch dafür, dass sie sich trotz Krankheit die Zeit nahm, ein letztes Mal nach Tippfehlern zu suchen.

Inhaltsverzeichnis

Danksagung **i**

Inhaltsverzeichnis **ii**

Liste der Tabellen **iii**

Liste der Abbildungen **iv**

Abkürzungen **v**

Symbole **vi**

1 *AIM* und *ELF* - Die Theorie **1**
1.1 Atome in Molekülen (*AIM*) 1
1.1.1 Was ist *AIM* 1
1.1.2 Die Elektronendichte 2
1.1.3 Kritische Punkte in $\rho(\mathbf{r})$ 3
1.1.4 Bader-Atome und offene Systeme 4
1.2 Die Elektronenlokalisationsfunktion (*ELF*) 7
1.2.1 Topologie der *ELF* 7
1.2.2 *AIM/ELF*-Subbassins 10
1.3 Populationsanalyse 11

2 Algorithmen **13**
2.1 Anforderungen an die Wellenfunktion 13
2.2 Finden von Sattelpunkten 14
2.2.1 Sattelpunkte der Dichte 14
2.2.2 Sattelpunkte der *ELF* 14
2.3 Grenzflächen 17
2.3.1 Konstruktion einfacher Grenzflächen 17
2.3.2 Konstruktion komplexer Grenzflächen 18
2.3.3 Analytische Grenzflächen 22

2.3.4 Gefittete Corebassins 27
2.4 Subraumintegration 28
2.4.1 Sphärische Integration von Subräumen 28
2.4.2 **Lebedev**-Integration 30
2.4.3 Surface-Intersection-Algorithmus 31
2.4.4 CoreIntersection 38
2.4.5 Auswertung und Zusammenfassung 38

3 Anwendungsbeispiele **40**
3.1 Technische Details der Rechnungen 40
3.2 Populationsanalyse - Die Roh-Ergebnisse 41
3.2.1 Das Wasser 42
3.2.2 Ammoniak 43
3.2.3 Methan 44
3.2.4 Schwefelwasserstoff 45
3.2.5 Phosphan 46
3.2.6 Silan 47
3.3 Topologische Besonderheiten 48
3.4 Interpretationen der Populationsanalyse 48
3.4.1 Die Rache der Numerik 48
3.4.2 Gegenüberstellungen 49
3.4.3 Durchdringung von Grenzflächen 51

A Herleitung von ELF **52**

B Herleitung des CoreIntersection-Algorithmus **55**

C XElf2 Language Reference **57**
C.1 Parameter Setting 57
C.1.1 Fitting Settings 57
C.1.2 Integration Settings 58
C.1.3 Global Settings 60
C.1.4 Graphic Settings 60
C.1.5 Newton-Raphson Settings 61
C.1.6 Surface Settings 61
C.2 Inits 62
C.3 Critical Points 63
C.3.1 Bader-CPs 63
C.3.2 *ELF*-CPs 63
C.4 Fittings 65
C.4.1 Core-Fits 65

C.4.2 Surface-Fits 65
C.5 I/O Commands 66
C.5.1 Einlesen von Daten 66
C.5.2 Grenzflächen 66
C.5.3 Graphic-Output 67
C.6 Graphics 69
C.7 Integration 70

Literaturverzeichnis **72**

Tabellenverzeichnis

1.1 Stabile kritische BADER-Punkte 4

3.1 Optimierte Geometrien der berechneten Moleküle. 40
3.2 Ergebnisse der Populationsanalyse von H_2O 42
3.3 Ergebnisse der Populationsanalyse von NH_3 43
3.4 Ergebnisse der Populationsanalyse von CH_4 44
3.5 Ergebnisse der Populationsanalyse von H_2S 45
3.6 Ergebnisse der Populationsanalyse von PH_3 46
3.7 Ergebnisse der Populationsanalyse von SiH_4 47
3.8 Elektronegativitäten nach ALLRED-ROCHOW 50
3.9 Vergleiche der A – H-Bindungen, sortiert nach aufsteigender Polarisation der Bindung. 50

Abbildungsverzeichnis

1.1 Logarithmierte Elektronendichte $\log(\rho(\mathbf{r}))$ von FCN, Schnitt durch die Molekül-(y,z)-Ebene als 3D-Plot 3
1.2 Konturplot der logarithmierten Elektronendichte $\log(\rho(\mathbf{r}))$ von FCN, Schnitt durch die Molekül-(y,z)-Ebene mit $(3,-1)$-Bindungskritischen Punkten 4
1.3 Konturplot der logarithmierten Elektronendichte $\log(\rho(\mathbf{r}))$ von FCN, Schnitt durch die Molekül-(y,z)-Ebene, mit *IASs* 5
1.4 *ELF*-Isosurfaces (blau: $\eta(\mathbf{r}) = 0.9$, rot:$\eta(\mathbf{r}) = 0.8$) von FCN und die Benennung der Lokalisationsbassins. Das Corebassin $C(F)$ wird von dem Flourlonepair V(F) umschlossen. 8
1.5 Eindimensionaler Schnitt durch durch die Molekülachse von FCN. (a) *ELF*: $\eta(\mathbf{r})$, (b) *ELF*-Core $\chi^2_\sigma(\mathbf{r})$. Kritische Punkte bleiben in (a) und (b) an der selben Stelle. 9
1.6 Konturplot der *ELF* von FCN mit Überlagerung der Grenzflächen zwischen atomaren Bassins (rot) und Lokalisationsbassins (grün). 10

2.1 Schematische Darstellung einer konstruierten atomaren Grenzfläche. 19
2.2 2D-Projektion einer Lokalisationsgrenzfläche von H_2O (oben) mit äquidistanten α_i (unten). 20
2.3 2D-Projektion einer Lokalisationsgrenzfläche von H_2O (oben) mit adaptiven α_i (unten). 21
2.4 Schema der Fitstruktur. 23
2.5 Einige Chebyshev-Polynome 24
2.6 Zusammenhang zwischen Pseudowinkel $\hat{\theta}$ Realwinkel θ. 26
2.7 Schema der sphärischen Integration.. 29
2.8 Ausschnitt aus einer Lebedev-Tabelle der Ordnung 9 mit $N = 590$. 30
2.9 Projektion des Lebedev-Einheitsvektors in die Eigenebene der Grenzfläche, um einen Startwert für θ zu berechnen. . . . 33

2.10 Schematische Darstellung der vier möglichen Fälle während der Minimierung von $D_{\vartheta_i,\varphi_i}$. 35

3.1 *Oben: ELF*-Isoflächen von H_20 ($\eta = 0.95 \ldots 0.8$ von links nach rechts). *Unten:* Lage der *ECPs*. 42
3.2 *Oben: ELF*-Isoflächen von NH_3 ($\eta = 0.97 \ldots 0.73$ von links nach rechts). *Unten:* Lage der *ECPs*. 43
3.3 *Oben: ELF*-Isoflächen von CH_4 ($\eta = 0.9 \ldots 0.4$ von links nach rechts). *Unten:* Lage der *ECPs*. 44
3.4 *Oben: ELF*-Isoflächen von H_2S ($\eta = 0.91 \ldots 0.8$ von links nach rechts). *Unten:* Lage der *ECPs*. 45
3.5 *Oben: ELF*-Isoflächen von PH_3 ($\eta = 0.93 \ldots 0.75$ von links nach rechts). *Unten:* Lage der *ECPs*. 46
3.6 *Oben: ELF*-Isoflächen von SiH_4 ($\eta = 0.9 \ldots 0.42$ von links nach rechts). *Unten:* Lage der *ECPs*. 47

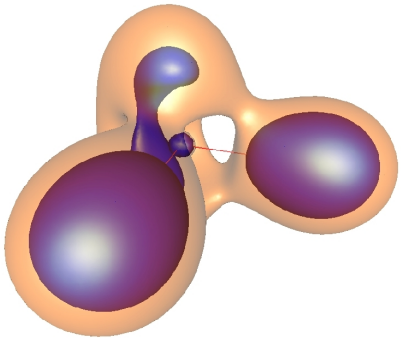
B.1 Schema zum CoreIntersection-Algorithmus 56

Abkürzungen

Å	:	Ångstrøm
Abb.	:	Abbildung
AIM	:	Atoms in Molecules - Theorie nach Bader [1]
BCP	:	bindungskritischer Punkt (*engl. bond critical point*)
bzgl.	:	bezüglich
bzw.	:	beziehungsweise
cm	:	Zentimeter
CP	:	Kritischer Punkt (*engl. Critical Point*)
ELF	:	Elektronenlokalisationsfunktion nach Becke und Edgecombe [2]
et. al.	:	und andere
evtl.	:	eventuell
h	:	Stunde (*engl. hour*)
IAS	:	Interatomare Grenzfläche (*engl. Interatomic Separatrix*)
min	:	Minute
MO	:	Molekül-Orbital
nm	:	Nanometer
QM	:	Quantenmechanik / quantenmechanisch
RCP	:	Ringkritischer Punkt (*engl. ring critical point*)
sec	:	Sekunde
z.B.	:	zum Beispiel

Symbole

$\mathbf{n}_{ij}^{\eta}$	:	Normalenvektor senkrecht zur Grenzfläche S_{ij}^{η}
$\mathbf{n}_{ab}^{\rho}$	:	Normalenvektor senkrecht zur Grenzfläche S_{ab}^{ρ}
$C(X)$	:	Corebassin des Atoms X
R^3	:	Der reale dreidimensionale Raum
S_{ij}^{η}	:	Eine Grenzfläche zwischen den Lokalisationsbassins der Attraktoren i und j in $\eta(\mathbf{r})$
S_{ab}^{ρ}	:	Eine interatomare Grenzfläche zwischen den atomaren Bassins der Attraktoren a und b in $\rho(\mathbf{r})$
$V(X)$	:	monosynaptisches Lonepairbassin des Atoms X
$V(X,Y)$	:	disynaptisches Valenzbassin der Atome X und Y
$V(X,Y)\|Z$	:	disynaptisches Valenzbassin der Atome X und Y in der atomaren Domäne von Z
$\eta(\mathbf{r})$	:	Elektronenlokalisationsfunktion an der Stelle $\mathbf{r}$ nach Becke und Edgecombe [2]
∇	:	Gradientenvektor $\nabla = \frac{\partial}{\partial x} + \frac{\partial}{\partial y} + \frac{\partial}{\partial z}$
ω	:	Rang eines kritischen Punktes
$\rho(\mathbf{r})$	:	Die Einelektronendichte an der Stelle $\mathbf{r}$.
σ	:	Signatur eines kritischen Punktes
Ω	:	Ein Subraum bzw. Subsystem
Ω_i^{η}	:	Lokalisationsbassin des Attraktors i in $\eta(\mathbf{r})$
Ω_a^{ρ}	:	atomares Bassin des Attraktors a in $\rho(\mathbf{r})$



Kapitel 1

AIM und *ELF* - Die Theorie

1.1 Atome in Molekülen (*AIM*)

Die Theorie der *Atome in Molekülen* ist das geistige Kind von Richard F.W. Bader vom Department of Chemistry an der McMaster University in Ontario, Canada. Sie wird in aller Ausführlichkeit in Bader's Monographie [1] erläutert. Im Rahmen dieser Arbeit werden nur die wichtigsten theoretischen Grundlagen, die für das Verständnis der folgenden Kapitel notwendig sind, dargestellt.

1.1.1 Was ist *AIM*

Ein Großteil heutiger *ab initio* QM-Rechnungen gilt der (immer besser werdenden) näherungsweisen Lösung der Schrödinger-Gleichung. Obwohl man damit gute Ergebnisse für Energien und Wellenfunktionen erhält, fragt sich der praktische Chemiker, ob und wie man darin den breiten chemischen Erfahrungsschatz wiederfindet. Im Verlaufe der Entwicklungen in der Chemie erkannte man, dass sich die Eigenschaften vieler Moleküle in erster Näherung aus charakterisitischen atomaren Eigenschaften sowie Bindungseigenschaften zusammensetzen lassen – wobei die angesprochenen atomaren Eigenschaften allerdings in zweiter Näherung wiederum von der molekularen Umgebung abhängen.

AIM soll die Lücke zwischen anerkannten und nützlichen Begriffen wie Atome und Bindungen auf der einen und den Lösungen der Schrödinger-Gleichung auf der anderen Seite schließen [3]. Es ist somit eine *interpretative* Theorie um chemische Einsichten und Konzepte aus den Ergebnissen moderner *ab initio*-Rechnungen zu gewinnen [4].

BADER stellt zunächst die Frage: Gibt es überhaupt so etwas wie Atome in Molekülen?

Wenn isolierte Atome sich zu einem Molekül verbinden, so gehen die meisten Chemiker implizit davon aus, dass diese individuellen Atome mehr oder weniger als solche existent bleiben und durch Wechselwirkungen untereinander einen Verbund bilden – das Molekül. Aber ist das wirklich so? Genauso gut könnte es sein, dass die entsprechenden Atome zu einem homogenen Objekt namens Molekül zusammenfließen und damit vollständig ihre individuelle Identität verlieren. Die Wellenfunktion eines Molekül „weiss“ *a priori* nichts von Untereinheiten, also Atomen.

AIM benutzt die Einelektronendichte um das Molekül durch mathematische Methoden in mehrere Subräume zu zerlegen. BADER definiert damit zwei wichtige Eckpunkte der anschaulichen Chemie: Atome und Bindungen. Durch BADER's *AIM* ist es möglich aus der Topologie der Elektronendichte Informationen über Ort und Ausdehnung dieser Atome zu gewinnen. Letztendlich ist somit auch ein Populationsanalyse dieser atomaren Domänen möglich.

1.1.2 Die Elektronendichte

Die Elektronendichte $\rho(\mathbf{r})$ ist der Ansatzpunkt von sowohl *AIM* als auch *ELF*. Man erhält sie z.B. aus QM-Rechnungen durch Integration der Wellenfunktion über alle Elektronen außer einem. Die Elektronendichte ist aber auch eine meßbare Größe – man erhält sie direkt aus Röntgenstrukturmessungen. $\rho(\mathbf{r})$ existiert also im realen dreidimensionalen Raum und nicht, wie die Wellenfunktion, nur in einem hochdimensionalen HILBERT-Raum.

Die Topologie von $\rho(\mathbf{r})$ ist Ausdruck der in dem System herrschenden Kräfte. Typischerweise wird die Topologie durch die Atomkerne bestimmt (Abb. 1.1) wodurch der Schluß naheliegt, dass die Eigenschaften eines Systems vollständig von den Eigenschaften der es bildenden Atome dominiert werden. Das deckt sich mit der oben gemachten Aussage, dass man die Atome durch ihre Eigenschaften in den Molekülen bemerkt.

Die massiven Peaks der Kerne scheinen aus einem Meer von Regionen niedriger Elektronendichte hervorzustechen. Für die Deutung von Bindungsverhältnissen sind jedoch gerade die Zonen *zischen* den Kernen von Bedeutung.

Bei der topologischen Analyse der Elektronendichte betrachtet man jedoch zweckmäßigerweise zunächst nicht die Elektronendichte selbst, sondern ihr Gradienten-Vektor-Feld $\nabla\rho(\mathbf{r})$.

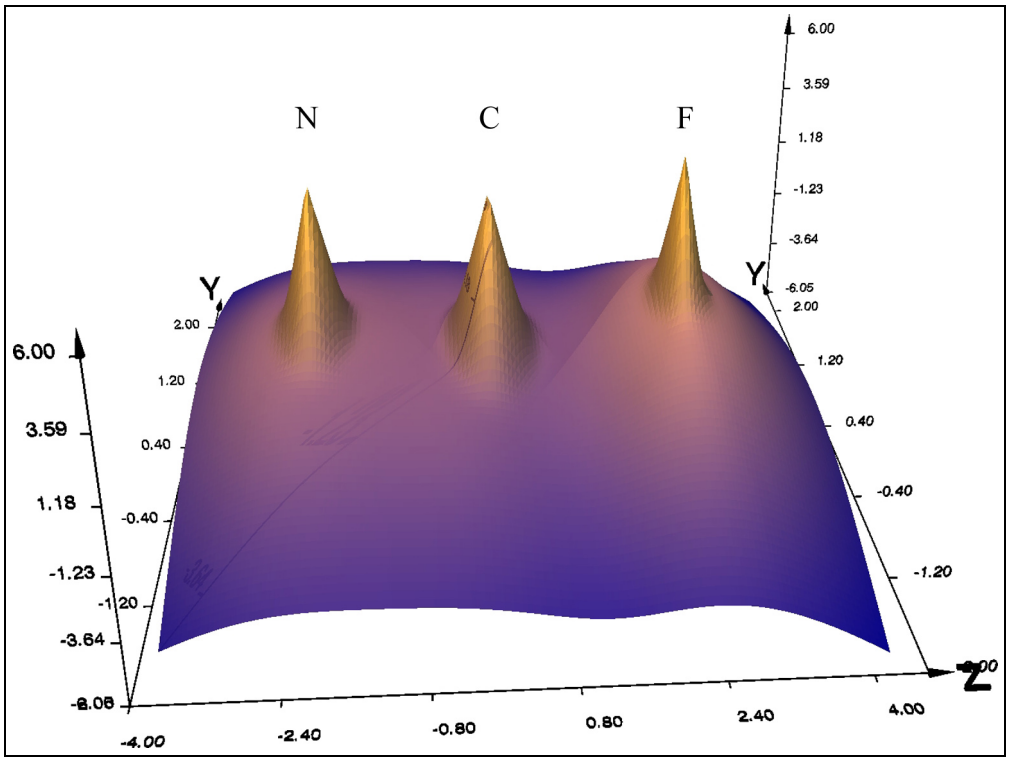


Abbildung 1.1: Logarithmierte Elektronendichte $\log(\rho(\mathbf{r}))$ von FCN, Schnitt durch die Molekül-(y, z)-Ebene als 3D-Plot

1.1.3 Kritische Punkte in $\rho(\mathbf{r})$

Jede topologische Eigenschaft in $\rho(\mathbf{r})$ ist verbunden mit einem *kritischen Punkt* an der Stelle $\mathbf{r}_{cp}$, so dass gilt

$$\nabla\rho(\mathbf{r}_{cp}) = 0 \tag{1.1}$$

Zur Charakterisierung der kritischen Punkte wird an $\mathbf{r}_{cp}$ der Hessian bestimmt und diagonalisiert. Der Rang ω ist dann die Anzahl der Eigenwerte des Hessians ungleich null. Die Signatur σ ist die algebraische Summe der Vorzeichen dieser Eigenwerte.

Von einigen Ausnahmen einmal abgesehen, sind alle kritischen Punkte für Moleküle in der Nähe einer energetisch stabilen Konfiguration vom Rang $\omega = 3$ für beliebige Moleküle. In Tabelle 1.1 sind die möglichen kritischen Punkte mit Rang $\omega = 3$ aufgeführt.

Da die Kerne echte Maxima in $\rho(\mathbf{r})$ darstellen, muss sich auf der Verbindungslinien zwischen jeweils zwei benachbarten Maxima ein Sattelpunkt befinden (Abb. 1.2). BADER identifiziert diejenige Verbindungslinie zwischen

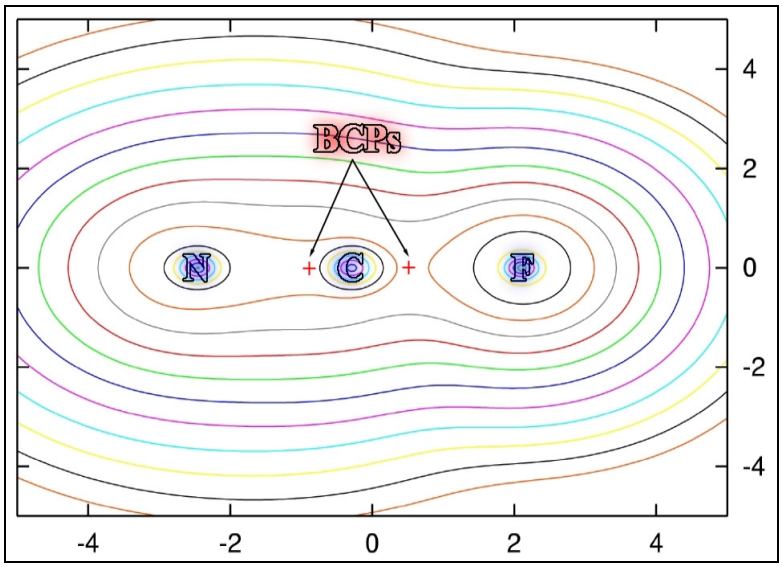


Abbildung 1.2: Konturplot der logarithmierten Elektronendichte $\log(\rho(\mathbf{r}))$ von FCN, Schnitt durch die Molekül-(y,z)-Ebene mit $(3,-1)$-Bindungskritischen Punkten

zwei Kernen, die durch einen $(3,-1)$-Sattelpunkt läuft als die chemische Bindung zwischen diesen Atomrümpfen. Folglich nennt man sie Bindungskritische Punkte (*BCPs*). In Ring- oder Käfigsystemen sind auch $(3,+1)$- bzw. $(3,+3)$-Sattelpunkte möglich, weshalb man hier von Ring- bzw. Käfigkritischen Punkten (*RCPs* bzw. *CCPs*) spricht.

1.1.4 Bader-Atome und offene Systeme

Ein BADER-Atom ist die Einheit aus Attraktor im Gradienten-Vektorfeld $\nabla\rho(\mathbf{r})$ und dessen Bassin Ω^{ρ}, also ein Subsystem des Moleküls und ein Sub-

Tabelle 1.1: Stabile kritische BADER-Punkte

Signatur	Beschreibung
$(3,-3)$	Maximum, meist ein Atomkern
$(3,-1)$	Sattelpunkt, Bindungskritischer Punkt (*BCP*), meist auf den Kernverbindungsachsen
$(3,+1)$	Sattelpunkt, Ringkritischer Punkt (*RCP*), innerhalb von Ringsystemen
$(3,+3)$	Minimum, Käfigkritischer Punkt (*CCP*), innerhalb von Käfigsystemen

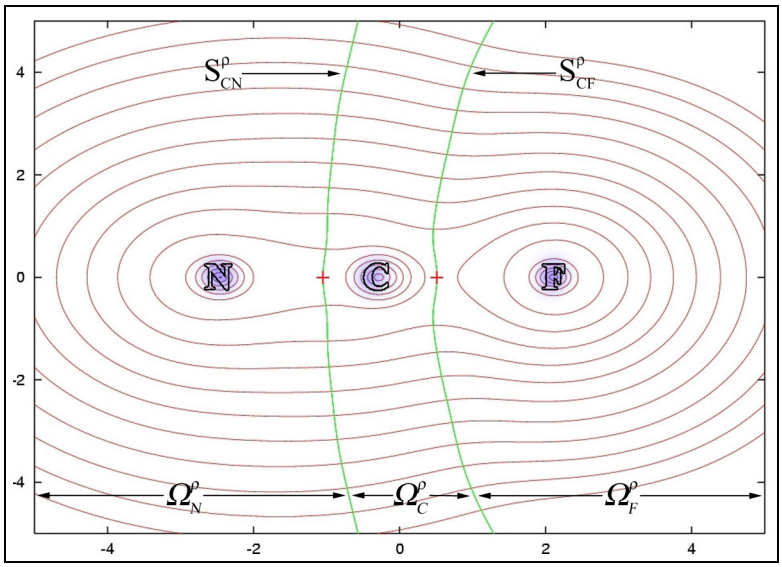


Abbildung 1.3: Konturplot der logarithmierten Elektronendichte $\log(\rho(\mathbf{r}))$ von FCN, Schnitt durch die Molekül-(y, z)-Ebene, mit *IASs*

raum des R^3. Der Attraktor ist meist ein isolierter $(3, -3)$-kritischer Punkt in $\rho(\mathbf{r})$ (exotische Beispiele, bei denen das nicht der Fall ist, finden sich in [5, 1]).

Das Bassin ist die Menge aller Punkte auf allen Gradientenpfaden in $\rho(\mathbf{r})$, die in diesem Attraktor enden. Zwei Bassins Ω_a und Ω_b sind durch *interatomare Seperatrices (IAS)* S^{ρ}_{ab} getrennt (Abb. 1.3). Das sind die Flächen, die niemals von einem Gradientenpfad geschnitten werden:

$$\nabla\rho(\mathbf{r}) \cdot \mathbf{n}^{\rho}_{ab}(\mathbf{r}) = 0 \quad \forall \mathbf{r} \in S^{\rho}_{ab} \tag{1.2}$$

mit $\mathbf{n}^{\rho}_{ab}(\mathbf{r})$ dem Normalenvektor senkrecht zur Ebene S^{ρ}_{ab} an der Stelle $\mathbf{r}$.

Der R^3 wird durch die IAS also in klar von einander abgegrenzte disjunkte Gebiete unterteilt, die man als die Atome in dem Molekül identifiziert. Somit ist jedem Atom ein definierter Raum und ein definierter Anteil an der Elektronendichte zugeteilt.

1.1.4.1 Eigentliche offene Systeme

Es ist die Eigenschaft von offenen Subsystemen, dass ein infinitesimaler Flux der Ladungsdichte durch deren Grenzen stattfindet. Das unterscheidet sie vom Gesamtsystem. Der Flux verschwindet allerdings zu null durch eine Flä-

che, die der Bedingung

$$\sum_{i}^{N} \nabla_i \Psi(r_1, ..., r_N) \cdot \mathbf{n} = 0 \tag{1.3}$$

mit $\Psi(r_1, ..., r_N)$ der Wellenfunktion des Gesamtsystems und dem Normalenvektor der Fläche $\mathbf{n}$, genügt. Dies ist automatisch erfüllt für eine Grenzfläche im Unendlichen, die somit das Gesamtsystem umschliesst. Da hier kein Flux existiert, ist die Energie des Gesamtsystems stationär.

Eine Erweiterung von SCHWINGER's Prinzip der stationären Aktion [6] besagt, dass eine Übertragung quantenmechanischer Gesetzmäßigkeiten vom Gesamtsystem auf Subsysteme nur dann möglich ist, wenn bestimmte Randbedingungen erfüllt sind. Solche offenen Subsysteme, die diesen Randbedingungen genügen, nennt man *eigentliche offene Systeme* [7].

Diese Randbedingungen verlangen, dass der Flux im Grandienten-Vektorfeld der Ladungsdichte an jedem Punkte der Grenzflächen $S^{\rho}(\mathbf{r})$, die das Subsystem einschließt, null wird. Das ist aber gerade jene Bedingung, die für die *IASs* gelten. Somit sind BADER-Atome eigentliche offene Systeme, die zwar frei sind, Ladung mit ihrer Umgebung auszutauschen, die aber im statistische Mittel genau so viele Elektronen empfangen, wie sie abgeben.

Wegen des Prinzips der stationären Aktion sind die Eigenschaften dieser eigentlichen offenen Systeme Ω in genau derselben Weise, definiert wie die des Gesamtsystems, und denselben Gesetzmäßigkeiten gehorchen. Ferner lassen sich die Eigenschaften des Gesamtsystems additiv aus den entsprechenden Eigenschaften der Subsystene zusammensetzen [8].

1.2 Die Elektronenlokalisationsfunktion (*ELF*)

Bei der traditionellen Beschreibungsweise der Chemie sind lokalisierte Gruppen von Elektronen von enormer Wichtigkeit, spricht man doch Core-Schalen, Bindungselektronenpaaren, Lonepairs, etc. [2] Die kanonischen Orbitale der HARTREE-FOCK-Theorie sind dagegen delokalisiert und liefern so keinen Anhaltspunkt für lokalisierte Elektronenpaare. Jedoch genügt eine unitäre Transformation, um aus den kanonischen Orbitalen lokalisierte Orbitale zu generieren, denn die HARTREE-FOCK-Theorie ist komplett durch ihre Einelektronendichtematrix definiert, die invariant bezüglich jeglicher unitärer Transformation ist. Eine sinnvolle Definition für Elektronenlokalisation ist daher bei der Dichtematrix anzusetzen und nicht bei den Orbitalen.

Von außerordentlicher Bedeutung ist hier das PAULI-Prinzip. Es dürfen sich niemals zwei Elektronen gleichen Spins auf den selben Raumkoordinaten aufhalten. Zwar stoßen sich auch Elektronen mit entgegengesetztem Spin ab, dieser Effekt ist aber auf die COULOMB-Abstoßung zurückzuführen, die auch zwischen Elektronen gleichen Spins wirkt, und nicht auf das PAULI-Prinzip. Sie ist gewöhnlich um eine Größenordnung kleiner. [9] So kann ein klassisches LEWIS-Elektronenpaar nur existieren, wenn beide Elektronen unterschiedlichen Spin haben.

1.2.1 Topologie der *ELF*

Die *ELF* erlaubt es solche Raumbereiche zu identifizieren, in denen LEWIS-Elektronenpaare (oder auch einzelne radikalische Elektronen) besonders stark lokalisiert sind.

Die *ELF* ist nach [2] definiert zu

$$\eta(\mathbf{r}) = (1 + \chi^2)^{-1} \tag{1.4}$$

$$\chi = \frac{\frac{1}{2}\sum_{i=1}^{N} |\hat{\nabla}\phi_i(\mathbf{r})|^2 - \frac{1}{8} \cdot \frac{|\hat{\nabla}\rho(\mathbf{r})|^2}{\rho(\mathbf{r})}}{c \cdot \rho(\mathbf{r})^{\frac{5}{3}}} \tag{1.5}$$

Für die Herleitung dieses Ausdrucks siehe *Anhang A* auf Seite 52.

In Analogie zu *AIM* nutzt man das Gradientenvektorfeld $\nabla\eta(\mathbf{r})$ um den R^3 in separierte Lokalisationsbassins Ω_i^η zu unterteilen. Jedes dieser Lokalisationsbassins ist die Einheit aus seinem Punktattraktor (oder bei linearen Molekülen auch ein ringförmiger Attraktor) der *ELF* und allen Punkten auf allen Gradientenpfaden, die in diesem Attraktor enden. Wieder kann man,

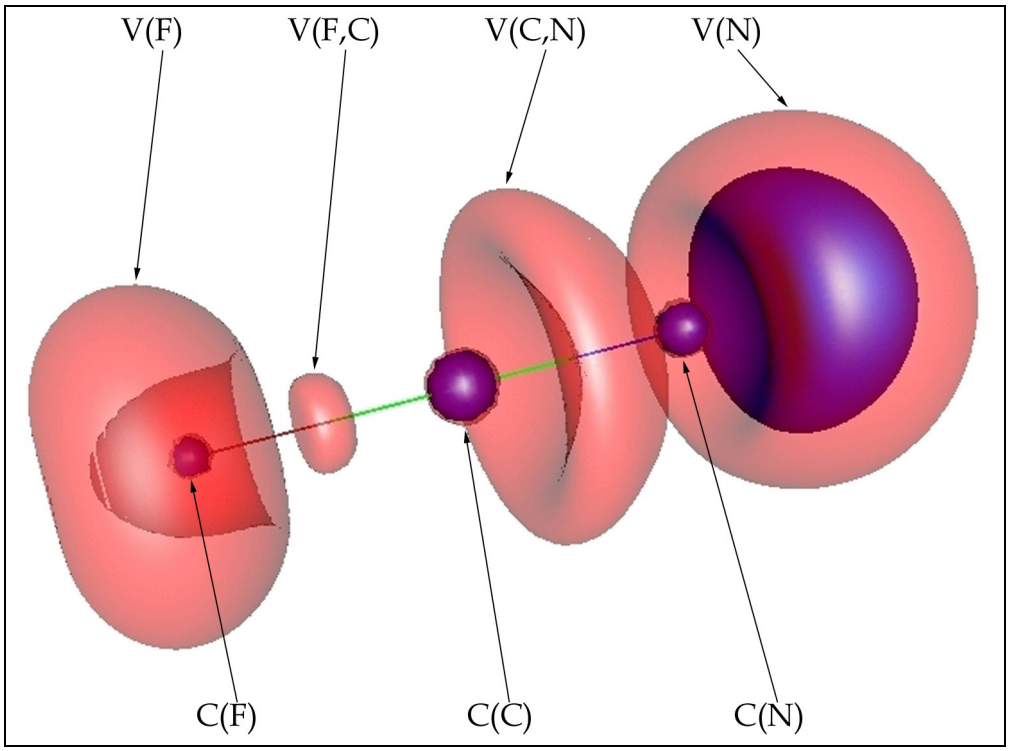


Abbildung 1.4: *ELF*-Isosurfaces (blau: $\eta(\mathbf{r}) = 0.9$, rot:$\eta(\mathbf{r}) = 0.8$) von FCN und die Benennung der Lokalisationsbassins. Das Corebassin $C(F)$ wird von dem Flourlonepair V(F) umschlossen.

ausgehend von einem Sattelpunkt in $\eta(\mathbf{r})$ ein Fläche $S_{ij}^{\eta}(\mathbf{r})$ konstruieren, die niemals von einem Gradientenpfad der *ELF* geschnitten wird:

$$\nabla\eta(\mathbf{r}) \cdot \mathbf{n}_{ij}^{\eta}(\mathbf{r}) = 0, \forall \mathbf{r} \in S_{ij}^{\eta}(\mathbf{r}) \tag{1.6}$$

Diese Fläche teilt zwei benachbarte Lokalisationsbassins Ω_i^{η} und Ω_j^{η} und enthält, genau wie ihr Analogon in *AIM*, einen $(3,-1)$-CP, hier allerdings in $\eta(\mathbf{r})$. Bei linearen Molekülen können statt $(3,-1)$-CPs auch Ringe von $(2,0)$-CPs auftreten.

Die Lokalisationsbassins selbst werden klassifiziert als *Corebassin* des Atoms X $C(X)$ und *Valenzbassin* V [10]. Sind mehrere Corebassins eines Atoms vorhanden, werden sie entsprechend der korrespondierenden Schale (*K,L,M...*) als $C_K(X)$, $C_L(X)$, etc. bezeichnet.

Die Valenzbassins benennt man nach den Atomen, die sie ausbilden. Wird ein Valenzbassin nur von einem Atom X ausgebildet, bezeichnet man es als *monosynaptisch* – sein Symbol ist $V(X)$. Dieses Bassin entspricht weitgehend dem Raumbereich, in dem ein oder mehrere klassische Lonepairs lokalisiert

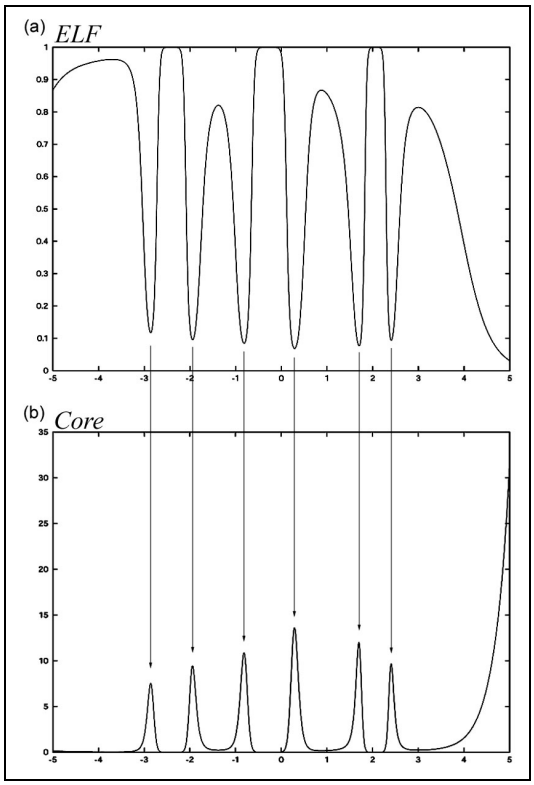


Abbildung 1.5: Eindimensionaler Schnitt durch durch die Molekülachse von FCN. (a) *ELF*: $\eta(\mathbf{r})$, (b) *ELF*-Core $\chi^2_\sigma(\mathbf{r})$. Kritische Punkte bleiben in (a) und (b) an der selben Stelle.

sind. *Disynaptische* Valenzbassins gehören zu zwei Atomen X und Y und tragen das Symbol $V(X,Y)$. Sie wiederum entsprechen den Raumbereichen eines oder mehrerer lokalisierter Bindungselektronenpaare. Es gibt auch höhersynaptische Valenzbassins, die entsprechend benannt werden. Abb. 1.4 zeigt dies am Beispiel des FCN.

Eine topologische Besonderheit sei hier schon einmal erwähnt, obwohl sie erst bei der Besprechung der Algorithmen gebraucht wird: Die kritischen Punkte liegen für $\eta(\mathbf{r})$ (Gleichung A.10, Seite 53) auf den selben Koordinaten wie sie für den quadrierten *ELF*-Core $\chi^2(\mathbf{r})$ (siehe Gleichung A.11, Seite 54 und Abb. 1.5) lägen. Lediglich die Vorzeichen der Hessian-Eigenwerte sind gerade entgegengesetzt. Somit kann unter Verwendung von $-\chi^2(\mathbf{r})$ statt $\eta(\mathbf{r})$ der Rechenaufwand drastisch reduziert werden, ohne Genauigkeit in der Lokalisierung der *ELF*-kritischen Punkte zu verlieren.

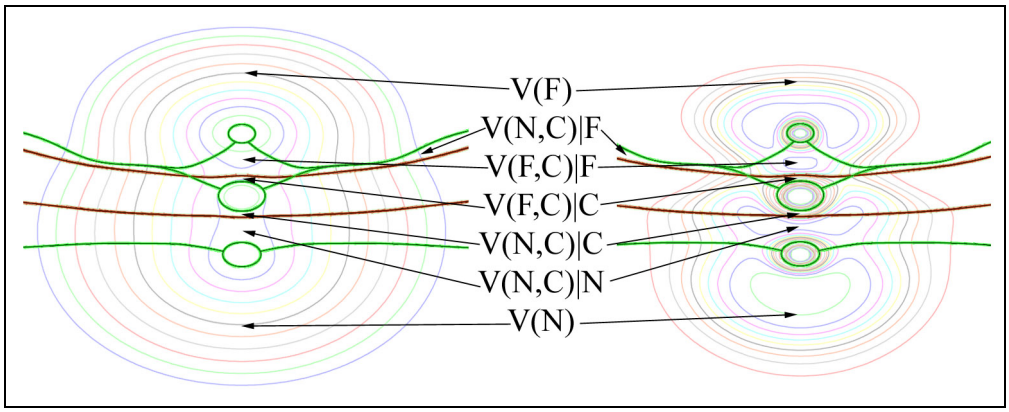


Abbildung 1.6: Konturplot der *ELF* von FCN mit Überlagerung der Grenzflächen zwischen atomaren Bassins (rot) und Lokalisationsbassins (grün).

1.2.2 *AIM/ELF*-Subbassins

Sowohl die atomaren als auch die Lokalisationsbassins stellen eine Partitionierung des dreidimensionalen Raums in untereinander abgegrenzte Regionen dar. Jedoch ist ein Lokalisationsbassin nicht notwendigerweise von einem atomaren Bassin getrennt. Vielmehr kann das Lokalisationsbassin in dem atomaren vollständig enthalten sein, oder die beiden Bassins überlappen sich. Abb. 1.6 zeigt die Überlagerung der atomaren und der Lokalisations-Grenzflächen für FCN.

Das Bassin

$$\Omega_s = \Omega_i^\eta \cap \Omega_a^\rho \tag{1.7}$$

aller Punkte, die sowohl in dem atomaren Bassin Ω_a^ρ als auch in dem Lokalisationsbassin Ω_i^η enthalten sind, nennt man ein *atomares Subbassin des Lokalisationsbassins*. Die Notation ist für monosynaptische Subbassins $V(X)|Z$ und für di- bzw. polysynaptische Subbassins $V(X,Y)|Z$. Gemeint ist hierbei der Teil des Lokalisationsbassins $V(X)$ bzw. $V(X,Y)$, der mit dem atomaren Bassin von Z überlagert wurde.

Anschaulich ist damit der jenige Teil des Lokalisationsbassins gemeint, der in der atomaren Domäne von Z liegt. Das Lokalisationsbassin wird also auf die beteiligten Atome aufgeteilt. Di- und polysynaptische Valenzbassins von Bindungen werden beispielsweise unter den an der Bindung beteiligten Atomen aufgeteilt. Selbst Monosynaptische Lonepairbassins können gelegentlich in „fremde“ atomare Bassins ragen und werden somit auch auf die Atome aufgeteilt. Monosynaptische Corebassins hingegen liegen nach aller Erfahrung ungeteilt vor.

1.3 Populationsanalyse

Die Integration von $\rho(\mathbf{r})$ innerhalb eines Bassin liefert die mittlere Elektronenpopulation dieses Bassins. Allgemein ist sie gegeben als

$$\bar{N}_s = \bar{N}(\Omega_s) = \int_{\Omega_s} \rho(\mathbf{r})\, d\tau. \tag{1.8}$$

Dies ist die zentrale Größe dieser Arbeit. Obwohl dieser Ausdruck recht harmlos aussieht, ist jedoch die Integration über einen Subraum, der von beliebig vielen verschiedenen Grenzflächen umschlossen wird, alles andere als trivial. Das gesamte Kapitel *Algorithmen* beschreibt nichts weiter als die Schritte die notwendig sind, um den Ausdruck (1.8) numerisch lösen zu können.

Obwohl der Ausdruck also recht kompliziert auszurechnen ist, stellt er eine simple, gut handhabbare, aus der Wellenfunktion gewonnene, Größe dar, die die Chemie einer Verbindung beschreibt.

Die Integration innerhalb eines *atomaren Bassins* liefert die mittlere Anzahl der Elektronen, die diesem Atom im molekularen Verbund angehören. Verglichen mit der Anzahl der Elektronen des ungebundenen Atoms ergibt sich die formale Ladung als Differenz der beiden Größen. Diese Methode ist schon lange bekannt und etabliert [11].

Die Integration innerhalb eines *Lokalisationsbassins* liefert die mittlere Anzahl der Elektronenen, die in der entsprechenden funktionalen Domäne lokalisiert sind. Dies kann z.B. die Anzahl der Elektronen in einer monosynaptischen *Core*- oder *LonePair*-Domäne sein. Vor allem aber kann so die mittlere Anzahl der Elektronen bestimmt werden, die in einer chemischen Bindung lokalisiert ist. Dies liefert den mittleren Bindungsgrad – eine sehr klassiche chemische Information, gewonnen aus der QM-Wellenfunktion. Für die Bindung zweier Atome X und Y, die das disynaptische Valenzbassin $V(X,Y)$ ausbilden, ergibt sich der Bindungsgrad b_{XY} als

$$b_{XY} = \frac{\bar{N}\left(V(X,Y)\right)}{2} \tag{1.9}$$

Diese Formel wurde bereits in [12] vorgeschlagen.

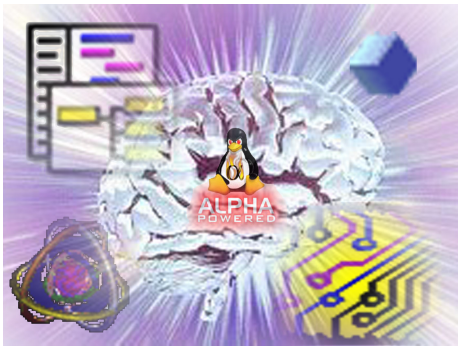
Die Überlagerung von atomaren mit Lokalisationsgrenzflächen und die Integration innerhalb dieser Subräume wurde erstmals von Jansen et. al. vorgeschlagen [13]. Eine erste Version der im Rahmen dieser Arbeit entstandenen Software (XELF 1.3) konnte erstmals die Integration innerhalb dieser Subräume bei linearen Molekülen durchführen [14]. Man erhält so quantitative Aussagen darüber, zu welchen Anteilen die Elektronen innerhalb eines polysynaptischen Lokalisationsbassin auf die es ausbildenen Atome verteilt

ist. Anschaulich kann man sagen, man erfährt wieviel Elektronen der Bindung den jeweiligen Atomen gehören. Diese Information erlaubt Aussagen über die *Polarität* der Bindung. Sie wird für zwei Atome X und Y, die eine disynaptische Valenzdomäne $V(X,Y)$ ausbilden, nach [14] quantifiziert durch den *Bindungspolaritätsindex*

$$p_{XY} = \frac{\bar{N}\left(V(X,Y)|X\right) - \bar{N}\left(V(X,Y)|Y\right)}{\bar{N}\left(V(X,Y)|X\right) + \bar{N}\left(V(X,Y)|Y\right)}. \tag{1.10}$$

Dieser Index reicht von $p_{XY} = 0$ für homopolare Bindungen bis zu $p_{XY} = 1$ für idealisierte ionische Bindungen.

Die Größen b_{XY} und p_{XY} zusammen ergeben eine gute quantitative Beschreibung der entsprechenden chemischen Bindungsverhältnisse, die zwar direkt aus der Wellenfunktion gewonnen wird, sich jedoch in der Sprache der praxisorientierten Chemie ausdrücken lässt.



Kapitel 2

Algorithmen

Hauptthema dieser Arbeit sind die Algorithmen, die eine Populationsanalyse überhaupt erst möglich machen. Ausgehend von einer von einem kommerziellen Programm erzeugten Wellenfunktion sind mehrere Schritte nötig, die jeweils mehrere verschiedene Algorithmen erfordern.

Sämtliche, hier beschriebenen Algorithmen sind in dem Programm XELF2 implementiert - ein FORTRAN90-Projekt von mehr als 39.000 Zeilen Sourcecode. Das Programm verfügt über eine eigene Scriptsprache (siehe Anfang C, Seite 57) und mehrere Import/Export-Filter. Ein Vorläufer dieses Programms ist XELF1 [14], welches jedoch auf linearsymmetrische Moleküle beschränkt ist.

2.1 Anforderungen an die Wellenfunktion

Ausgangspunkt für jede Populationsanalyse ist eine Wellenfunktion, wie sie aus vielen kommerziellen QM-Programmpaketen exportiert werden kann. Dabei eignen sich solche Formate besonders, die die Möglichkeit bieten, eine Funktion zu implementieren, die für einen Punkt $\mathbf{r}$ die Einelektronendichte liefert – also solche, die sich an analytischen Ausdrücken mit entsprechenden Koeffizientensätzen orientieren. *Grid*- oder *Cube*-Formate, die feste Werte für $\rho(\mathbf{r})$ liefern, sind eher ungeeignet.

Wie man aus Gleichung (1.5) ersehen kann, sind nötig für die Berechnung von $\eta(\mathbf{r})$ die Ausdrücke für die MOs $\phi_i(\mathbf{r})$, die Einelektronendichte $\rho(\mathbf{r})$ sowie die ersten Ableitungen der beiden Größen. Da im weiteren Verlauf der Gradient und der Hessian der *ELF* benötigt wird, werden somit auch die ersten, zweiten und dritten Ableitungen von $\phi_i(\mathbf{r})$ und $\rho(\mathbf{r})$ gebraucht. Deshalb sind analytische Ausdrücke für letztere unverzichtbar.

2.2 Finden von Sattelpunkten

2.2.1 Sattelpunkte der Dichte

Von den zwei Typen von Sattelpunkten, nämlich Sattelpunkte in $\rho(\mathbf{r})$ und Sattelpunkte in $\eta(\mathbf{r})$, sind die in $\rho(\mathbf{r})$ erheblich leichter zu bestimmen. Das liegt zum einem daran, dass die Topologie der Einelektronendichte bei weitem nicht so kompliziert ist, wie die der *ELF*. Zum anderen weiß man schon recht genau, wo die *BCPs* liegen müssen – zwischen jeweils zwei Atomen, die eine Bindung ausgebildet haben.

Diese Erkenntnis ist eine erhebliche Erleichterung für das numerische Bestimmen der Position der *BCPs*, da alle bekannten Verfahren zur Bestimmung von Sattelpunkten einen guten Startwert benötigen, um brauchbare Ergebnisse zu liefern.

So auch das Verfahren, dass in der aktuellen Implementation von XElf2 zur Anwendung kommt: das Newton-Raphson-Verfahren [15]. Jedoch mit der Erkenntnis über den ungefähren Aufenthaltsort der *BCPs*, lässt sich ein schlüssiger Algorithmus aufstellen: Es werden die Punkte in der Mitte von jeweils zwei Kernen berechnet, deren Entfernung im Bereich normaler Bindungslängen liegt, und von dort aus eine Newton-Raphson-Suche gestartet.

Bei einigen Dreifachbindungen kommt es jedoch zu dem Phänomen der *Pseudoatome*: Auf dem Mittelpunkt der Kernverbindungsachse befindet sich kein Sattelpunkt sondern ein Maximum in $\rho(\mathbf{r})$, was ja eigentlich charakteristisch für ein Atom ist. Folglich gibt es zwei echte Sattelpunkte links und rechts von diesem Pseudoatom auf der Kernverbindungsachse. Startet man die Suche nach Sattelpunkten auf dem Pseudoatom, so liegt es am numerischen Rauschen, welchen der beiden Sattelpunkte man findet. Um das zu vermeiden wird zusätzlich bei 1/3 und 2/3 der Kernverbindungsachse eine Newton-Raphson-Suche gestartet.

Alle gefundenen Sattelpunkte werden in eine temporäre Datei geschrieben. Da es vorkommen kann, dass verschiedene Startpunkte den selben Sattelpunkt liefern, werden in einem zweiten Schritt die Einträge in der temporären Datei sortiert, wobei bei mehrfach vorkommenden Punkten derjenige mit dem niedrigsten Gradienten übernommen wird.

2.2.2 Sattelpunkte der *ELF*

Viel schwieriger gestaltet sich die Suche nach den Sattelpunkten der *ELF*, und zwar aus mehreren Gründen. Zum einen ist es nicht so einfach, ihre ungefähre Lage abzuschätzen, wie dies bei *BCPs* der Fall ist. Zum anderen

ist die Topologie der *ELF* deutlich komplexer als die der Elektronendichte, was eine einfache Newton-Raphson-Suche schon Mal überfordern kann.

2.2.2.1 Mit brutaler Gewalt

Eine *brute-force*-Methode ist es, einfach von jedem Punkt eines Gitters, das das Molekül umschließt, eine Sattelpunktsuche zu starten. Das alte, zweidimensionale XElf1 tat dies auch noch, aber durch die lineare Symmetrie der Moleküle beschränkte sich die Suche auf ein 2d-Grid. Das neue XElf2 sieht diesen Modus auch noch für dreidimensionale Moleküle vor, aber eher als letzten Ausweg, wenn alle im Weiteren vorgestellten Methoden zu keinem Erfolg führen (was recht selten, aber nicht völlig unmöglich ist).

Die Erfahrung zeigt allerdings, dass die *ECPs* meist einen *ELF*-Wert nicht kleiner als 0.5 besitzen. XElf2 besitzt deshalb die Methode `FindCPRange`, mit der in einer Box nach *ECPs* gesucht wird, die alle *ELF*-Werte größer als einen Referenzwert umgibt (siehe Abschnitt C.3.2, Seite 63). So wird der Suchbereich auf ein meist erträgliches Maß eingegrenzt.

Auch hier ist es natürlich möglich, dass Sattelpunkte mehrmals von verschiedenen Startpunkten aus gefunden werden. Deshalb ist auch hier eine nachträgliche Sortierung erforderlich.

2.2.2.2 OctaedricGrid-Algorithmus

Eine elegante und schnelle Methode, geeignete Startpunkte für eine Newton-Raphson-Suche zu finden wurde in der aktuellen Version von XElf2 implementiert. Das Prinzip besteht darin, auf einem äquidistanten Grid grob numerische Gradienten zu bestimmen.

Das Grid wird in einem ersten Schritt mit den zu den Punkten gehörenden *ELF*-Werten gefüllt. In einem zweiten Schritt werden nun die numerischen Gradienten gebildet, in dem, ausgehend von einem Zentralpunkt, in oktaedrischer Koordination jeweils die Differenz der *ELF*-Werte an den Punkten über und unter, links und rechts, vor und hinter dem Zentralpunkt gebildet wird, und durch die *reale* Distanz der Punktpaare dividiert wird.

Punkte, für die der numerische Gradient in der Größenordnung 10^{-4} bis 10^{-3} liegt, werden im `raw`-Format in eine temporäre Datei gespeichert und bedürfen noch der Sortierung. Danach kann diese Datei als Input für den weiter unten beschriebenen ReFine-Algorithmus dienen.

Zentraler Bestandteil dieses Algorithmus' ist ein Grid, dessen Ausmaße so bestimmt sein sollten, dass sie alle *ELF*-Werte $\eta(\mathbf{r}) > 0.5$ einschliesßen. Das ist zugleich der größte Vorteil als auch der gravierendste Nachteil dieser Methode. Nachteil dieser Methode ist der *wirklich enorme* Speicherbedarf.

Vernünftige Gridauflösungen von 0.01 bis 0.001 erfordern für kleine Moleküle wie H_2O oder CH_4 zwischen 10 MByte und 900MByte. Während ersterer Wert auch HomePCs keine Schwierigkeiten bereitet, findet man letztere Speicherbestückung schon nur noch bei eingefleischten Power-Usern oder bei größeren Workstations.

Auf der *Habenseite* steht ganz klar die höhere Geschwindigkeit. Während einer gridlosen Rastersuche würden die *ELF*-Werte einundderselben Stelle mehrmals berechnet. Das Grid wird jedoch nur einmal vollständig gefüllt. Danach werden die benötigten Werte aus dem Grid gelesen, was *deutlich* schneller ist, als die Neuberechnung.

Hinzu kommt die *vollständige Parallelisierbarkeit*. Teilt man das Grid in mehrere Subgrids, die auf je einem Node laufen, so ist keine Kommunikation zwischen den Nodes notwendig. So kann man auch das Speicherproblem gering halten, da jeder Node nur „seinen“ Bereich kennen muss. Um die gefundenen Ergebnisse dann zusammenzuführen, gibt es die Methode `MergeFiles` (siehe Abschnitt C.9, Seite 63).

2.2.2.3 ReFine-Algorithmus

Es ist recht schwierig am Anfang einer Sattelpunktsuche ein sinnvolles Maß für den Zielgradienten zu finden. Setzt man ihn zu niedrig an, werden „schwierige“ Sattelpunkte nicht gefunden. Setzt man ihn zu hoch an, werden die Punkte nicht genau genug gefunden, oder gar Punkte gefunden, die gar keine Sattelpunkte sind. Das ist in der Numerik kein unbekanntes Problem.

Genauso sind die Sattelpunkte, die aus der OctaedricGrid-Methode gewonnen werden nur grobe Schätzwerte, die noch verfeinert werden müssen.

Die ReFine-Methode in der aktuellen Version fängt mit einem sehr kleinen Zielgradienten an und erhöht diesen bei Misserfolg adaptiv, so dass nach zwei bis drei Versuchen der Sattelpunkt mit der maximal erreichbaren Genauigkeit gefunden werden kann. So können Sattelpunkte mit Gradienten um die 10^{-18} gefunden werden, wogegen man sich bei nichtadaptiven Standardalgorithmen über Gradienten von 10^{-8} schon freuen durfte.

2.3 Grenzflächen

Hat man nun alle Sattelpunkte beisammen, geht es daran, die Grenzflächen zu konstruieren. Auch hier tuen sich Welten zwischen der eher simplen Konstruktion der atomaren Grenzflächen und den hochkomplexen Lokalisationsgrenzflächen auf.

2.3.1 Konstruktion einfacher Grenzflächen

Das Grundprinzip der Konstruktion von Grenzflächen ist sowohl bei *AIM* als auch *ELF* das gleiche: Ausgehend von dem Sattelpunkt werden Gradientenpfade kontruiert. Dazu werden die Eigenwerte und Eigenvektoren des Hessians am Ort des Sattelpunktess bestimmmt. Die Eigenvektoren, die zu den zwei negativen Eigenwerten gehören, spannen die *Eigenebene* auf. In diese Eigenebene wird ein Kreis mit (theoretisch unendlich) kleinem Radius gelegt (nennen wir ihn *Eigenkreis*), von dessen Randpunkten die Gradientenpfade starten. Das ist notwendig, da am Sattelpunkt selbst der Gradient zu klein ist, um eine Richtung anzugeben.

Jeder Gradientenpfad $\mathbf{x}(s)$ ist das Integral der Differentialgleichung

$$\frac{d\mathbf{x}(s)}{ds} = \nabla\rho(\mathbf{x}(s)) \tag{2.1}$$

für den Anfangswert $\mathbf{x}(0) = r_0$, dem Radius des Eigenkreises [11]. Also sind die Punkte $\mathbf{x}(s)$ auf dem Gradientenpfad gegeben als

$$\mathbf{x}(s) = \mathbf{x}(0) + \int_0^s \nabla\rho(\mathbf{x}(t))\, dt \tag{2.2}$$

oder ausgedrückt mit Hilfe der Pfadlänge l (die Bogenlänge entlang eines Gradientenpfades)

$$d\mathbf{x} = \frac{\nabla\rho(\mathbf{x})}{|\nabla\rho(\mathbf{x})|} \cdot dl. \tag{2.3}$$

Gleichung (2.3) ist der Ausdruck, der auch implementiert wurde – der sog. *stabilisierte Eulerschritt* [15]. Die Schrittlänge $s = dl$ sollte sinnvollerweise klein genug gewählt werden, damit der Gradientenpfad auch „glatt“ ist.

Soweit ist die Konstruktion der Grenzflächen sowohl bei *AIM* als auch bei *ELF* vollkommen identisch.

2.3.1.1 Atomare Grenzflächen

Seien $\mathbf{e}_1$ und $\mathbf{e}_2$ die Eigenvektoren, die zu den beiden negativen Eigenwerten des Hessian gehören, und sei $\mathbf{r}_{CP}$ der Ortsvektor des kritischen Punktes, so gilt für die Punkte $\mathbf{p}$ auf der Kreisbahn des Eigenkreises mit dem Radius r

$$\mathbf{p}(\alpha_i) = \mathbf{r}_{CP} + r \cdot (\cos(\alpha_i) \cdot \mathbf{e}_1 + \sin(\alpha_i) \cdot \mathbf{e}_2)\,. \tag{2.4}$$

Der Winkel α wird dabei im Intervall $[0; 2\pi]$ in n_a-viele Winkelabschnitt unterteilt:

$$\alpha_i = i \cdot \frac{2\pi}{n_a} \quad (i = 1, 2, 3...n_a). \tag{2.5}$$

Damit entstehen n_a-viele äquidistante Startpunkte für Gradientenpfade auf dem Kreisumfang.

Der Radius r des Eigenkreises wird jetzt so gewählt, dass für alle i gilt

$$\nabla\rho\,(\mathbf{p}(\alpha_i)) > 10^{-3}, \tag{2.6}$$

d.h. es ist gewährleistet, dass an jedem Startpunkt für einen Gradientenpfad der Gradient von ρ groß genug ist, um einen EULER-Schritt zu ermöglichen.

Im Programm wird vom Benutzer lediglich die Angabe von n_a, der Anzahl an Gradientenpfaden und n_l, der Anzahl der Schritte der Länge s erwartet. Dann werden zuerst die Eigenvektoren bestimmt, und alle α_i in einer Tabelle abgelegt (Minimierung der Aufrufe von trigonometrischen Funktionen). Durch *LineStepping* wird sodann ein geeignetes r ermittelt. Dann werden für alle α_i n_l-viele EULER-Schritte der Länge s berechnet und abgespeichert. Das Ergebnis ist eine Folge von (x, y, z)-Punkten, die die einzelnen Punkte auf den jeweiligen Gradientenpfaden sind (Abb. 2.1).

2.3.2 Konstruktion komplexer Grenzflächen

Rein theoretisch sollte die Konstruktion der Lokalisationsgrenzflächen auf die selbe Art und Weise wie bei den atomaren Grenzflächen möglich sein, denn auch sie sind nichts weiter als eine Anzahl von Gradientenpfaden, die von einem Sattelpunkt ausgehen. Jedoch gäbe es kein eigenes Kapitel über die Lokalisationsgrenzflächen, wenn das so einfach wäre. Mehrere Widrigkeiten erfordern aufwendige Algorithmen, die im folgenden beschrieben werden sollen.

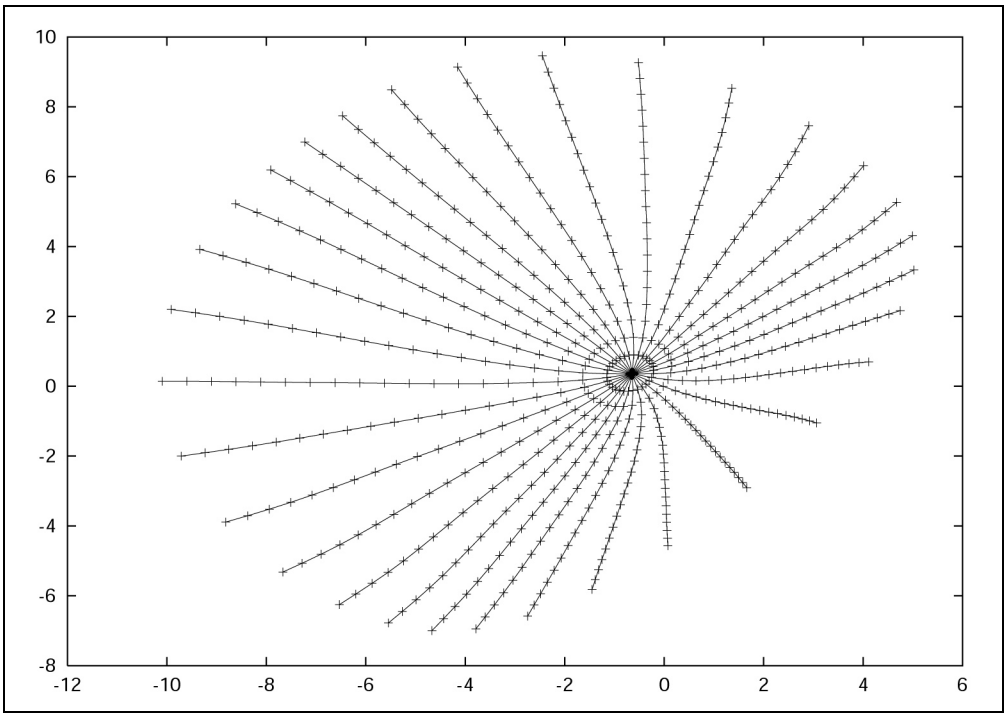


Abbildung 2.1: Schematische Darstellung einer konstruierten atomaren Grenzfläche.

2.3.2.1 Der erste Schritt

Die Umgebung eines *ECPs* ist keineswegs kreisförmig genug, um für den ersten Schritt, wie bei den atomaren Grenzflächen, einen Kreis zu rechtfertigen. Es kommt im Gegenteil nahezu immer vor, dass sich in der Eigenebene in eine Richtung ein flaches Tal erstreckt, während man sich in einer anderen Richtung betrachtet eher am Grunde eines Brunnenschachtes wähnt.

Schon seit der ersten Programmversion XELF1 wurde deshalb ein adaptives Verfahren implementiert. Grundsätzlich starten die Gradientenpfade wieder von einem Punkt $\mathbf{p}(\alpha_i)$ aus (Gleichung (2.4)), jedoch wird dieses Mal r für jedes α_i neu bestimmt, so dass für jedes α_i ein r_i bestimmt wird für das gilt

$$\nabla\eta\left(\mathbf{p}(\alpha_i)\right) = \nabla\eta\left(\mathbf{r}_{CP} + r_i \cdot (\cos(\alpha_i) \cdot \mathbf{e}_1 + \sin(\alpha_i) \cdot \mathbf{e}_2)\right) > 10^{-3}. \qquad (2.7)$$

Meist entsteht so eine Art Eigenellipsoid, nicht selten jedoch weitaus komplexere Formen, hauptsächlich bestimmt durch die Symmetriegruppe des Moleküls.

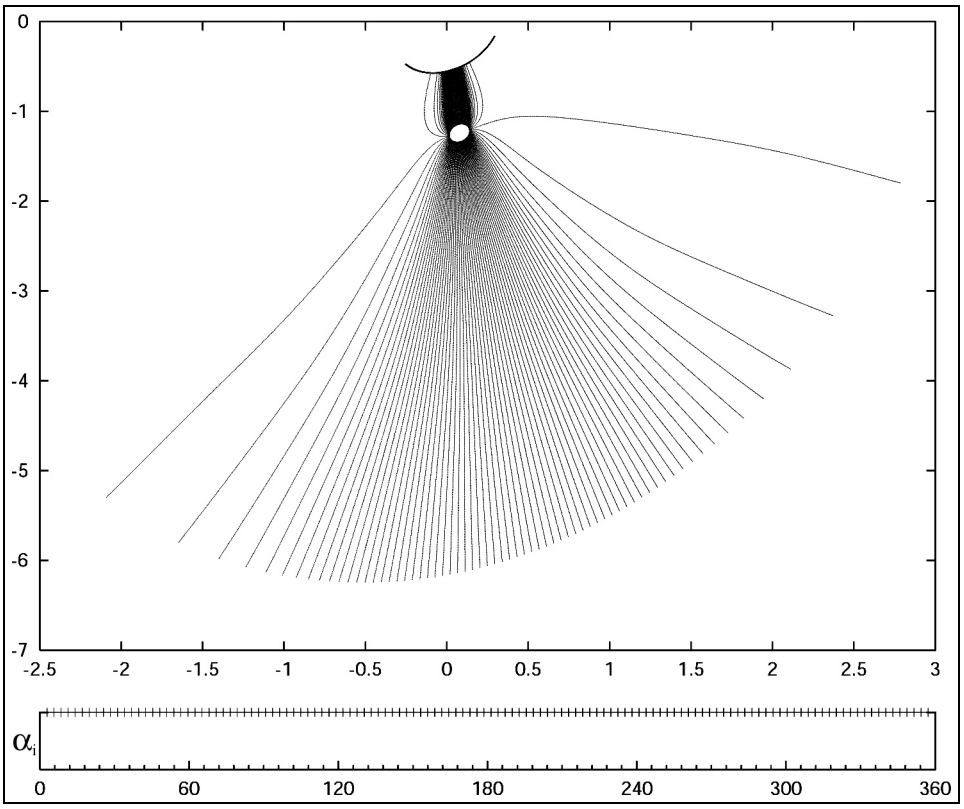


Abbildung 2.2: 2D-Projektion einer Lokalisationsgrenzfläche von H_2O (oben) mit äquidistanten α_i (unten).

2.3.2.2 Kritische Winkel und Bifurkationszonen

Bei der Konstruktion der Grenzflächen ergibt sich, nachdem man nun den Initialschritt adaptiv bestimmt hat, folgendes Phänomen: Geht man kontinuierlich die α_i-Werte durch, so gibt es zwei aufeinanderfolgende Winkel, deren Gradientenpfade in vollkommen verschiedene Richtungen zeigen und zwischen denen eine Lücke entsteht (Abb. 2.2). Egal wie viele oder wie enge Winkel man konstruiert, es bleibt immer eine Lücke.

Zwischen diesen beiden Winkeln muß also eine Art „kritischer Winkel“ liegen, dessen Gradientenpfad die Lücke schließt. Anders ausgedrückt, der kritische Winkel erzeugt einen Gradientenpfad, der auf einen Bifurkationspunkt in der *ELF* zuläuft. In einer infinitesimalen Umgebung dieser Stelle zeigt der Gradient in zwei entgegengesetzte Richtungen, der Gradientenpfad müsste sich also am Bifurkationspunkt teilen. Da das der EULER-Algorithmus nicht vorsieht, hängt es vom Startwinkel α_i ab, ob der Gradientenpfad haarscharf rechts oder haarscharf links am Bifurkationspunkt vorbeiläuft. Deshalb

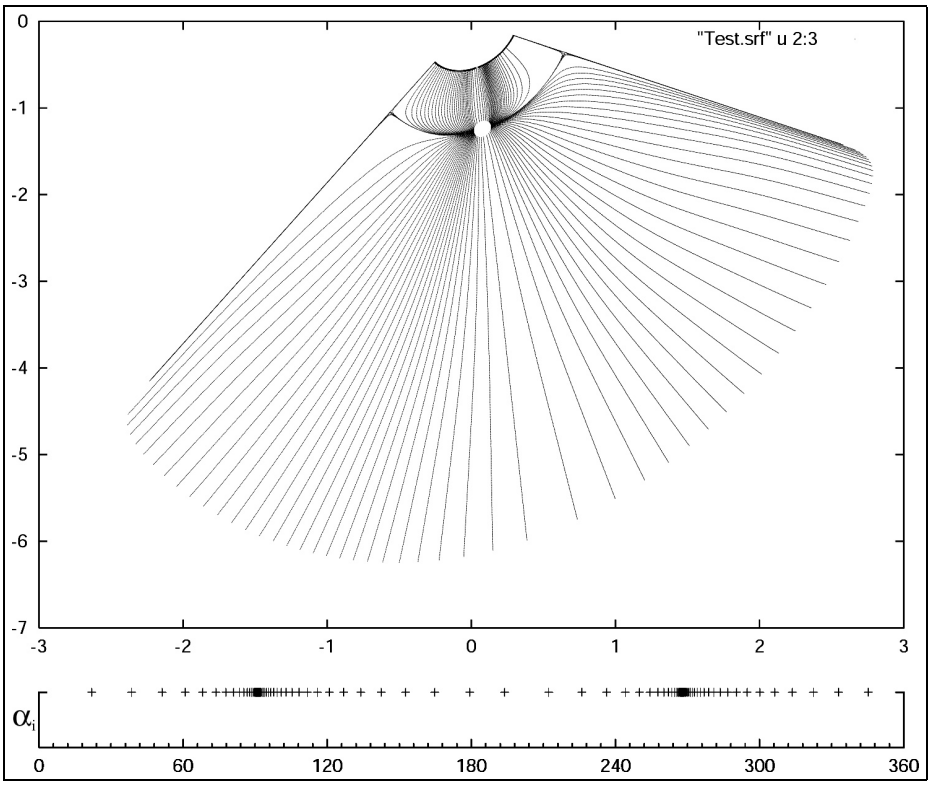


Abbildung 2.3: 2D-Projektion einer Lokalisationsgrenzfläche von H_2O (oben) mit adaptiven α_i (unten).

bleibt immer eine kleine Lücke.

Meine Erfahrungen haben gezeigt, dass es notwendig ist, den kritischen Winkel auf bis zu 10^{-8} Bogenmaß genau zu treffen, um die Lücke wenigstens einigermaßen zu schließen. Das würde aber eine sehr große Zahl an Gradientenpfaden erzeugen, deren Handhabung die weiteren Rechnungen stark in die Länge ziehen würde. Besonders ärgerlich ist das, da nur ein bisschen von den kritischen Winkeln entfernt die eher einfache Struktur der Fläche eine solche Genauigkeit gar nicht erforderlich macht.

Um dieses Problem zu lösen wird ein Weg beschrieben, der das Problem zwar recht elegant löst, an anderer Stelle jedoch dafür einige Probleme bereitet. Die naheliegendste Lösung ist es, nicht mehr äquidistante α_i zu verwenden. Um die kritischen Winkel herum sind sehr viele α_i nötig, ansonsten eher wenige. Um das zu bewerkstelligen, werden, nachdem die kritischen Winkel durch *LineStepping* bestimmt wurden, um diese herum Exponentialkurven konstruiert, die die variable Punktdichte erzeugen (Abb. 2.3).

Dazu wird zunächst jeweils zwischen $\alpha_i = 0$ und dem ersten kritischen

Winkel eine e-Funktion der Form $y = -e^{-x}$ im Intervall $x \in [-8 : 0]$ mit der entsprechenden Anzahl an äquidistanten x-Werten generiert und y danach so skaliert, dass $y \in [0 : \alpha_{crit}]$ liegt. Durch die besondere Form der e-Funktion werden die Punkte zum kritischen Winkel hin dichter.

Als nächstes wird für den Bereich vom aktuellen kritischen Winkel bis zur Mitte zwischen diesem und dem nächsten kritischen Winkel eine Funktion der Form $y = e^x$ im Intervall $x \in [0 : 5]$ generiert und entsprechend skaliert. Von hier bis zum nächsten kritischen Winkel wird wieder die erst genannte e-Funktion verwendet, bis alle kritischen Winkel berücksichtigt wurden. Die Form der so erzeugten Winkelfunktion ist in Abb. 2.6 zu sehen.

2.3.3 Analytische Grenzflächen

Nun liegen die Grenzflächen als eine Serie von diskreten Punkten vor, abgespeichert auf der Festplatte. Doch läßt sich damit nur recht umständlich rechnen. Es wäre indes viel praktischer, analytische Ausdrücke für die Grenzflächen zu verwenden. Obwohl es im Bereich der reinen *AIM*-Integration bereits Ansätze gibt, Bassins ohne explizite Berechnung der Grenzflächen zu integrieren [16], erschien dem Autor dieser Arbeit ein Ansatz mit analytischen Grenzflächen als vielversprechender.

Eine Grenzfläche ist an sich ein zweidimensionales Objekt (Abb. 2.4). Ausgehend von einem Aufpunkt (praktischerweise nimmt man hier den kritischen Punkt der Grenzfläche) kann jeder Punkt p auf der Fläche beschrieben werden durch:

$$p_x = s_x(\theta, l) \tag{2.8}$$
$$p_y = s_y(\theta, l) \tag{2.9}$$
$$p_z = s_z(\theta, l) \tag{2.10}$$

oder in Vektorschreibweise

$$\mathbf{p} = \mathbf{s}(\theta, l) \tag{2.11}$$

Dabei ist θ ein Winkel auf dem Eigenkreis um den Kritischen Punkt und l ist die Bogenlänge entlang des Strahls, der zu diesem Winkel gehört. Ziel ist es jetzt, einen Ausdruck für $\mathbf{f}(\theta, l)$ zu erhalten.

Nach der Konstruktion der Grenzfläche liegen n_a-viele Strahlen mit jeweils n_l-vielen Schritten der Länge s vor. Jeder Strahl für sich ist eindimensional abhängig von l und nur parametrisch abhängig von θ

$$\mathbf{p} = \mathbf{s}_\theta(l) \tag{2.12}$$

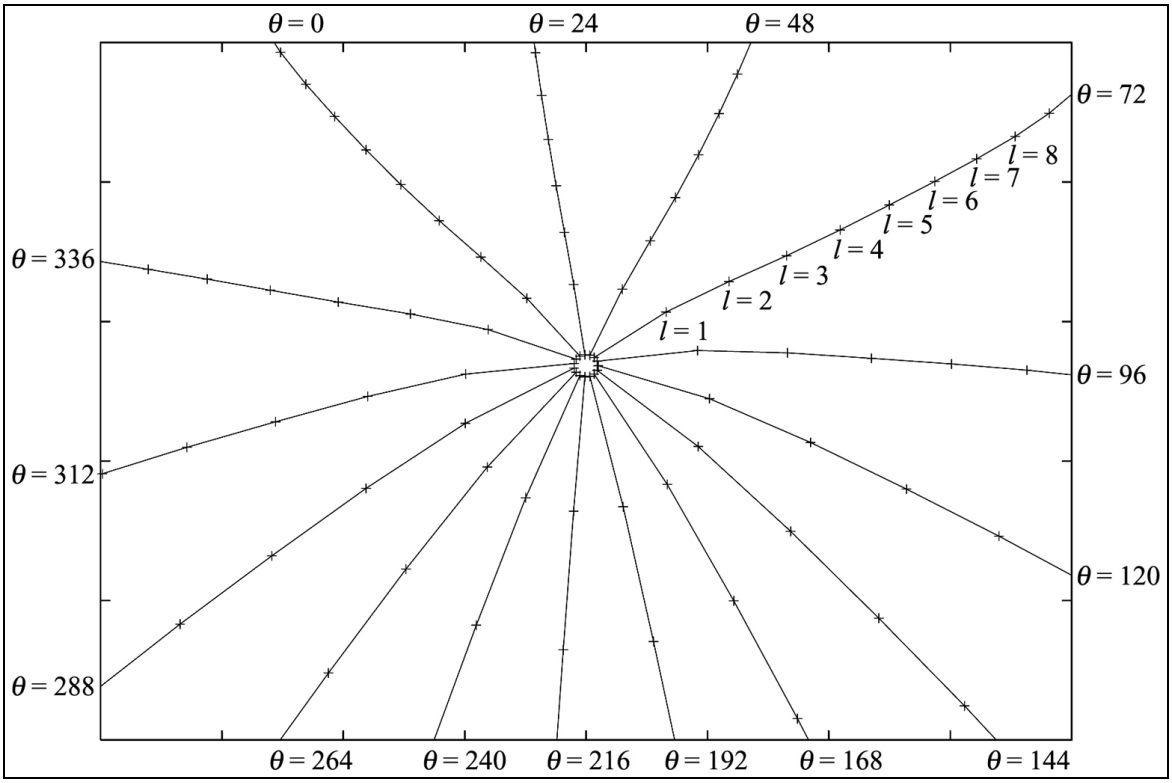


Abbildung 2.4: Schema der Fitstruktur.

Für jeden Strahl sind also zunächst drei Funktionen $s_{\alpha,\theta}(l)$ $(\alpha = x, y, z)$ gesucht.

In der Literatur gibt es die verschiedensten Ansätze zur Lösung eines Problems dieser Art [15]. Am einfachsten erscheint es, CHEBYSHEV-Polynome $T_n(x)$ (Abb. 2.5) zu verwenden:

$$T_n(x) = cos(n \cdot arccos(x)) \quad (x \in [-1; 1]) \tag{2.13}$$

Ein wie oben beschrieben konstruierter Gradientenpfad hat eine maximale Bogenlänge von l_{max} (bei unseren Rechnungen 25 bohr). Der Bereich $[0 : l_{max}]$ lässt sich deshalb leicht auf den zulässigen Definitionsbereich $[-1 : 1]$ der CHEBYSHEV-Polynome abbilden.

Dadurch, dass es sich dabei um *orthogonale Polynome* handelt, gestaltet sich die Bestimmung der Koeffizienten recht einfach [15]. Das Ergebnis ist für jeden der n_a-vielen Strahlen drei Sätze von N-vielen CHEBYSHEV-Koeffizienten, die jeweils die Funktionen für die x-, y- und z-Richtung bestimmen.

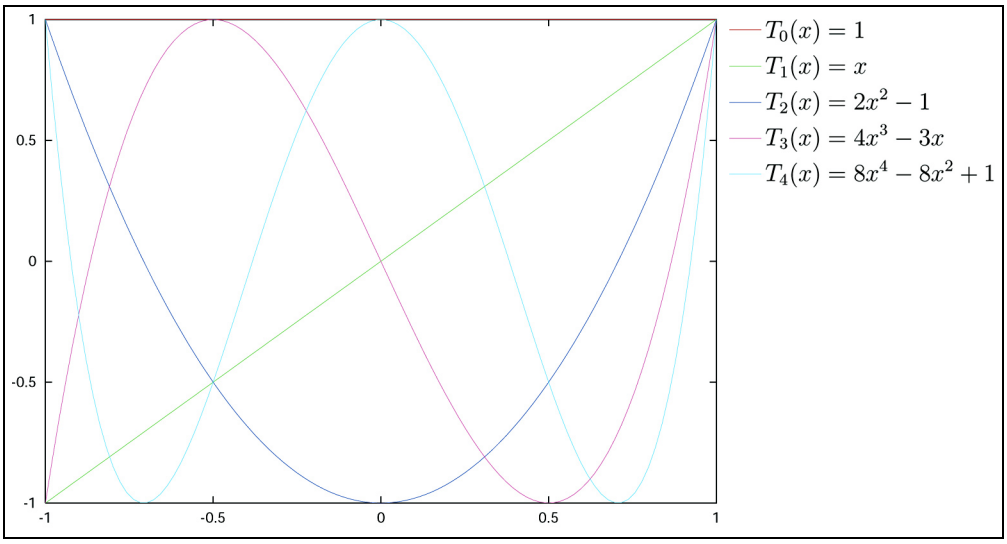


Abbildung 2.5: Einige CHEBYSHEV-Polynome

$$x_{i,\theta}(l) = \sum_{n=0}^{N} c^x_{n,i,\theta} T_n(l) \tag{2.14}$$

$$y_{i,\theta}(l) = \sum_{n=0}^{N} c^y_{n,i,\theta} T_n(l) \tag{2.15}$$

$$z_{i,\theta}(l) = \sum_{n=0}^{N} c^z_{n,i,\theta} T_n(l) \tag{2.16}$$

Dabei ist i die Nummer der Grenzfläche.

Durch die Art der Konstruktion sind die $c_{n,i,\theta}$ für jeweils gleiche n und i 2π-periodisch in θ. Um also auch noch zwischen den diskreten α_i zu interpolieren, werden die CHEBYSHEV-Koeffizienten einer Grenzfläche der gleichen Ordnung n durch eine FOURIER-Reihe entwickelt:

$$c_{n,i}(\theta) = \sum_{k=0}^{M} \left(a_{n,i,k}\cos(k\theta) + b_{n,i,k}\sin(k\theta)\right) \tag{2.17}$$

Der vollständige analytische Ausdruck für die Grenzfläche i ergibt sich somit zu:

$$x_i(\theta, l) = \sum_{n=0}^{N} \left(\sum_{k=0}^{M} \left(a^x_{n,i,k}\cos(k\theta) + b^x_{n,i,k}\sin(k\theta)\right) \right) T_n(l) \tag{2.18}$$

$$y_i(\theta, l) = \sum_{n=0}^{N} \left(\sum_{k=0}^{M} \left(a^y_{n,i,k} \cos(k\theta) + b^y_{n,i,k} \sin(k\theta) \right) \right) T_n(l) \tag{2.19}$$

$$z_i(\theta, l) = \sum_{n=0}^{N} \left(\sum_{k=0}^{M} \left(a^z_{n,i,k} \cos(k\theta) + b^z_{n,i,k} \sin(k\theta) \right) \right) T_n(l) \tag{2.20}$$

$$\mathbf{s}_i(\theta, l) = \begin{pmatrix} x_i(\theta, l) \\ y_i(\theta, l) \\ z_i(\theta, l) \end{pmatrix} \tag{2.21}$$

Durch die besondere Art des FOURIER-Polynoms sind für M-viele FOURIER-Terme $2M$-viele Gradientenpfade notwendig [17]. Und aus n_l Schritten der Länge s pro Gradientenpfad ergäben sich n_l-viele CHEBYSHEV-Koeffizienten. Davon werden jedoch nicht alle verwendet, sondern nur die, die den Gesamtfehler des CHEBYSHEV-Fits kleiner halten als 10^{-5}.

Um eine Grenzfläche also vollständig zu beschreiben sind in letzter Instanz M-viele Sinus- und M-viele Cosinus-Koeffizienten für jede der N-vielen CHEBYSHEV-Terme in alle drei Raumrichtungen notwendig. Das geht einher mit einer Reduzierung des Speicherbedarfs, wie folgendes Beispiel zeigen soll: Eine atomare Grenzfläche sei berechnet worden mit 256 Gradientenpfaden mit jeweils 1000 Schritten der Länge 0.01 bohr. Jeder Schritt ergibt drei Raumkoordinaten, was in der Genauigkeit einer FORTRAN-Varibale doppelter Genauigkeit $3 \cdot 8 = 24$ Byte entspricht. Für die gesamte Fläche sind das also $256 \cdot 1000 \cdot 8 = 1,9$ MByte. Erfahrungsgemäß sind für einfache Flächen nur die ersten 15 CHEBYSHEV-Koeffizienten relevant, und aus den 256 Pfaden ergeben sich 128 Sinus- und 128 Cosinus-Koeffizieten für drei Raumrichtungen in doppelter Genauigkeit. Der Speicherbedarf für den Fit ist also $(128 \cdot 2) \cdot 15 \cdot (3 \cdot 8) = 90$ kByte.

2.3.3.1 Gefaltete Flächen

Zur Bestimmung der FOURIER-Koeffizienten dienen die Integrale

$$a_k = \frac{1}{\pi} \int_0^{2\pi} f(x) \cos(k \cdot x)\, dx \quad (k = 0, 1, ..., n) \tag{2.22}$$

$$b_k = \frac{1}{\pi} \int_0^{2\pi} f(x) \sin(k \cdot x)\, dx \quad (k = 1, 2, ..., n) \tag{2.23}$$

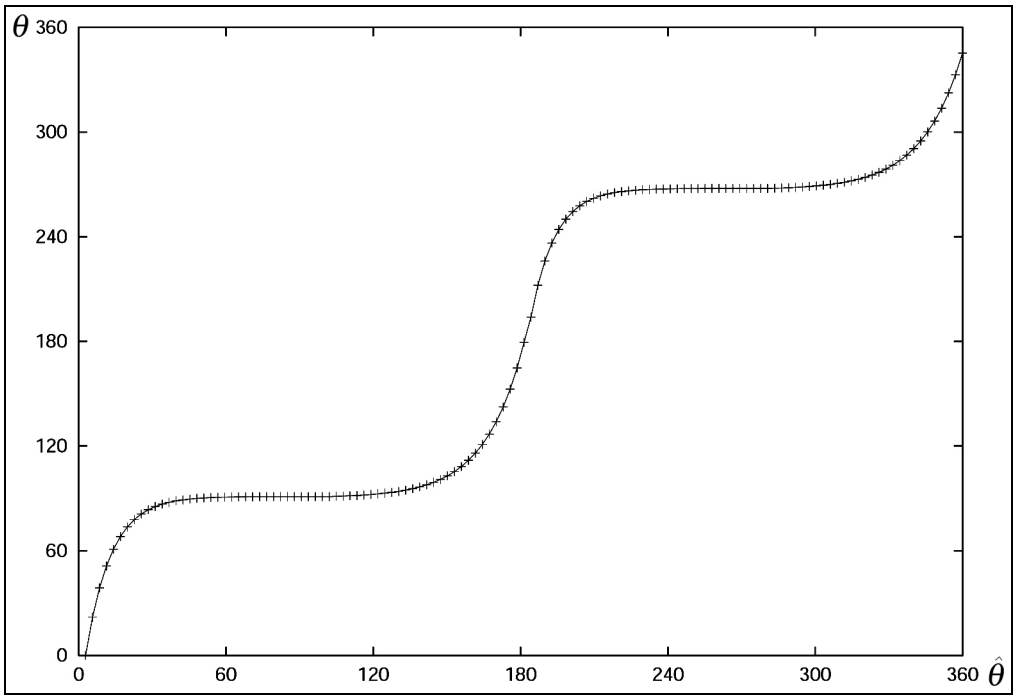


Abbildung 2.6: Zusammenhang zwischen Pseudowinkel $\hat{\theta}$ Realwinkel θ.

Unter Ausnutzung der *Trapezregel* [17] kann das Integral in eine Summe über diskrete, *äquidistante* Stützstellen umgeformt werden:

$$a_k = \frac{2}{M} \sum_{j=1}^{M} f(x_j) \cos(k \cdot x_j) \quad (k = 0, 1, ...) \tag{2.24}$$

$$b_k = \frac{2}{M} \sum_{j=0}^{M} f(x_j) \sin(k \cdot x_j) \quad (k = 1, 2, ...) \tag{2.25}$$

$$x_j = j \cdot \frac{2\pi}{M} \quad (j = 0, 1, 2, ..., M) \tag{2.26}$$

Dass die Stützstellen auch wirklich äquidistant sind, ist bei atomaren Grenzflächen und bei Lokalisationsgrenzflächen ohne Bifurkationszone auch so, bei Lokalisationsgrenzflächen mit Bifurkationszone aber nicht. Um trotzdem auch solche Flächen fitten zu können, wird im Fit-Algorithmus folgender Gedankengang implementiert: Wir bilden die nichtäquidistanten *realen* Winkel θ auf äquidistante *Pseudowinkel* $\hat{\theta}$ ab – wir falten quasi den Werteraum. Jedem äquidistanten Pseudowinkel ist demnach einer der nicht-äquidistanten Realwinkel zugeordnet (Abb. 2.6).

So kann die Integration weiterhin durch die Trapezregel ausgeführt werden, jedoch muß man bei allen weiteren Rechnungen mit diesen Fits nachhalten, welche Fläche gefaltet ist um gegebenenfalls zwischen den Real- und Pseudowinkeln umrechnen zu können (siehe Abschnitt 2.4.3.1, Seite 33).

2.3.4 Gefittete Corebassins

Wie schon oben besprochen, gestaltet sich das Finden von *ECPs* als recht schwierig. Vor allem jedoch die *ECPs* von Corebassins sind nur mit großem Aufwand zu bestimmen. Jedoch kann man in erster Näherung annehmen, dass die Corebassins die Form einer Kugel haben.

Es werden Kugeln in der Weise um einen Kern angefittet, dass die Summe der *ELF*-Werte an den Punkten auf der Kugeloberfläche ein lokales Minimum annimmt. Es werden nur noch Mittelpunkt und Radius gespeichert.

2.4 Subraumintegration

Das Integrieren innerhalb eines Subraums, dessen Form man erst zur Laufzeit kennen kann, stellt das Hauptproblem der Populationsanalyse dar. Konnte die 2D-Version des Programs (XELF1) noch auf einem Grid mit einer Art intelligentem Kästchenzählen gute Ergebnisse liefern, solange man nur die Auflösung des Grids hoch genug wählte [14], verbietet sich das bei dem 3D-Fall aus mehreren Gründen.

Zum einem der *Speicherbedarf*: Erfahrungen mit XELF1 haben gezeigt, dass eine Grid-Auflösung von 0.005 bohr notwendig sind, um bei der Integration zwei gesicherte Nachkommastellen angeben zu können. Die Ausdehnung des Grids sollte zudem so gewählt sein, dass alle Punkte mit $\rho(\mathbf{r}) < 10^{-10}$ innerhalb des Grids liegen. Für das kleine, nichtlineare Molekül H_2O ergäbe sich so ein Quader mit den ungefähren Kantenmaßen $17.5 \times 12 \times 12$ bohr. Bei einer Auflösung von 0.005 bohr ergäben sich die Ausmaße $3500 \times 2400 \times 2400$ Punkte. Da jeder Punkt des Grids 1 Byte umfassen muss, um eine Karte mit maximal 255 Grenzflächen darstellen zu können, kommt man auf einen Speicherbedarf von 18.8 GByte (!!!). Von Molekülen wie CH_4 oder gar längeren Alkangerüststrukturen ganz zu schweigen.

Zum anderen würde die eigentliche Integration recht aufwendig werden, wenn man ein vernünftiges Quadraturverfahren in drei Dimensionen anwendet. Ausgehend von der 2D-Integration in XELF1, würde die Integration von H_2O mit dem auf 3D hochskalierten Verfahren schätzungsweise 143 Stunden (!!!) brauchen.

Deshalb wurde ein ganz anderer Ansatz verfolgt, inspiriert von Verfahren, wie sie bei der reinen *AIM*-Integration genutzt werden [18].

2.4.1 Sphärische Integration von Subräumen

Ausgehend von einem Integrationsaufpunkt $\mathbf{p}_0$ werden sphärisch 1D-*Integrationsstrahlen* $\mathbf{R}_{\vartheta\varphi}$ ausgesandt, deren Richtung von dem Kugelkoordinatenwinkeln ϑ und φ abhängen. Jeder Punkt auf diesem Integrationsstrahl kann also ausgedrückt werden als

$$\mathbf{R}_{\vartheta\varphi}(r) = \mathbf{p}_0 + r \cdot \mathbf{e}(\vartheta, \varphi) \tag{2.27}$$

mit r der Länge des Integrationsstrahls vom Ursprung bis zum aktuellen Punkt und dem Einheitsvektor $\mathbf{e}(\vartheta, \varphi)$, der die Richtung entsprechend der Winkel ϑ und φ angibt.

Trifft ein solcher Integrationsstrahl auf eine Grenzfläche, so ist er selbst und damit seine Länge endlich. Genausogut kann ein Integrationsstrahl auch

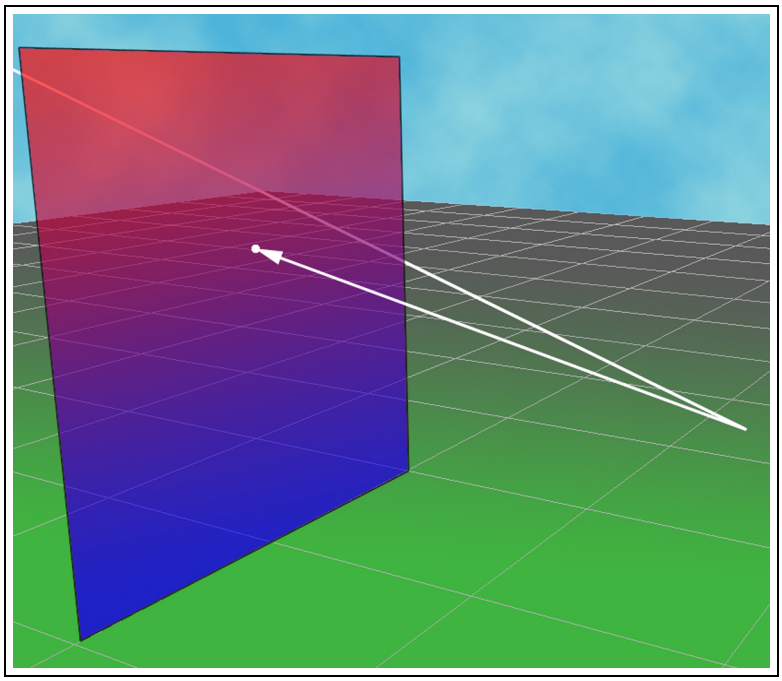


Abbildung 2.7: Schema der sphärischen Integration..

ins Unendliche gehen. In jedem Fall aber ist seine Länge ebenfalls eine Funktion der Kugelkoordinatenwinkel:

$$|\mathbf{R}_{\vartheta\varphi}| = r(\vartheta, \varphi). \tag{2.28}$$

Zur Veranschaulichung siehe auch Abb. 2.7.

Entlang des Integrationsstrahls $\mathbf{R}_{\vartheta\varphi}$ mit der Länge $r(\vartheta, \varphi)$ wird jetzt eindimensional die Dichte integriert:

$$I^1 = I^1(\vartheta, \varphi) = \int_0^{r(\vartheta,\varphi)} \rho\left(r \cdot \mathbf{e}(\vartheta, \varphi)\right)\, r^2\, dr. \tag{2.29}$$

Die Integration über die Kugelkoordinatenwinkel erfolgt jetzt nach

$$I^3 = \int_0^{\pi} \int_0^{2\pi} I^1(\vartheta, \varphi)\, d\varphi\, \sin(\vartheta)\, d\vartheta \tag{2.30}$$

$$= \int_0^{\pi} \int_0^{2\pi} \int_0^{r(\vartheta,\varphi)} \rho\left(r \cdot \mathbf{e}(\vartheta, \varphi)\right)\, r^2\, dr\, d\varphi \sin(\vartheta)\, d\vartheta. \tag{2.31}$$

i	Theta(i)	Phi(i)	w(i)	i	Theta(i)	Phi(i)	w(i)
1	1.57080	0.00000	0.30951E-03	418	0.67562	4.41765	0.18498E-02
2	1.57080	3.14159	0.30951E-03	419	2.46597	1.27606	0.18498E-02
3	1.57080	1.57080	0.30951E-03	420	2.46597	1.86553	0.18498E-02
4	1.57080	4.71239	0.30951E-03	421	2.46597	5.00712	0.18498E-02
5	0.00000	0.00000	0.30951E-03	422	2.46597	4.41765	0.18498E-02
6	3.14159	0.00000	0.30951E-03	423	0.92928	1.34206	0.18498E-02
7	0.95532	0.78540	0.18524E-02	424	0.92928	1.79953	0.18498E-02
8	0.95532	2.35619	0.18524E-02	425	0.92928	4.94112	0.18498E-02
9	0.95532	5.49779	0.18524E-02	426	0.92928	4.48366	0.18498E-02
10	0.95532	3.92699	0.18524E-02	427	2.21231	1.34206	0.18498E-02
11	2.18628	0.78540	0.18524E-02	428	2.21231	1.79953	0.18498E-02
12	2.18628	2.35619	0.18524E-02	429	2.21231	4.94112	0.18498E-02
13	2.18628	5.49779	0.18524E-02	430	2.21231	4.48366	0.18498E-02
14	2.18628	3.92699	0.18524E-02	431	1.38812	0.65422	0.18498E-02
15	1.47847	0.78540	0.18718E-02	432	1.38812	2.48738	0.18498E-02
16	1.47847	2.35619	0.18718E-02	433	1.38812	5.62897	0.18498E-02
17	1.47847	5.49779	0.18718E-02	434	1.38812	3.79581	0.18498E-02
18	1.47847	3.92699	0.18718E-02	435	1.75347	0.65422	0.18498E-02
19	1.66312	0.78540	0.18718E-02	436	1.75347	2.48738	0.18498E-02
20	1.66312	2.35619	0.18718E-02	437	1.75347	5.62897	0.18498E-02
21	1.66312	5.49779	0.18718E-02	438	1.75347	3.79581	0.18498E-02
22	1.66312	3.92699	0.18718E-02	439	0.92928	0.22873	0.18498E-02
23	0.78965	0.13019	0.18718E-02	440	0.92928	2.91286	0.18498E-02
24	0.78965	3.01140	0.18718E-02	441	0.92928	6.05445	0.18498E-02
25	0.78965	6.15299	0.18718E-02	442	0.92928	3.37033	0.18498E-02
26	0.78965	3.27179	0.18718E-02	443	2.21231	0.22873	0.18498E-02
27	2.35194	0.13019	0.18718E-02	444	2.21231	2.91286	0.18498E-02
28	2.35194	3.01140	0.18718E-02	445	2.21231	6.05445	0.18498E-02
29	2.35194	6.15299	0.18718E-02	446	2.21231	3.37033	0.18498E-02
30	2.35194	3.27179	0.18718E-02	447	0.42941	0.42610	0.17139E-02
31	0.78965	1.44060	0.18718E-02	448	0.42941	2.71549	0.17139E-02
32	0.78965	1.70099	0.18718E-02	449	0.42941	5.85708	0.17139E-02
33	0.78965	4.84258	0.18718E-02	450	0.42941	3.56769	0.17139E-02
34	0.78965	4.58220	0.18718E-02	451	2.71219	0.42610	0.17139E-02
35	2.35194	1.44060	0.18718E-02	452	2.71219	2.71549	0.17139E-02

Abbildung 2.8: Ausschnitt aus einer LEBEDEV-Tabelle der Ordnung 9 mit $N = 590$.

2.4.2 Lebedev-Integration

Numerisch wird die Integration über die Kugelkoordinatenwinkel in XELF2 durch die LEBEDEV-Integration [19] (eine Anwendung der SOBOLEV-Quadratur [20]) gelöst. Dabei wird die kontinuierliche Integration über die Winkel ϑ und φ ersetzt durch eine endliche Summe von N-vielen gewichteten Samplepoints:

$$\int \sin(\vartheta)\, d\vartheta \int d\varphi\, f(\vartheta, \varphi) = \sum_{i=1}^{N} f(\vartheta_i, \varphi_i) \cdot w_i \tag{2.32}$$

Dreh und Angelpunkt dieser SOBOLEV-Quadratur sind tabellierte Triplets $(\vartheta_i, \varphi_i, w_i)$. Die Samplepoints auf der Kugeloberfläche müssen so verteilt und gewichtet werden, dass der Fehler minimal ist. Der Berechnung solcher Tabellen hat LEBEDEV sein Leben gewidmet und in der Zeit von 1976 bis heute immer ausgefeiltere und genauere Listen von Integrationswinkeln und -gewichten berechnet und veröffentlicht [21, 22, 23, 24, 19]. Abb. 2.8 zeigt einen Ausschnitt aus einer solchen LEBEDEV-Tabelle.

In unserem Fall wird durch die LEBEDEV-Integration

$$I^3 = \int_0^{\pi} \int_0^{2\pi} \int_0^{r(\vartheta,\varphi)} \rho\left(r \cdot \mathbf{e}(\vartheta,\varphi)\right)\, r^2\, dr\, d\varphi\, \sin(\vartheta)\, d\vartheta \tag{2.33}$$

$$= \sum_{i=1}^{N} w_i \cdot \int_0^{r(\vartheta_i,\varphi_i)} \rho\left(r \cdot \mathbf{e}(\vartheta_i,\varphi_i)\right)\, r^2\, dr \tag{2.34}$$

Es bleibt also nur noch die 1D-Integration entlang des Integrationsstrahls zu lösen. Dies geschieht wie in der alten Version des Programs durch eine adaptive CLENSHAW-CURTIS-Quadratur, wobei die CHEBYSHEV-Koeffizienten durch eine *fast discrete cosine transformation (fDCT)* bestimmt werden [15].

Als schwierigstes Problem bleibt noch, dass für jedes Winkelpaar (ϑ_i, φ_i) die Länge des Integrationsstrahls $r(\vartheta_i, \varphi_i)$ neu bestimmt werden muß.

2.4.3 Surface-Intersection-Algorithmus

Ein Strahl durchstößt eine Grenzfläche dann, wenn der Abstand zu einem Punkt $\mathbf{p}(r)$ auf dem 1D-Integrationsstrahl und der 2D-Grenzfläche $\mathbf{s}(l,\theta)$ null wird [18]:

$$D_{\vartheta_i,\varphi_i} = \mathbf{p}(r)_{\vartheta_i,\varphi_i} - \mathbf{s}(l,\theta) = 0 \tag{2.35}$$

wobei

$$\mathbf{p}(r)_{\vartheta_i,\varphi_i} = \begin{pmatrix} x_0 + r \cdot \sin(\vartheta_i)\cos(\varphi_i) \\ y_0 + r \cdot \sin(\vartheta_i)\sin(\varphi_i) \\ z_0 + r \cdot \cos(\vartheta_i) \end{pmatrix} \tag{2.36}$$

$$\mathbf{s}(l,\theta) = \begin{pmatrix} x(l,\theta) \\ y(l,\theta) \\ z(l,\theta) \end{pmatrix} \tag{2.37}$$

mit r der Position auf dem 1D-Integrationsstrahl, mit dem Aufpunkt (x_0, y_0, z_0) und den quasianalytischen Ausdrücken für die Grenzfläche $x(l,\theta)$, $y(l,\theta)$ und $z(l,\theta)$ (siehe Abschnitt 2.3.3 auf Seite 22).

Also ist die Abstandsfunktion $D_{\vartheta_i,\varphi_i}$

$$\begin{aligned} D(l,\theta,r)_{\vartheta_i,\varphi_i} = & \quad (x(l,\theta) - x_0 - r \cdot \sin(\vartheta_i)\cos(\varphi_i))^2 \\ & + \quad (y(l,\theta) - y_0 - r \cdot \sin(\vartheta_i)\sin(\varphi_i))^2 \\ & + \quad (z(l,\theta) - z_0 - r \cdot \cos(\vartheta_i))^2 \end{aligned} \tag{2.38}$$

zu minimieren.

Da alle Komponenten analytisch bekannt sind, sind somit auch die analytischen Ableitungen verfügbar. Zur Minimierung wurde deshalb ein adaptives konjugiertes Gradientenverfahren nach POWELL verwendet.

Es gibt zwei Arten von Strahlen: *endliche*, die eine Grenzfläche durchstoßen, und *unendliche*, die nie eine Grenzfläche durchstoßen. Ob ein Strahl endlich oder unendlich ist, hängt von der Grenzfläche ab: einige Grenzflächen werden durchstoßen, andere nicht. Also muss für alle Grenzflächen $r(\vartheta_i, \varphi_i)$ bestimmt werden. Das kleinste von allen endlichen $r(\vartheta_i, \varphi_i)$ ist das für die Integration gewünschte.

Sind alle $r(\vartheta_i, \varphi_i)$ unendlich, sollte die Integration theoretisch von 0 bis ∞ gehen. In der Praxis nimm man ein maximales r_{max}, so dass gilt:

$$\rho\left(r_{max} \cdot \mathbf{e}(\vartheta, \varphi)\right) \quad < \quad 10^{-10} \quad \forall\left(\vartheta \in [0;\pi], \varphi \in [0;2\pi]\right) \tag{2.39}$$

$$\mathbf{e}(\vartheta, \varphi) \quad = \quad \begin{pmatrix} \sin(\vartheta)\cos(\varphi) \\ \sin(\vartheta)\sin(\varphi) \\ \cos(\vartheta) \end{pmatrix} \tag{2.40}$$

Es muß jedoch noch überprüft werden, ob wirklich $D_{\vartheta_i,\varphi_i} = 0$ ist, denn natürlich gibt es auch lokale Minima, die keinem Durchstoßpunkt entsprechen.

Während dieses Verfahren für die recht simplen interatomare Grenzflächen zuverlässig in akzeptabler Geschwindigkeit funktioniert, ergeben sich vielseitige Probleme bei den deutlich komplizierteren Formen der Lokalisationsgrenzflächen.

2.4.3.1 Startpunkte

Trotz jahrelanger Optimierung von Minimierungsroutinen kranken alle gradientenbasierten Algorithmen an ihrer Empfindlichkeit gegenüber dem Startpunkt. Jedoch ist es schwierig, einen geeigneten Startpunkt zu schätzen. Zum einen, da man erst zur Laufzeit weiß, wie und wo die Flächen liegen, zum anderen wird die Richtung des Integrationsstrahls aus der LEBEDEV-Tabelle entnommen, die auch erst zur Laufzeit erzeugt wird. Weiterhin muss man beachten, dass Lokalisationsflächen meist gefaltet vorliegen (siehe Abschnitt 2.3.3.1, Seite 25).

Zur Lösung dieses Problems werden zwei Wege beschritten. Um einen guten Startwert für den Winkel der Grenzfläche θ zu errechnen, wird der LEBEDEV-Einheitsvektor $\mathbf{e}_{Leb}$ in die Eigenebene der Grenzfläche, die von den

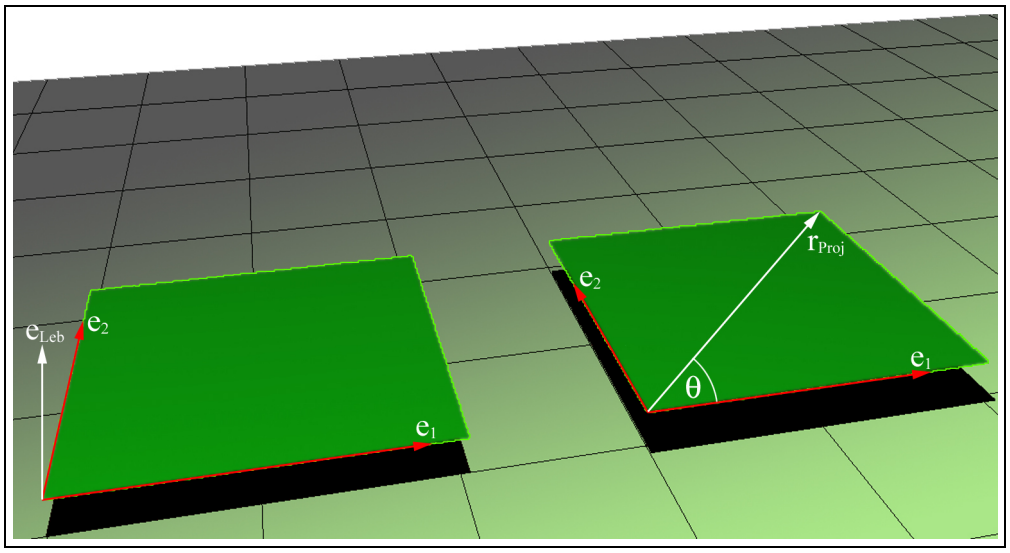


Abbildung 2.9: Projektion des LEBEDEV-Einheitsvektors in die Eigenebene der Grenzfläche, um einen Startwert für θ zu berechnen.

beiden Eigenvektoren $\mathbf{e}_1$ und $\mathbf{e}_2$ des $(3,-1)$-kritischen Punktes aufgespannt wird, projeziert:

$$\mathbf{r}_{Proj} = \begin{pmatrix} \mathbf{e}_1 \cdot \mathbf{e}_{Leb} \\ \mathbf{e}_2 \cdot \mathbf{e}_{Leb} \end{pmatrix} \tag{2.41}$$

$$\mathbf{r}_{Proj} \cdot \mathbf{e}_1 = |\mathbf{r}_{Proj}| \cdot |\mathbf{e}_1| \cdot \cos(\theta_{guess}) \tag{2.42}$$

$$\cos(\theta_{guess}) = \frac{\mathbf{r}_{Proj} \cdot \mathbf{e}_1}{|\mathbf{r}_{Proj}| \cdot |\mathbf{e}_1|} = \frac{\mathbf{r}_{Proj} \cdot \mathbf{e}_1}{\sqrt{(\mathbf{e}_1 \cdot \mathbf{e}_{Leb})^2 + (\mathbf{e}_2 \cdot \mathbf{e}_{Leb})^2}} \tag{2.43}$$

Siehe auch Abb. 2.9.

Bei ungefalteten Grenzflächen ist θ_{guess} schon so genau, dass bei der Minimierung so gut wie kein Schritt in θ-Richtung notwendig ist. Bei gefalteten Grenzflächen stellt θ_{guess} jedoch einen realen Winkel dar, so dass die Faltung noch berücksichtigt werden muss, um einen Startwert für die Minimierung, die ja mit gefalteten Pseudowinkeln $\hat{\theta}$ rechnet, zu erhalten. Um diese Umrechnung zu ermöglichen, werden die Werte für α_i von allen Grenzflächen im Speicher gehalten. Zwischen zwei benachbarten α_i wird linear interpoliert.

Es sollte an dieser Stelle erwähnt werden, dass es während der Minimierung (bzw. während des Intersection-Algorithmus') völlig gleichgültig ist, ob mit realen oder Pseudowinkeln gerechnet wird, da im Endeffekt nur die Länge des Integrationsstrahls als Ergebnis weiterverwendet wird. Bei welchem

Winkel die Fläche durchstoßen wurde und ob es sich dabei um den realen oder Pseudowinkel handelt spielt für die Integration keine Rolle. Durch die Struktur des Fits wird jedoch während der Minimierung zwangsläufig nur in äquidistanten Pseudowinkeln gerechnet. Nur während der Projektion des LEBEDEV-Einheitsvektors in die Eigenebene muss man die Faltung berücksichtigen.

Um Startwerte für r und l zu erhalten wird eine 2D-Gridsuche bei festem θ durchgeführt. Der Vorteil dabei ist, dass für beliebige Werte für r und l aber bei festem θ die CHEBYSHEV-Koeffizienten der Flächen gleich bleiben und nicht jedesmal neu berechnet werden müssen. So ist eine Gridsuche in akzeptabler Zeit möglich.

2.4.3.2 Konvergenzkriterium

Idealerweise würde bei einer Minimierung $D_{\vartheta_i,\varphi_i} = 0$ werden, sollte der Integrationsstrahl die Fläche wirklich durchstoßen. Im Reich der Numerik stellt sich jedoch die Frage, wie genau Null eigentlich Null genug ist. Auf der einen Seite steht da das Verlangen, eine Rechnung noch in der Lebensspanne des Benutzers zu vollenden, auf der anderen Seite sind eigentlich unendlich viele Schritte nötig, um ein Minimum exakt zu treffen.

Man kann vier Fälle eines Minimierungslaufes unterscheiden (Abb. 2.10). Der erste und einfachste Fall ist, wenn der Integrationsstrahl die Fläche fast senkrecht schneidet. Die resultierende Abstandsfunktion $D_{l_{min},\theta_{min}}(r)$ (l_{min} und θ_{min}, so dass $D(r)$ minimal wird) weist ein steiles Tal auf, in dem die Gradienten um das Minimum herum sehr schnell größere Werte annehmen. Die Minimierungsroutine benötigt nur wenige Schritte für eine erfolgreiche Minimierung, $D_{\vartheta_i,\varphi_i}$ wird wirklich nahezu Null und die Gradienten liegen in der Größenordnung von 10^{-18}.

Im zweiten Fall trifft der Integrationsstrahl die Fläche unter einem sehr flachen Winkel. Um das Minimum herum, ist die Änderung des Abstandes nur minimal, so dass in der Abstandsfunktion $D_{l_{min},\theta_{min}}(r)$ ein flaches Tal um das Minimum herum resultiert. Eine solche Topologie ist jedoch ein sehr schwieriger Fall für gradientenbasierte Algorithmen, denn im schlimmsten Fall (und der tritt aller Erfahrung nach sehr häufig auf) ist das Abbruchkriterium eines Mindestgradienten schon erfüllt, obwohl die Routine noch neben dem Minimum liegt. Das mag für einige Anwendungen nicht so schlimm sein, geht hier jedoch damit einher, dass $D_{\vartheta_i,\varphi_i}$ deutlich von Null verschieden ist. Trotzdem muß die Fläche als getroffen gelten.

Eine intuitive Lösung dieses Problems ist, eine Schwelle anzugeben, ab der die Abstandsfunktion als Null behandelt wird. Um die Schwelle adaptiv zu gestalten wird das Verhältnis von Abstand zu Länge eines Integrationsstrahls

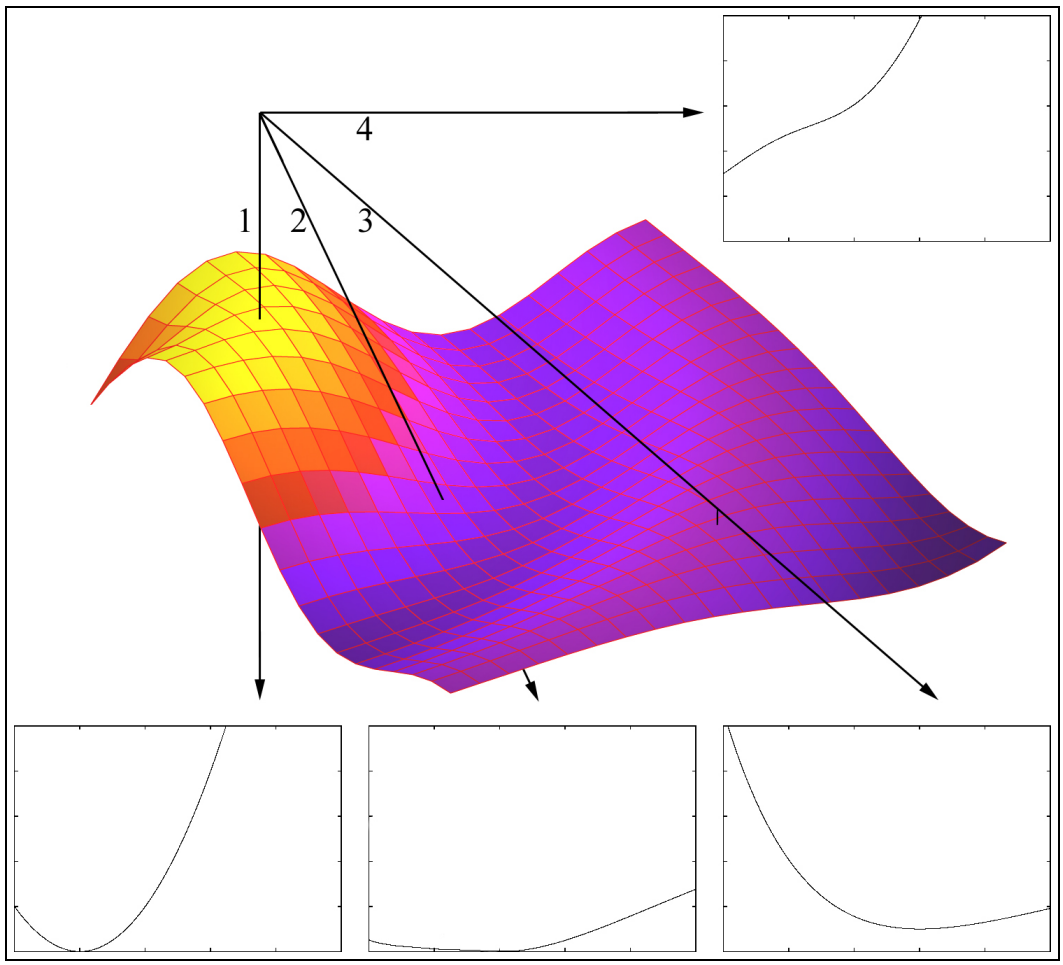


Abbildung 2.10: Schematische Darstellung der vier möglichen Fälle während der Minimierung von $D_{\vartheta_i,\varphi_i}$.

mit einem Schwellwert verglichen, da im Allgemeinen lange Strahlen unter flacheren Winkeln auf eine Grenzfläche stoßen, als kurze.

Diese Lösung bringt aber weitere Probleme mit sich für den dritten Fall: Ein Integrationsstrahl geht haarscharf an einer Fläche vorbei, ohne sie jedoch zu durchstoßen. Die Abstandsfunktion eines solchen Strahles weist ein Minimum auf, welches nur geringfügig von Null verschieden ist. Ist ein solcher Integrationsstrahl auch noch recht lang, so erfüllt er das Schwellenwertkriterium und würde als endlich betrachtet. Dies führte dazu, dass die gesamte Dichte hinter dem Minimum nicht berücksichtigt wird.

Eine einfache Lösung ist hier leider nicht möglich, da zur Laufzeit nicht einfach zwischen dem zweiten und dem dritten Fall unterschieden werden kann. Das vorliegende Programm arbeitet daher so, dass, wenn das Schwel-

lenwertkriterium verfehlt wird, die Minimierung mit einem niedrigeren Zielgradienten und einem etwas verschobenen Startpunkt neu gestartet wird. Handelt es sich um einen echten Durchstoßpunkt, der nur zu flach angeschnitten wird, so wird diese Prozedur $D_{\vartheta_i,\varphi_i}$ erniedrigen. Das kann man so lange fortsetzen, bis der Abstand wirklich nahezu Null wird (besser ist es jedoch eine Maximalanzahl an Neustarts vorzugeben). Im Falle eines Minimums, das keinen Durchstoßpunkt darstellt, wird sich $D_{\vartheta_i,\varphi_i}$ nicht auf Null bringen lassen oder sogar ins Unendliche abdriften. Diese Lösung enthält jedoch als fragiles Element die erlaubte Anzahl an Neustarts, denn wer sagt denn, dass nach n-vielen gescheiterten Versuchen, nicht der $(n+1)$-te die erhoffte Lösung bringt?

Der vierte Fall (Integrationsstrahl weit von der Fläche entfernt) ist recht unkompliziert, da die Minimierungsroutine nach wenigen Schritten feststellt, dass es keine Richtung gibt, die $D_{\vartheta_i,\varphi_i}$ minimiert und mit einem entsprechenden Rückgabewert aussteigt.

2.4.3.3 Bifurkationspunkte

Schwierigkeiten bereiten auch Integrationsstrahlen, die die Grenzfläche in der Nähe der kritischen Winkel (siehe Abschnitt 2.3.2, Seite 18) durchstoßen. Trotz des aufwendigen Fittens der Grenzflächen um die Bifurkationswinkel herum (Abschnitt 2.3.3, Seite 22) hat es sich für die Minimierungsroutine als sehr schwierig erwiesen, während eines Minimierungsschrittes den kritischen Winkel zu überschreiten. Also ist auch hier die Wahl des Startwinkels essentiell. Jedoch genügt das nicht.

Um die Chance einer erfolgreichen Minimierung zu erhöhen, werden die Startwerte für θ, die in der Nähe des kritischen Winkels liegen, an diesem gespiegelt und so eine Minimierung von der anderen Seite der Bifurkationszone versucht. Diese Doppelstrategie führt in den meisten Fällen zum Erfolg.

2.4.3.4 Erlaubter Wertebereich

Es kann vorkommen, dass die Gradienten es vorgeben, den gültigen Wertebereich für l oder r zu verlassen. Ein Strahl, der vom Integrationsaufpunkt aus von der Fläche weggeht, hat natürlich einen Durchstoßpunkt bei einem negativen Wert für r, nur ist das nicht die gewünschte Lösung. Und die Chebyshev-Polynome sind auch nur in einem bestimmten Intervall definiert. Während der Minimierung eine kontinuierliche Abfrage auf den Wertebereich durchzuführen und gegebenenfalls den Minimierungsschritt zu verbieten beeinträchtigt jedoch deutlich die Erfolgschancen der Minimierung.

Auf ein negatives r reagiert das vorliegende Programm deshalb erst *nach*

erfolgter Minimierung. In einem solchen Fall wird r auf einen sehr großen Wert gesetzt und die Minimierung erneut gestartet. Man kann davon ausgehen, dass in einem solchen Fall l und θ bereits optimiert sind und r sich also von der anderen Seite der Grenzfläche nähert. Ein weiter entfernter Durchstoßpunkt wird so auch gefunden. Liefert dieser wiederholte Versuch wieder ein negatives r, so gibt es offensichtlich keinen Durchstoßpunkt in dieser Richtung, der Integrationsstrahl ist unendlich.

Um jedoch auf elegante Weise Ergebnisse außerhalb des erlaubten Wertebereichs zu vermeiden, wird ein kleiner Kniff angewendet: verlässt eine Variable ihren erlaubten Bereich, wird zu der Abstandsfunktion $D_{\vartheta_i,\varphi_i}$ eine e-Funktion hinzuaddiert.

$$-10 < r < 0 \quad \Rightarrow \quad \bar{D}_{\vartheta_i,\varphi_i}(l,\theta,r) = D_{\vartheta_i,\varphi_i}(l,\theta,r) + e^{-\alpha r} - 1 \tag{2.44}$$

$$\frac{\partial \bar{D}_{\vartheta_i,\varphi_i}(l,\theta,r)}{\partial r} = \frac{\partial D_{\vartheta_i,\varphi_i}(l,\theta,r)}{\partial r} - \alpha \cdot e^{-\alpha r} \tag{2.45}$$

Um jedoch Registerüberläufe zu vermeiden, wird ab einem bestimmten Wert nur noch eine lineare Funktion addiert.

$$r < -10 \quad \Rightarrow \quad \bar{D}_{\vartheta_i,\varphi_i}(l,\theta,r) = D_{\vartheta_i,\varphi_i}(l,\theta,r) - \alpha \cdot r \tag{2.46}$$

$$\frac{\partial \bar{D}_{\vartheta_i,\varphi_i}(l,\theta,r)}{\partial r} = \frac{\partial D_{\vartheta_i,\varphi_i}(l,\theta,r)}{\partial r} - \alpha \tag{2.47}$$

Der Parameter α gibt dabei an, wie steil die Funktionen mit fallendem r ansteigen.

Da für l Werte außerhalb des Definitionsbereichs $[0 : l_{max}]$ der CHEBYSHEV-Polynome schlicht weg unsinnige Ergebnisse liefern, wird das gleiche Verfahren angewandt, um diesen Fall während der Minimierung schon zu vermeiden.

$$-10 < l < 0 \quad \Rightarrow \quad \bar{D}_{\vartheta_i,\varphi_i}(l,\theta,r) = D_{\vartheta_i,\varphi_i}(l,\theta,r) + e^{-\alpha l} - 1 \tag{2.48}$$

$$\frac{\partial \bar{D}_{\vartheta_i,\varphi_i}(l,\theta,r)}{\partial l} = \frac{\partial D_{\vartheta_i,\varphi_i}(l,\theta,r)}{\partial l} - \alpha \cdot e^{-\alpha l} \tag{2.49}$$

$$l < -10 \quad \Rightarrow \quad \bar{D}_{\vartheta_i,\varphi_i}(l,\theta,r) = D_{\vartheta_i,\varphi_i}(l,\theta,r) - \alpha \cdot l \tag{2.50}$$

$$\frac{\partial \bar{D}_{\vartheta_i,\varphi_i}(l,\theta,r)}{\partial l} = \frac{\partial D_{\vartheta_i,\varphi_i}(l,\theta,r)}{\partial l} - \alpha \tag{2.51}$$

$$l_{max} < l < l_{max} + 10 \Rightarrow \bar{D}_{\vartheta_i,\varphi_i}(l,\theta,r) = D_{\vartheta_i,\varphi_i}(l,\theta,r) + e^{\alpha(l-l_{max})} - 1 \quad (2.52)$$

$$\frac{\partial \bar{D}_{\vartheta_i,\varphi_i}(l,\theta,r)}{\partial l} = \frac{\partial D_{\vartheta_i,\varphi_i}(l,\theta,r)}{\partial l} + \alpha \cdot e^{\alpha(l-l_{max})} \quad (2.53)$$

$$l > l_{max} + 10 \quad \Rightarrow \quad \bar{D}_{\vartheta_i,\varphi_i}(l,\theta,r) = D_{\vartheta_i,\varphi_i}(l,\theta,r) + \alpha \cdot (l - l_{max}) \quad (2.54)$$

$$\frac{\partial \bar{D}_{\vartheta_i,\varphi_i}(l,\theta,r)}{\partial l} = \frac{\partial D_{\vartheta_i,\varphi_i}(l,\theta,r)}{\partial l} + \alpha \quad (2.55)$$

Zu große Werte für α können aber unter Umständen schon bei kleinen Abweichung von r oder l vom erlaubten Wertebereich zu so großen Gradienten führen, dass die Minimierung fehlschlägt. Dies ist erfahrungsgemäß bei $\alpha > 3$ der Fall.

2.4.4 CoreIntersection

Viel einfacher, als den Durchstoßpunkt eines Integrationsstrahls durch eine Grenzfläche zu finden, ist dies bei den kugelförmigen Core-Fits, da dies analytisch möglich ist:

$$l^2 + 2l \cdot \mathbf{R} \cdot \mathbf{e}_{Leb} + (\mathbf{R}^2 - r^2) = 0 \quad (2.56)$$

Die gesuchte Länge des Integrationsstrahls ist die kleinere von (evtl.) zwei reellen Lösungen von Gleichung (2.56) (Herleitung siehe Anhang B, Seite 55). Gibt es dagegen nur negative oder komplexe Lösungen für l, so verfehlt der Integrationsstrahl die Kugel und ist somit unendlich.

2.4.5 Auswertung und Zusammenfassung

Am Ende des Intersection-Laufes stehen für jede Grenzfläche und jeden Core-Fit ein Wert für die Länge des Integrationsstrahls im Speicher. Die gewünschte Länge ist nun der kleinste von allen Werten, denn er repräsentiert den Abstand vom Integrationsaufpunkt bis zur *nächstgelegenen* Grenzfläche. Anschließend wird eindimensional entlang des Integrationsstrahls bis zu der bestimmten Länge nach Clenshaw-Curtis integriert.

Fassen wir also den Ablauf der Subraumintegration noch einmal zusammen, wobei diesmal der Programmablauf im Vordergrund steht:

Der Benutzer initialisiert eine Wellenfunktionsinstanz und lädt die Fits für die benötigten Grenzflächen und Cores. Anschließend setzt er die Integrationsparameter wie LEBEDEV-Ordnung, *fDCT*-Genauigkeit, etc. und legt den Integrationsaufpunkt fest.

In Abhängigkeit von der LEBEDEV-Ordnung werden zunächst zwei LEBEDEV-Tabellen generiert – eine $(\vartheta_i, \varphi_i, w_i)$-Tabelle und eine $(e_{x,i}, e_{y,i}, e_{z,i})$-Tabelle. Erstere wird für die Integration benötigt, letztere für den Startwert θ_{guess}.

Nun werden die Einträge in den LEBEDEV-Tabellen abgearbeitet. Für jedes Winkelpaar (also für jeden Integrationsstrahl) müssen die Durchstoßpunkte *aller* geladenen Grenzflächen und aller CoreFits bestimmt werden.

Für jede Grenzfläche wird als erstes θ_{guess} bestimmt und damit eine Gridsuche für Startwerte von l und r ausgeführt. Mit diesen Werten wird die POWELL-Minimierung gestartet. Bringt die Minimierung ein Ergebnis, bei dem die Abstandsfunktion $D_{\vartheta_i,\varphi_i} = 0$ wird und $r > 0$ ist, so wird der Wert gespeichert und die nächste Fläche wird bearbeitet.

Ist das nicht der Fall, werden die oben besprochenen Spezialfälle abgearbeitet. Bei einem *Distanzfehler* ($D_{\vartheta_i,\varphi_i} \neq 0$) wird bei minimal versetzten Startwerten und kleinerem Zielgradienten die Minimierung neu gestartet. Bringt das eine Optimierung von $D_{\vartheta_i,\varphi_i}$, gilt die Fläche als getroffen.

Wenn es zu einem Ergebnis $r < 0$ kommt, wird die Minimierung mit $r = \infty$ (bzw. einem Maximalwert für r) neu gestartet. War θ_{guess} in der Nähe einer Bifurkationszone, wird eine weitere Minimierung mit gespiegeltem θ versucht.

Ist bis jetzt keine erfolgreiche Minimierung möglich gewesen, wird der Integrationsstrahl als unendlich angesehen und zur nächsten Grenzfläche gewechselt.

Sind alle Flächen abgearbeitet, werden in einer weiteren Schleife die Durchstoßpunkte mit den CoreFits berechnet.

Sind alle Durchstoßpunkte ermittelt, findet die Auswertung und die 1D-Integration statt. Das Ergebnis wird, multipliziert mit dem LEBEDEV-Gewicht w_i, in einem Register aufaddiert. Danach geht es mit dem nächsten Eintrag in der LEBEDEV-Tabelle weiter, bis alle Winkelpaare bearbeitet wurden.

Kapitel 3

Anwendungsbeispiele

3.1 Technische Details der Rechnungen

Zur Erzeugung der bei der Berechnung von *ELF* auf *DFT*-Niveau benötigten Einteilchendichtematrizen wurde das Programm GAUSSIAN98 [25] verwendet, im Speziellen der `wfn`-Exportfilter. Alle Moleküle wurden in *aug-cc-pVTZ*-Basis mit dem *B3LYP*-Funktional geometrieoptimiert und dann auf dem selben Niveau die Dichtematrix in Form von *MOs* exportiert. Die erhaltenen Geometrien sind in Tabelle 3.1 aufgeführt.

Alle weiteren Rechnungen wurden mit XELF2 ausgeführt. Als erstes wurden die *BCPs* und die *ECPs* gesucht und die entsprechenden Grenzflächen konstruiert. Es wurden für jede Grenzfläche 256 Winkel mit jeweils 2500 Schritten der Länge 0.01 bohr erzeugt. Diese wurden dann gefittet. Es wurden immer mindestens 32 CHEBYSHEV-Koeffizienten berücksichtigt, aus den 256 Winkeln wurden entsprechend $2 \cdot 128$ *FFT*-Koeffizienten (128 Cosinus- und 128 Sinuskoeffizienten). Zum Schluß wurden noch die Radien der sphä-

Tabelle 3.1: Optimierte Geometrien der berechneten Moleküle.

Molekül	$d(\mathrm{A}-\mathrm{H})$	$a(\mathrm{HAH})$
H_2O	0.9619 Å	105.00°
NH_3	1.0132 Å	107.23°
CH_4	1.0883 Å	109.48°
H_2S	1.3447 Å	92.54°
PH_3	1.4209 Å	93.49°
SiH4	1.4823 Å	109.48°

rischen Core-Fits bestimmt.

Mit all diesen Daten konnte dann die eigentliche Integration der Bassins erfolgen. Alle Integrationen erfolgten mit LEBEDEV11-Grad (974 Integrationswinkel). Für die CLENSHAW-CURTIS-Quadratur wurden jeweils 512 Koeffizienten für die *fDCT* verwendet, mit einem erlaubten Maximalfehler von 10^{-20}. Bei den *AIM*-Integrationen wurden die Atomkerne als Integrationsaufpunkte verwendet. Diese bilden ebenfalls die Integrationsaufpunkte für die Berechnung der *ELF*-Populationen der Core- sowie der Wasserstoffbassins. Die Integrationsaufpunkte der Bassins der einsamen Elektronenpaare lagen auf den molekularen Symmetrieachsen in möglichst geringem Abstand (0.2 bis 0.3 bohr) von der nächstgelegenen Separatrix des Cores. Bei den Moleküle H_2O und H_2S, bei denen es eigentlich jeweils zwei Lonepairbassins gibt, wurden beide gemeinsam integriert. Die V(A,H)|H-Subbassins wurden ebenfalls von den Wasserstoffkernen aus integriert, während die Aufpunkte der V(A,H)|A-Subbassins auf der Mitte zwischen den entsprechenden *BCPs* und der Coreseparatrix lagen.

3.2 Populationsanalyse - Die Roh-Ergebnisse

Die in dieser Arbeit interessierenden Größen sind zunächst einmal die mittleren Elektronenpopulationen der verschiedenen Bassins $\bar{N}_s$. Diese wurden in den folgenden Tabellen sowohl für die totalen *AIM*-Bassins und die totalen *ELF*-Bassins als auch für die durch *AIM*-Separatrizes unterteilten *ELF*-Subbassins angegeben. Im letzten Fall werden ebenfalls die Bindungspolaritäten p_{XH} (Gleichung (1.10), Seite 12) aufgeführt.

3.2.1 Das Wasser

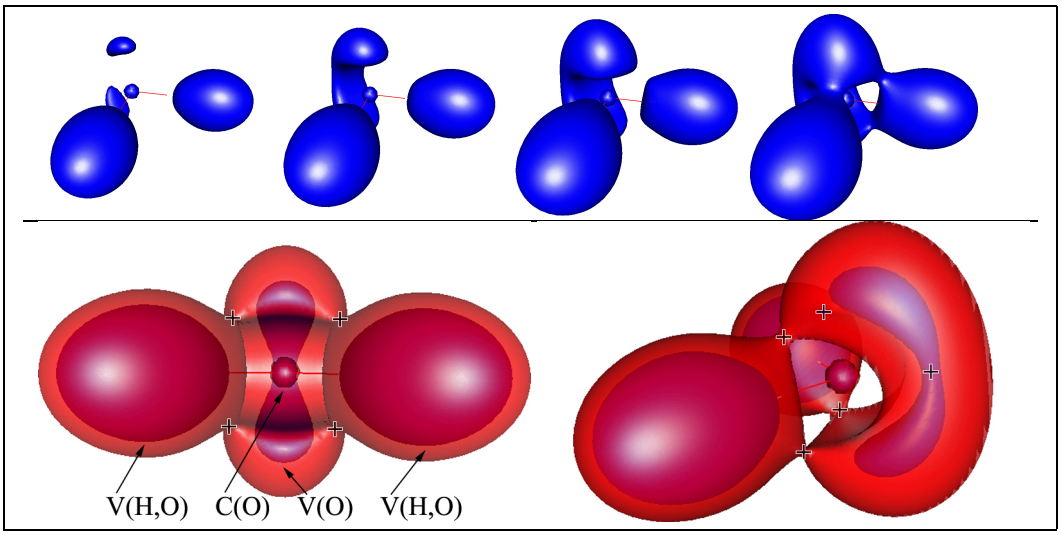


Abbildung 3.1: *Oben: ELF*-Isoflächen von H_20 ($\eta = 0.95 \ldots 0.8$ von links nach rechts). *Unten:* Lage der *ECPs*.

Tabelle 3.2: Ergebnisse der Populationsanalyse von H_2O

AIM	O1	H2	H3	$\sum$
N	9.14	0.43	0.43	10.00
q	-1.14	$+0.57$	$+0.57$	± 0.00

ELF	V(O1,H2)	V(O1,H3)	V(O1)	C(O1)	$\sum$
N	1.67	1.66	4.67	2.13	10.13

ELF/AIM	V(O1,H2)		V(O1,H3)		$\sum$
	O1	H2	O1	H3	
N	1.23	0.43	1.22	0.43	10.11
$\sum$	1.66		1.65		
p_{OH}	$+0.48$		$+0.48$		

3.2.2 Ammoniak

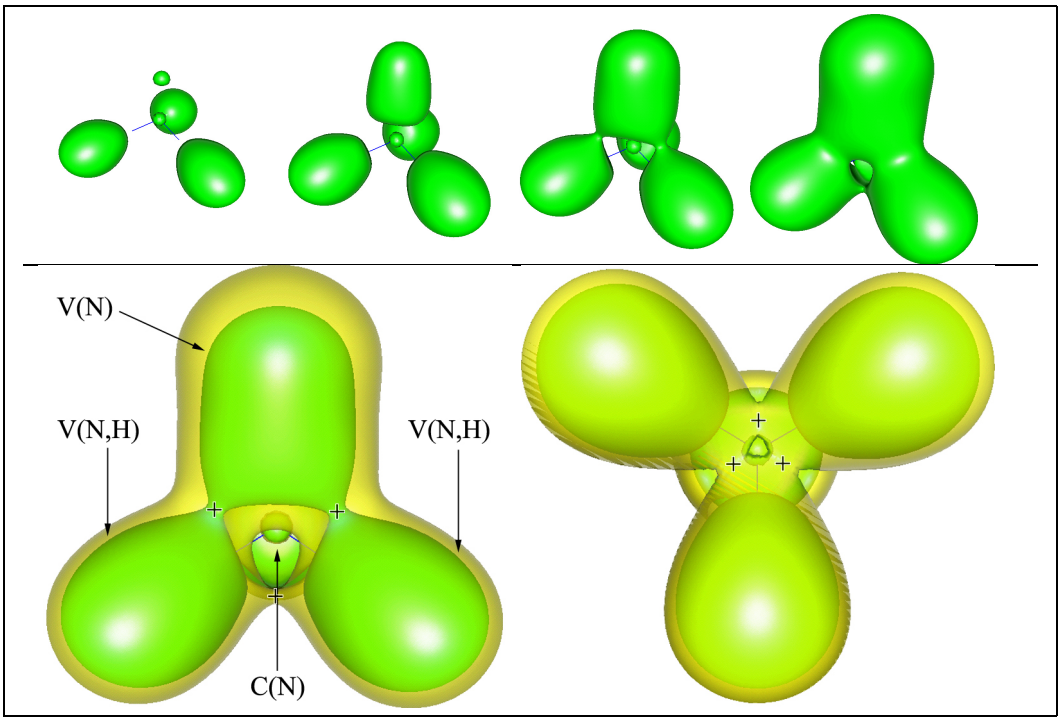


Abbildung 3.2: *Oben: ELF*-Isoflächen von NH_3 ($\eta = 0.97 \ldots 0.73$ von links nach rechts). *Unten:* Lage der *ECPs*.

Tabelle 3.3: Ergebnisse der Populationsanalyse von NH_3

AIM	N1	H2	H3	H4	$\sum$
$\bar{N}$	8.00	0.66	0.66	0.66	9.98
q	−1.00	+0.34	+0.34	+0.34	+0.02

ELF	V(N1,H2)	V(N1,H2)	V(N1,H2)	V(N1)	C(N1)	$\sum$
$\bar{N}$	1.92	1.91	1.93	2.11	2.12	9.99

ELF/	V(N1,H2)		V(N1,H3)		V(N1,H4)		$\sum$
AIM	N1	H2	N1	H3	N1	H4	
$\bar{N}$	1.26	0.66	1.25	0.66	1.27	0.66	9.99
$\sum$	1.92		1.91		1.93		
p_{NH}	+0.31		+0.31		+0.32		

3.2.3 Methan

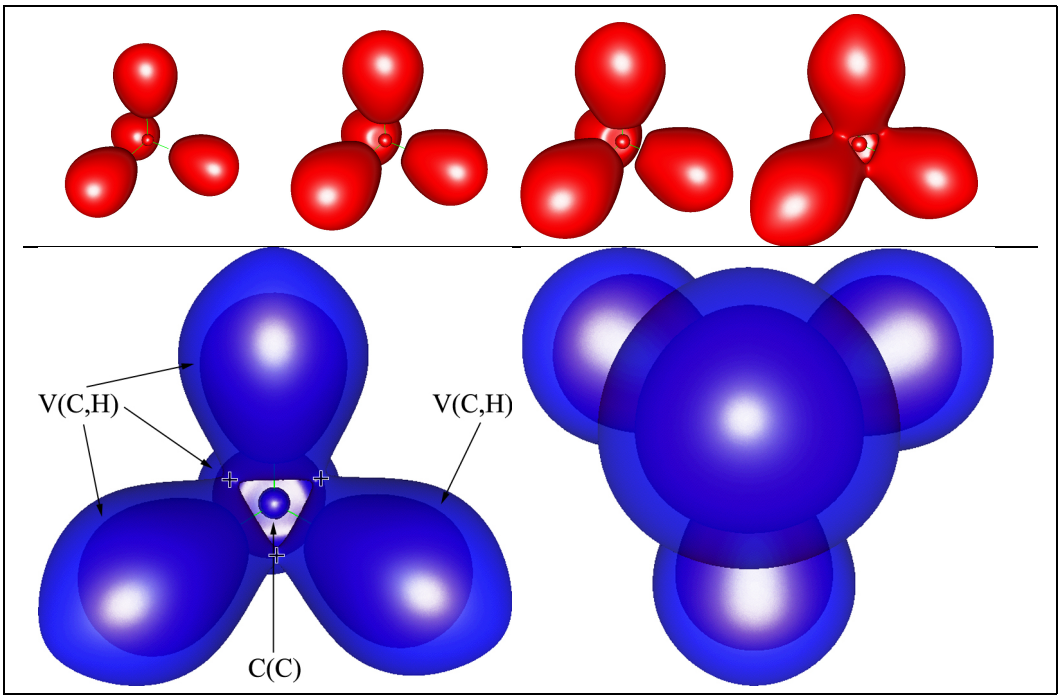


Abbildung 3.3: *Oben: ELF*-Isoflächen von CH_4 ($\eta = 0.9 \ldots 0.4$ von links nach rechts). *Unten:* Lage der *ECPs*.

Tabelle 3.4: Ergebnisse der Populationsanalyse von CH_4

AIM	C1	H2	H3	H4	H5	$\sum$
N	5.97	1.01	1.01	1.01	1.01	10.01
q	+0.03	−0.01	−0.01	−0.01	−0.01	−0.01

ELF	V(C1,H2)	V(C1,H3)	V(C1,H4)	V(C1,H5)	C(C1)	$\sum$
N	1.97	1.96	1.98	1.96	2.10	9.97

ELF/	V(C1,H2)		V(C1,H3)		V(C1,H4)		V(C1,H5)		$\sum$
AIM	C1	H2	C1	H3	C1	H4	C1	H5	
N	1.09	1.01	1.04	1.01	1.00	1.01	1.09	1.01	10.36
$\sum$	2.10		2.05		2.01		2.10		
p_{CH}	−0.04		−0.01		±0.00		−0.04		

3.2.4 Schwefelwasserstoff

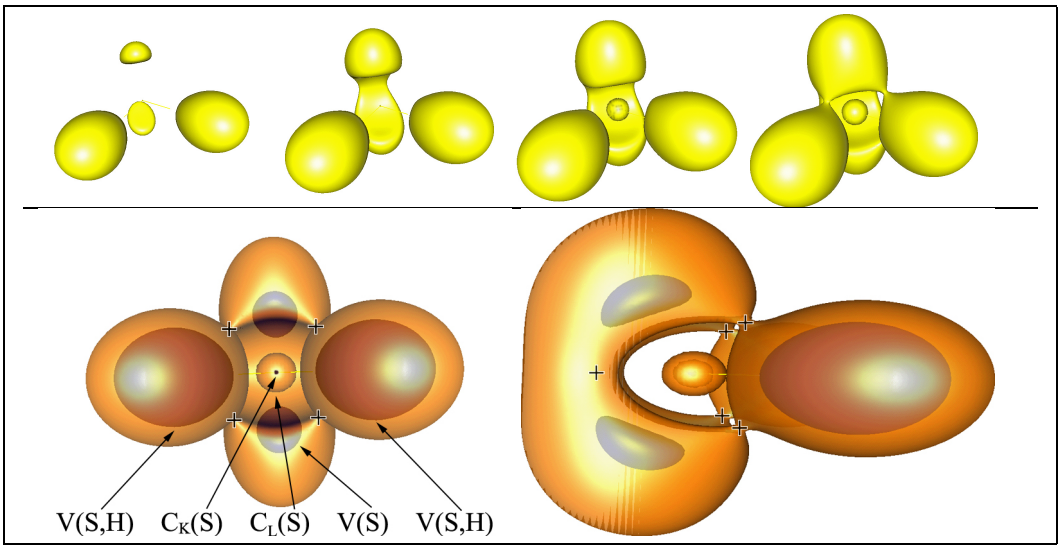


Abbildung 3.4: *Oben: ELF*-Isoflächen von H_2S ($\eta = 0.91 \ldots 0.8$ von links nach rechts). *Unten:* Lage der *ECPs*.

Tabelle 3.5: Ergebnisse der Populationsanalyse von H_2S

AIM	S1	H2	H3	$\sum$
$\bar{N}$	15.91	1.06	1.06	18.03
q	+0.09	−0.06	−0.06	−0.03

ELF	V(S1,H2)	V(S1,H3)	V(S1)	C_K(S1)	C_L(S1)	$\sum$
N	1.96	1.93	4.01	2.20	7.86	17.96

ELF/	V(S1,H2)		V(S1,H3)		V(S1)			$\sum$
AIM	S1	H2	S1	H3	S1	H2	H3	
$\bar{N}$	0.93	1.04	0.91	1.04	3.97	0.02	0.02	17.99
$\sum$	1.97		1.96			4.01		
p_{SH}	−0.06		−0.07			—		

3.2.5 Phosphan

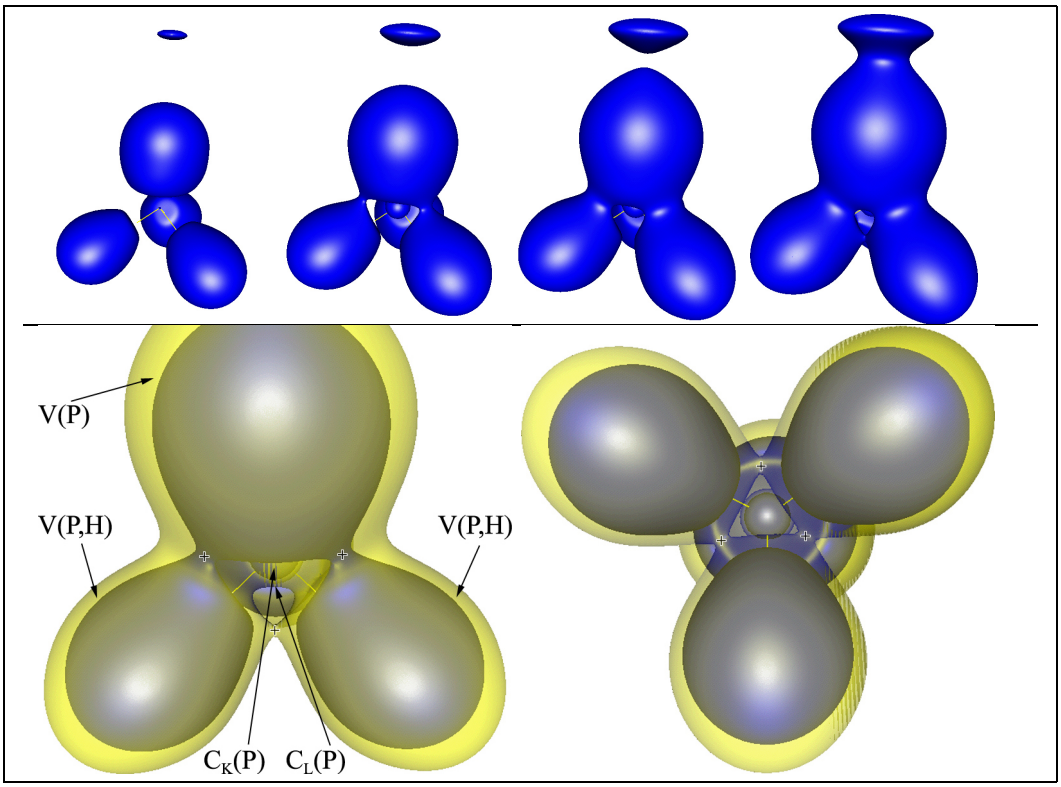


Abbildung 3.5: *Oben: ELF*-Isoflächen von PH_3 ($\eta = 0.93 \ldots 0.75$ von links nach rechts). *Unten:* Lage der *ECPs*.

Tabelle 3.6: Ergebnisse der Populationsanalyse von PH_3

AIM	P1	H2	H3	H4	$\sum$
$\bar{N}$	13.45	1.50	1.50	1.50	17.95
q	+1.55	−0.50	−0.50	−0.50	+0.05

ELF	V(P1,H2)	V(P1,H3)	V(P1,H4)	V(P1)	C_K(P1)	C_L(P1)	$\sum$
N	2.02	1.99	1.98	2.00	2.20	7.87	18.06

ELF/	V(P1,H2)		V(P1,H3)		V(P1,H4)		$\sum$
AIM	P1	H2	P1	H3	P1	H4	
N	0.52	1.50	0.64	1.50	0.62	1.50	18.35
$\sum$	2.02		2.14		2.12		
p_{PH}	−0.49		−0.41		−0.41		

3.2.6 Silan

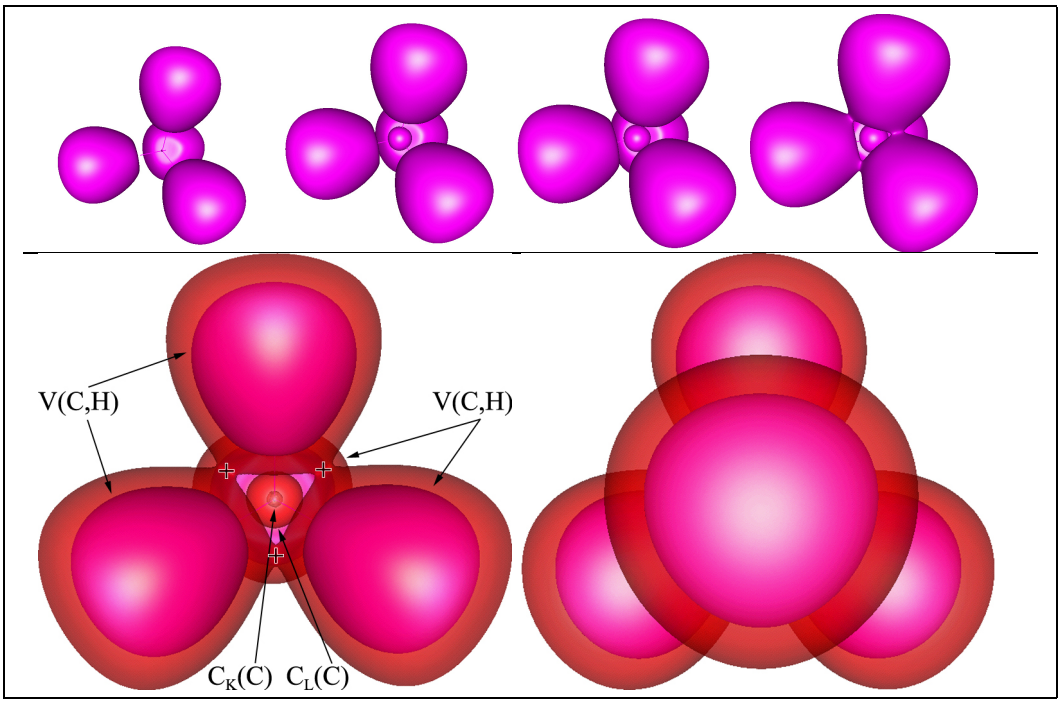


Abbildung 3.6: *Oben: ELF*-Isoflächen von SiH_4 ($\eta = 0.9 \ldots 0.42$ von links nach rechts). *Unten:* Lage der *ECPs*.

Tabelle 3.7: Ergebnisse der Populationsanalyse von SiH_4

AIM	Si1	H2	H3	H4	H5	$\sum$
$\bar{N}$	11.28	1.67	1.67	1.67	1.67	17.96
q	+2.72	−0.67	−0.67	−0.67	−0.67	+0.04

ELF	V(Si1,H2)	V(Si1,H3)	V(Si1,H4)	V(Si1,H5)	C_K(Si1)	C_L(Si1)	$\sum$
N	2.06	1.98	2.01	1.98	2.19	7.88	18.10

ELF/	V(Si1,H2)		V(Si1,H3)		V(Si1,H4)		V(Si1,H5)		$\sum$
AIM	Si1	H2	Si1	H2	Si1	H4	Si1	H5	
$\bar{N}$	0.38	1.67	0.32	1.67	0.33	1.67	0.33	1.67	18.11
$\sum$	2.05		1.99		2.00		2.00		
p_{SiH}	−0.63		−0.67		−0.67		−0.67		

3.3 Topologische Besonderheiten

Man kann anhand der *ELF*-Isosurfaces der einzelnen Moleküle einige interessante Eigenschaften beschreiben.

Beim H_2O gibt es zwei Arten von *ECPs*: Vier (aus Symmetriegründen) äquivalente Sattelpunkte, die an den Verbindungsstellen zwischen V(O,H) und V(O) liegen und einen einzelnen, der V(O) noch einmal in zwei Subbassins unterteilt. Die äquivalenten *ECPs* liegen jeweils auf gleichen *ELF*-Niveaux. Die Isofläche $\eta = 0.9$ stellt das Wasser genau so dar, wie man es im Lewis-Elektronenpaarmodell erwarten würde: zwei Bindungsbereiche zu den Wasserstoffatomen, einen Rumpfbereich um das Kohlenstoffatom herum sowie zwei einsame Elektronenpaare.

Auch beim NH_3 gibt es zwei Arten von *ECPs*. Die ersten drei äquivalenten Punkte liegen an den Verbindungsstellen zwischen V(N,H) und V(N), die anderen drei äquivalenten *ECPs* an denen zwischen den V(N,H)-Bassins.

Das CH_4 besitzt (aus Symmetriegründen) nur eine Sorte von *ECPs* an den Verbindungsstellen der diversen V(C,H)-Bassins.

Diese Muster wiederholen sich bei den jeweils ähnlichen Verbindungen der dritten Periode: also H_2S wie H_2O, PH_3 wie NH_3 und SiH_4 wie CH_4.

Beim PH_3 kommt noch eine Besonderheit hinzu. Das Lonpair des P besitzt eine Art „Mütze“. Das scheint ein Artefakt des Basissatzes zu sein. Mit viel Mühe findet man sogar einen *ECP* zwischen dem Lonepair und der „Mütze“.

3.4 Interpretationen der Populationsanalyse

Im folgenden sollen die oben vorgestellten Roh-Ergebnisse der Populationsanalyse interpretiert und diskutiert werden.

3.4.1 Die Rache der Numerik

Idealerweise sollten die Ergebnisse für äquivalente Bassins im selben Molekül auch vom Wert her identisch sein. Wie man auf den vorhergegangenen Seiten jedoch sehen konnte, ist dies nur bei der reinen *AIM*-Integration in guter Näherung (bis auf drei Nachkommastellen) gegeben, während dies für die *ELF*- und die kombinierte *ELF/AIM*-Integration meist (leider) nicht der Fall ist. Nach sehr sorgfältiger Prüfung ist jedoch ein Fehler im Programm selber so gut wie ausgeschlossen. Vielmehr sind numerische Ursachen für die verbleibenden *Inäquivalenzen* anzuführen.

Die Schwierigkeiten, die im Intersection-Algorithmus auftreten, sind an anderer Stelle ja schon ausführlich behandelt werden (Abschnitt 2.4.3, Seite 31). Trotz ausgefeilter Algorithmik bleibt eine kleine Ungewissheit bei der Bestimmung der Durchstoßpunkte. Und diese Unbestimmtheit führt bei der reinen *ELF-Integration* zu Schwankungen in den Ergebnissen. Je weniger LEBEDEV-Punkte man verwendet, desto gewichtiger werden fehlerhafte Intersection-Ergebnisse. Man kann also mit steigender LEBEDEV-Ordnung davon ausgehen, dass die Ergebnisse immer sicherer werden. Eine weitere Ursache für numerische Differenzen in den Populationen von symmetrieäquivalenten Atomen und Bindungen ist, dass in der aktuellen Programmversion die Koordinatensysteme aller LEBEDEV-Integrationsstrahlen parallel liegen. Solche parallelen Bassin-Koordinatensysteme lassen sich aber im Allgemeinen *nicht* durch Symmetrietransformation ineinander überführen. Eine gleichmäßigere Ausfüllung des Raums, durch Verwendung höherer Ordnungen der LEBEDEV-Integration, sollte dieses Problem ebenfalls vermindern. Einige Testrechnungen bestätigen dann auch die Vermutung, dass sich für immer mehr LEBEDEV-Punkte die Ergebnisse „einpendeln“.

Eine weitere Subtilität der LEBEDEV-Integration darf nicht verschwiegen werden. Die Quadratur ist am genauesten, wenn man entlang der Integrationsstrahlen von einem Gebiet hoher Dichte am Integrationsaufpunkt in ein Gebiet geringer Dichte integriert. Ist dies nicht der Fall, kann es sein, dass ein Integrationsstrahl bei größeren Abständen mehr oder weniger zufällig durch ein kleines Gebiet hoher Dichte geht. Die Punkte mit den selben Abständen auf benachbarten Integrationsstrahlen sind allerdings bei größeren Abständen schon recht weit voneinander entfernt. Da die Dichte zwischen diesen Strahlen gleichsam interpoliert wird (durch die Gewichtsfunktion) kann man sich so eine mehr oder weniger zufällige Überschätzung der Dichte einhandeln. Leider ist es bei den Aufpunkten der *ELF*- und speziell der kombinierten *ELF/AIM*-Integration oft so, dass der Idealfall einer Integration von Gebieten hoher in Gebiete niedriger Elektronendichte, nicht immer gegeben ist – es tritt also genau der *worst case* ein.

Im folgenden werden die Ergebnisse dehalb nicht blind weiter verwendet. Vielmehr wird das Ergebnis der kombinierten *ELF/AIM*-Integration, das am besten mit den reinen *AIM*- und *ELF*-Integrationen übereinstimmt, als am meisten realistisches erachtet. Davon ausgehend finden alle folgenden Interpretationen statt.

3.4.2 Gegenüberstellungen

Um ein Gefühl für die Polarität einer Bindung zu bekommen, zieht man bisweilen die *Elektronegativität* der Bindungspartner heran. Große Differenzen

der Elektronegativität sollten stark polarisierte Bindungen hervorbringen.

Tabelle 3.8 zeigt die Elektronegativität nach ALLRED-ROCHOW [26] für die hier verwendeten Elemente. Tabelle 3.9 stellt nun die Differenz der Elektronegativitäten den aus der Populationsanalyse errechneten Bindungspolaritäten gegenüber. Ferner werden Bindungslänge und Bindungsgrad (Gleichung (1.9), Seite 11) angegeben.

Wie man sieht, korrelieren die Trends von ΔEN und p_{AH} sehr schön miteinander – mit einer Ausnahme: Bromwasserstoff. Die Gründe hierfür können jedoch in der Tatsache liegen, dass alle Rechnungen nicht-relativistisch durchgeführt wurden. Es kann sein, dass dies für Brom schon nicht mehr zulässig ist.

Tabelle 3.8: Elektronegativitäten nach ALLRED-ROCHOW

H	C	N	O	F
2.2	2.5	3.1	3.5	4.1
	Si	P	S	Cl
	1.7	2.1	2.4	2.8
				Br
				2.7

Tabelle 3.9: Vergleiche der A − H-Bindungen, sortiert nach aufsteigender Polarisation der Bindung.

Bindung	r_{AH}	b_{AH}	ΔEN	p_{AH}
Si − H	1.482 Å	1.0	−0.5	−0.67
P − H	1.421 Å	1.0	−0.1	−0.49
S − H	1.345 Å	1.0	+0.2	−0.07
Br − H[a]	1.425 Å	0.9	+0.5	−0.04
C − H	1.088 Å	1.0	+0.3	±0.00
Cl − H[a]	1.283 Å	0.9	+0.4	+0.14
N − H	1.013 Å	1.0	+0.9	+0.31
O − H	0.962 Å	0.8	+1.3	+0.48
F − H[a]	0.924 Å	0.7	+1.9	+0.62

[a] Referenz [14]

3.4.3 Durchdringung von Grenzflächen

Bei fast allen hier vorgestellten Molekülen ist auffällig, dass der Wert bei der kombinierten *ELF/AIM*-Integration für die V(A,H)|H-Subbassins exakt dem Wert für die Wasserstoffbassins der reinen *AIM*-Integration entsprechen – mit der Ausnahme H_2S.

Der Grund für diese Übereinstimmung liegt darin, dass die atomaren Grenzflächen und die Lokalisationsgrenzflächen sich nicht durchdringen bzw. erst soweit von den Regionen hoher Dichte entfernt durchdringen, dass das bei der Integration kaum noch eine Rolle spielt. Vom Integrationsaufpunkt (dem Wasserstoff) aus gesehen ist also auch bei der kombinierten *ELF/AIM*-Integration die atomare Separatrix *cum grano salis* die nächstgelegene Grenzfläche. Deshalb stimmen die Werte überein.

Das ist aber keineswegs immer so, wie das H_2S zeigt. Hier durchdringen sich die atomaren und die Lokalisationsgrenzflächen schon relativ früh. Da die *ELF*-Grenzfläche also ein Stückchen in das atomare Bassin des Wasserstoffs hineinragt, ist der Wert der kombinierten *ELF/AIM*-Integration kleiner als der der reinen *AIM*-Integration. Auf der anderen Seite kommt es so zu dem Phänomen, dass ein Teil der einsamen Elektronenpaare am Schwefel zur atomaren Domäne des Wasserstoffs gehört.

Anhang A

Herleitung von ELF

Die HARTREE-FOCK-Wahrscheinlichkeit, zwei Elektronen mit dem selben Spin σ an den Positionen **1** und **2** zu finden ist

$$P_2^{\sigma\sigma}(\mathbf{1},\mathbf{2}) = \rho_\sigma(\mathbf{1})\rho_\sigma(\mathbf{2}) - |\rho_1^\sigma(\mathbf{1},\mathbf{2})|^2. \tag{A.1}$$

Dabei ist $\rho_1^\sigma(\mathbf{1},\mathbf{2})$ die HARTREE-FOCK Einteilchendichtematrix.

$$\rho_1^\sigma(\mathbf{1},\mathbf{2}) = \sum_i^\sigma \psi_i^*(\mathbf{1})\psi_i(\mathbf{2}) \tag{A.2}$$

Es wird nur über Orbitale mit σ-Spin summiert. Wenn ein σ-Spin-Elektron mit Sicherheit an Position **1** ist, dann ist die Wahrscheinlichkeit, ein zweites σ-Spin-Elektron an Position **2** zu finden

$$\begin{aligned} P_{cond}^{\sigma\sigma}(\mathbf{1},\mathbf{2}) &= \frac{P_2^{\sigma\sigma}(\mathbf{1},\mathbf{2})}{\rho_\sigma(\mathbf{1})} \\ &= \rho_\sigma(\mathbf{2}) - \frac{|\rho_1^\sigma(\mathbf{1},\mathbf{2})|^2}{\rho_\sigma(\mathbf{1})}. \\ \rho_\sigma(\mathbf{1}) &: \text{totale } \sigma\text{-Spindichte bei } \mathbf{1} \end{aligned} \tag{A.3}$$

$P_{cond}^{\sigma\sigma}(\mathbf{1},\mathbf{2})$ nennt man *bedingte Paarwahrscheinlichkeit.*

Unter Ausnutzung der bekannten Zusammenhänge der Einteilchendichtematrix für den Fall, dass beide Koordinaten identisch sind

$$\rho_1^\sigma(\mathbf{1},\mathbf{1}) = \rho_\sigma(\mathbf{1}) \tag{A.4}$$

und dem Integral des Betragsquadrates integriert über die Koordinate **2**

$$\int |\rho_1^\sigma(\mathbf{1},\mathbf{2})|^2 \, d\mathbf{2} = \rho_\sigma(\mathbf{1}) \tag{A.5}$$

kann gezeigt werden, dass die bedingte Paarwahrscheinlichkeit am Referenzpunkt **1** ein weiteres Teilchen mit σ-Spin zu finden gerade Null ist:

$$P^{\sigma\sigma}_{cond}(\mathbf{1},\mathbf{1}) = 0. \tag{A.6}$$

Man spricht hier auch von einem FERMI-*Loch*.

Nun kann die bedingte Paarwahrscheinlichkeit $P^{\sigma\sigma}_{cond}$ durch eine TAYLOR-Reihe, sphärisch gemittelt um einen Mittelpunkt $\mathbf{r}$ entwickelt werden:

$$P^{\sigma\sigma}_{cond}(\mathbf{r},s) = \frac{1}{3}\left[\tau_0 - \frac{1}{4}\frac{(\nabla\rho_\sigma)^2}{\rho_\sigma}\right]s^2 + ... \tag{A.7}$$

$$\begin{aligned} \mathbf{r} &= \text{Kreismittelpunkt} \\ s &= \text{Radius} \\ \tau_0 &= \sum_i^\sigma |\nabla\phi_i|^2 \qquad (A.8) \\ &= \text{kinetische Energie} \end{aligned}$$

Je geringer die Wahrscheinlichkeit, ein spin-gleiches Elektron am Referenzpunkt zu finden, desto mehr ist das Referenzelektron lokalisiert. Es kann sich dort also höchstens noch ein zweites Teilchen mit entgegengesetztem Spin befinden. Da im geschlossenschaligen Fall alle bisherigen Überlegungen genau so für ein Elektron mit entgegengesetztem Spin gelten, muss sich hier also in diesem Fall ein *Elektronenpaar* befinden.

Die Elektronenlokalisation hängt also ab von

$$D_\sigma = \tau_0 - \frac{1}{4}\frac{(\nabla\rho_\sigma)^2}{\rho_\sigma}, \tag{A.9}$$

welches durch die Tatsache, dass Gleichung (A.7) eine Wahrscheinlichkeitsdichte darstellt, notwendigerweise positiv ist. D_σ verschwindet für den Spezialfall, dass in Multielektronensystemen regionale Domänen von einem einzelnen, lokalisierten σ-Spinorbital beherrscht werden. Auch das spricht für die Tatsache, dass D_σ ein vernünftiges Maß für die Elektronenpaarlokalisation ist.

D_σ verhält sich jedoch reziprok, d.h. für eine hohe Elektronenlokalisation wird D_σ klein und ein großes D_σ zeigt eine geringe Elektronenlokalisation an. Deshalb wird in [2] die Elektronenlokalisationsfunktion (*ELF*) definiert zu

$$\eta = (1 + \chi^2_\sigma)^{-1} \tag{A.10}$$

mit

$$\chi_\sigma = \frac{D_\sigma}{D_\sigma^0} \tag{A.11}$$

dem *ELF*-Core und

$$D_\sigma^0 = \frac{3}{5}(6\pi^2)^{2/3}\rho_\sigma^{5/3} \tag{A.12}$$

dem D_σ-Wert für ein homogenes Elektronengas als Normierungsfaktor. So ist der *ELF*-Core χ_σ eine dimensionslose Größe mit dem Wert D_σ als Referenz. Die *ELF* besitzt damit einen Wertebereich von

$$0 \leq \eta \leq 1. \tag{A.13}$$

Die obere Grenze $\eta(\mathbf{r}) = 1$ steht für eine perfekte Lokalisation und $\eta(\mathbf{r}) = \frac{1}{2}$ für eine elektronengasartige Struktur.

Es gibt noch weitere Interpretationen der Elektronenlokalisationsfunktion im Rahmen der Dichtefunktionaltheorie von Kohn und Sham [27]. In der Dichtefunktionaltheorie erscheint nicht die kinetische Energie, sondern die kinetische Energie eines hypothetischen Systems mit derselben Dichte wie das exakte System, in dem aber die Teilchen zwar ebenfalls Fermionen, aber wechselwirkungsfrei sind.

Die Größe D_σ gibt dann den Überschuss an kinetischer Energie eines Fermions auf Grund des Pauli-Prinzips an: Fermionen mit gleichem Spin können sich wegen der Antisymmetrieforderung nicht beliebig nahe kommen, was anschaulich gesprochen zu einer Ausweichbewegung mit entsprechendem Anwachsen der kinetischen Energie führen muss. Die Konsequenz ist letztlich dieselbe, wie bei den Überlegungen, die auf der Hartree-Fock-Theorie beruhen: Orte, an denen D_σ klein ist, zeigen hohe Lokalisation eines Teilchens mit gegebenem Spin an; hier kann sich dann höchstens noch ein Teilchen entgegengesetzten Spins aufhalten.

Anhang B

Herleitung des CoreIntersection-Algorithmus

Jeder Punkt auf der Oberfläche einer Kugel mit dem Radius r und dem Ortsvektor des Mittelpunkts $\mathbf{c}$ kann dargestellt werden als

$$\mathbf{x}_r = \mathbf{c} + r \cdot \begin{pmatrix} \sin(\vartheta)\cos(\varphi) \\ \sin(\vartheta)\sin(\varphi) \\ \cos(\vartheta) \end{pmatrix}. \tag{B.1}$$

Sei $\mathbf{L}$ der Ortsvektor zum Integrationsaufpunkt (Abb. B.1), so gilt für den Vektor $\mathbf{R}$ vom Kugelmittelpunkt zum Integrationsaufpunkt

$$\mathbf{R} = \mathbf{L} - \mathbf{c}. \tag{B.2}$$

Jeder Punkt auf dem Integrationsstrahl in Richtung des LEBEDEV-Einheitsvektors $\mathbf{e}_{Leb}$ ist erreichbar durch

$$\mathbf{x}(l) = \mathbf{c} + \mathbf{R} + l \cdot \mathbf{e}_{Leb}. \tag{B.3}$$

Ein Integrationsstrahl durchstößt die Kugeloberfläche also dann, wenn die Gleichung

$$\mathbf{x}_r = \mathbf{x}(l) \tag{B.4}$$

eine positive, reelle Lösung für l besitzt.

Damit suchen wir eine Lösung der Gleichung

$$\mathbf{R} + l \cdot \mathbf{e}_{Leb} = r \cdot \begin{pmatrix} \sin(\vartheta)\cos(\varphi) \\ \sin(\vartheta)\sin(\varphi) \\ \cos(\vartheta) \end{pmatrix} \tag{B.5}$$

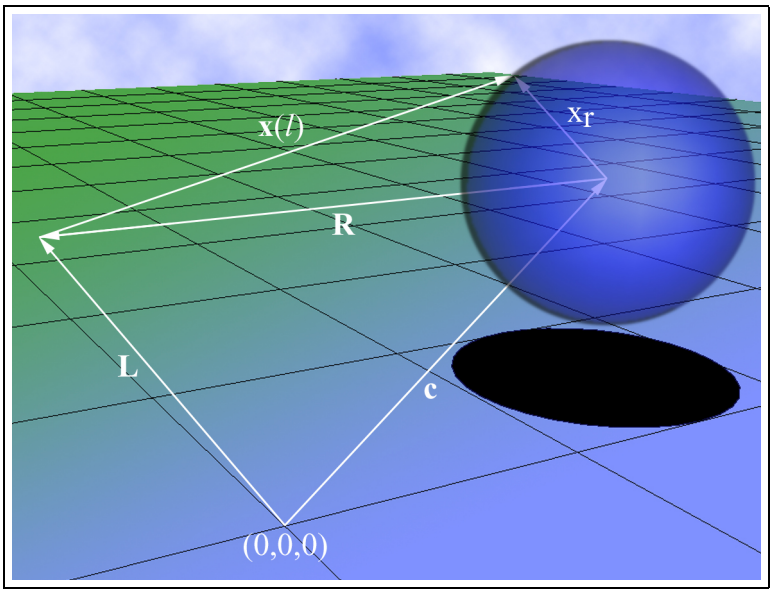


Abbildung B.1: Schema zum CoreIntersection-Algorithmus

was quadriert

$$(\mathbf{R} + l \cdot \mathbf{e}_{Leb})^2 = r^2 \cdot 1 \tag{B.6}$$

ergibt. Somit findet man den Wert von l aus

$$l^2 + 2l\mathbf{R}\mathbf{e}_{Leb} + (\mathbf{R}^2 - r^2) = 0, \tag{B.7}$$

wobei nur eine Lösung dieser Gleichung sinnvoll ist.

Anhang C

XElf2 Language Reference

C.1 Parameter Setting

Für die Parameter gilt folgende Nomenklatur:

^{i}x : x ist vom Typ *integer*
^{d}x : x ist vom Typ *double precision*
^{s}x : x ist vom Typ *string*
^{v}x : x ist vom Typ *vector* entspricht $^{d}x_1$ $^{d}x_2$ $^{d}x_3$

C.1.1 Fitting Settings

`MaxCC` $^{i}MaxCC$	Setzt die Fittingvariable $MaxCC$ auf den Wert $^{i}MaxCC$. Sie gibt die Anzahl an CHEBYSHEV-Koeffizienten an, die auf jeden Fall berücksichtigt werden.
`MaxCError` $^{d}MaxCErr$	Setzt die Fittingvariable $MaxCErr$ auf den Wert $^{d}MaxCErr$. Diese Variable bestimmt den maximalen Fehler, der während der Bestimmung der CHEBYSHEV-Koeffizienten erlaubt ist. *Default*: $MaxCErr = 13^{-3}$.

C.1.2 Integration Settings

`CCAcc` $^{d}CCAcc$	Setzt die Integrationsvariable $CCAcc$ auf den Wert $^{d}CCAcc$. Dieser Wert gibt die Zielgenauigkeit für die DCT während der Clenshaw-Curtis-Quadratur an. *Default*: $CCAcc = 10^{-16}$.
`Lebedev` $^{i}Lebedev$	Setzt die Integrationsvariable $Lebedev$ auf den Wert $^{i}Lebedev$. Gibt die algebraische Ordnung für die Lebedev-Quadratur an und damit die Anzahl der Lebedev-Punkte.

Lebedev	Punkte
1	14
2	26
3	50
4	110
5	194
6	230
7	302
8	434
9	590
10	770
11	974
12	1202
13	2030
14	3074
15	4334
16	5294
17	5810

Default: $Lebedev = 9$.

`MaxDist`
${}^{d}MaxDist$

Setzt die Integrationsvariable $MaxDist$ auf den Wert ${}^{d}MaxDist$. Steht für die maximal erlaubte Entfernung zwischen Integrationsstrahl und Fläche während des *Intersection*-Algorithmus. *Default*: $MaxDist = 10^{-2}$.

`MaxRadius`
${}^{d}MaxRad$

Setzt die Integrationsvariable $MaxRad$ auf den Wert ${}^{d}MaxRad$. Dieser Wert bestimmt während der Domänenintegration die Maximallänge in bohr, die ein Integrationsstrahl annehmen darf, bevor er als *unendlich lang* behandelt wird *Default*: $MaxRad = 25$bohr.

`MinIter`
${}^{i}MinIter$

Setzt die Integrationsvariable $MinIter$ auf den Wert ${}^{i}MinIter$. Gibt die Anzahl an Iterationen an, die für die Minimierung im *Intersection*-Algorithmus erlaubt sind. *Default*: $MinIter = 500$.

`MinTry`
${}^{i}MinTry$

Setzt die Integrationsvariable $MinTry$ auf den Wert ${}^{i}MinTry$. Legt die Anzahl an Versuchen fest, die der *Intersection*-Algorithmus für die Minimierung hat. *Default*: $MinTry = 10$.

`nbFFT`
${}^{i}nbFFT$

Setzt Integrationsvariable $nbFFT$ auf den Wert ${}^{i}nbFFT$. Dieser Wert bestimmt die Anzahl an Koeffizienten für die DCT während der Clenshaw-Curtis-Quadratur an. *Default*: $nbFFT = 512$.

`Refresh` $^{i}Refresh$	Setzt Integrationsvariable $Refresh$ auf den Wert $^{i}Refresh$. Während der LEBEDEV-Integration wird die Anzeige für die geschätzte Rechenzeit alle $Refresh$ Punkte aktualisiert. *Default*: $Refresh = 20$.
`PPot` $^{d}PPot$	Setzt die Integrationsvariable $PPot$ auf den Wert $^{d}PPot$. Gibt den Faktor des Pseudopotentials in der Abstandsfunktion an. *Default*: $PPot = 1.0$.

C.1.3 Global Settings

`Range` $^{d}Range$	Setzt die globale Variable $Range$ auf den Wert $^{d}Range$. *Default*: $Range = 10^{-2}$.
`Resolution` $^{d}Resolution$	Setzt die globale Variable $Resolution$ auf den Wert $^{d}Resolution$. *Default*: $Resolution = 0.1$
`TermGrad` $^{d}TermGrad$	Setzt die globale Variable $TermGrad$ auf den Wert $^{d}TermGrad$. *Default*: $TermGrad = 10^{-10}$.

C.1.4 Graphic Settings

`GRAPH`	Schaltet die Erzeugung von GNUPLOT-Dateien während des Flächenfits AN.
`NOGRAPH`	Schaltet die Erzeugung von GNUPLOT-Dateien während des Flächenfits AUS.

C.1.5 Newton-Raphson Settings

`NewRaph`	Schaltet die Suche nach Sattelpunkten mit der Standard-NEWTON-RAPHSON-Methode an.
`LevMarq`	Schaltet die Suche nach Sattelpunkten mit der NEWTON-RAPHSON-Methode *und* LEVENBERG-MARQUARDT-Optimierung an.

C.1.6 Surface Settings

`InitStep` $^{d}InitStep$	Setzt die Flächenvariable $InitStep$ auf den Wert $^{d}InitStep$. Gibt die Schrittweite für den ersten Schritt weg vom kritischen Punkt während der Konstruktion von *AIM*- oder *ELF*-Grenzlächen an. *Default*: $InitStep = 10^{-1}$.

C.2 Inits

`WFNInit` $^{s}WFNFile$.wfn	Initialisiert das Wavefunction-Object. Läd das `wfn`-File $WFNFile$ und initialisiert Speicher und globale Felder.
`ELFInit`	Initialisiert das *ELF*-Object.
`InitFits` $^{i}nbFits$ $[^{s}FitName.\text{fit}]^{nbFits}$	Initialisiert das Fit-Object und liest alle *FitName* ein.

C.3 Critical Points

C.3.1 Bader-CPs

`FindBPs` $^{s}TempFile$.tmp	Sucht zwischen allen Atomen nach $(3,-1)$-kritischen Punkten in $\rho(\mathbf{r})$ mit einem einfachen NEWTON-RAPHSON-Algorithmus. Die gefundenen Punkte werden als Rohdaten in $TempFile$ geschrieben.
`SortBPs` $^{s}TempFile$.tmp $^{s}BPFile$.xlf	Sortiert die Rohdaten aus $TempFile$ und schreibt sie im `xlf`-Format in die Datei $BPFile$. Bei Punkten mit sehr ähnlichen Koordinaten (bis auf 10^{-3} bohr) wird derjenige mit dem niedrigsten Gradienten übernommen.

C.3.2 *ELF*-CPs

`FindCPRange` $^{s}TempFile$.tmp	Im gesamte Bereich $\eta(\mathbf{r}) \leq Range$ werden alle Punkte auf einem Raster mit der Auflösung *Resolution* als Startpunkt für eine Sattelpunktsuche genommen. Die gefundenen Punkte stehen als Rhodaten in $TempFile$.
`MergeFiles` $^{s}In1$.tmp $^{s}In2$.tmp ^{s}Out.tmp	Fügt die temporären Datein $In1$ und $In2$ zu der neuen Datei Out zusammen.
`ReFineCPs` $^{i}Iter$ $^{d}TermGrad$ ^{s}In.xlf ^{s}Out.tmp	Versucht die Genauigkeit der CPs in *In* auf *TermGrad* in *Iter* Schritten zu erhöhen. Misslingt das bleibt der Originalpunkt erhalten. Die Ausgabe erfolgt als Rohdaten in *Out*.

`ReFineGuess` $^{i}Iter$ $^{d}TermGrad$ ^{s}In.xlf ^{s}Out.tmp	Versucht die Genauigkeit der CPs in *In* auf *TermGrad* in *Iter* Schritten zu erhöhen. Misslingt das wird der Punkt verworfen. Die Ausgabe erfolgt als Rohdaten in *Out*.
`ScanCPRange` $^{s}TempFile$.tmp	Es wird ein kubisches Raster für den Bereich $\eta(\mathbf{r}) \leq Range$ mit der Auflösung *Resolution* im Speicher angelegt und mit Werten für $\eta(\mathbf{r})$ gefüllt. Danach wird eine Gridsuche nach Sattelpunkten gestartet. Die gefundenen Punkte stehen als Rhodaten in $TempFile$.
`ScanCPs` $^{d}XMin$ $^{d}XMax$ $^{d}YMin$ $^{d}YMax$ $^{d}ZMin$ $^{d}ZMax$ $^{s}TempFile$.tmp	Wie `ScanCPRange` jedoch mit den angegebenen Grenzen für das Grid.
`SortCPs` $^{s}TempFile$.tmp $^{s}CPFile$.xlf	Sortiert die Rohdaten aus $TempFile$ und schreibt sie im `xlf`-Format in die Datei $CPFile$. Bei Punkten mit ähnlichen Koordinaten wird derjenige mit dem niedrigsten Gradienten übernommen.

C.4 Fittings

C.4.1 Core-Fits

`FitCores` ^{s}*TempFile*.tmp ^{s}*CoreFile*.xlf	Fittet um alle Kerne spärische ELF-Domänen an. Temporäre Ergebnisse werden in *TempFile* abgelegt. Nach dem Lauf stehen die Ergebnisse im `xlf`-Format in *CoreFile*.

C.4.2 Surface-Fits

`FitSurfaces` ^{s}*SrfFile*.srf ^{s}*FitName*	Fittet die Fläche in *SrfFile* entsprechend den Fit-Parametern.

C.5 I/O Commands

C.5.1 Einlesen von Daten

`ReadBPs` ^{s}*BPFile*.xlf	Liest die BADER-CPs aus *BPFile* in den Speicher. Ist das Flag *VIS* gesetzt, werden die Punkte im DISLIN-Fenster angezeigt.
`ReadCoreFits` ^{s}*CoreFile*.xlf	Liest die sphärischen *ELF*-Domänen aus *CoreFile* in den Speicher. Ist das Flag *VIS* gesetzt, werden die Sphären im DISLIN-Fenster angezeigt.
`ReadCPs` ^{s}*CPFile*.xlf	Liest die *ELF*-CPs aus *CPFile* in den Speicher. Ist das Flag *VIS* gesetzt, werden die Punkte im DISLIN-Fenster angezeigt.

C.5.2 Grenzflächen

`CleanSurfaces` ^{s} *In*.srf ^{s}*Out*.srf	Glättet die Flächen in *In* und schreibt sie nach *Out*.
`WriteBader` ^{i}*nbAngl* ^{i}*nbSteps* ^{d}*StepLen* ^{s}*SrfFile*.srf	Konstruiert ausgehend von allen BADER-CPs im Speicher die atomaren Grenzflächen mit *nbAngl* Pfaden zu je *nbSteps* Schritten der Länge *StepLen*. Die Flächen werden im `srf`-Format in *SrfFile* geschrieben.
`WriteELFSrf` ^{i}*nbAngl* ^{i}*nbSteps* ^{d}*StepLen* ^{s}*SrfFile*.srf	Konstruiert ausgehend von allen *ELF*-CPs im Speicher die Lokalisationsgrenzflächen mit *nbAngl* Pfaden zu je *nbSteps* Schritten der Länge *StepLen*. Die Flächen werden im `srf`-Format in *SrfFile* geschrieben.

`WriteFit` $^{i}pFit$ $^{i}nbAngl$ $^{i}nbSteps$ $^{d}StepLen$ $^{s}SrfFile$.srf	Konstruiert entsprechend dem Fit *pFit* im Speicher eine Grenzfläche mit *nbAngl* Pfaden zu je *nbSteps* Schritten der Länge *StepLen*. Die Flächen werden im `srf`-Format in *SrfFile* geschrieben.
`WriteAllFits` $^{i}nbAngl$ $^{i}nbSteps$ $^{d}StepLen$ $^{s}SrfFile$.srf	Wie `WriteFit`, jedoch werden alle Fits im Speicher behandelt.

C.5.3 Graphic-Output

`WriteELFRange` $^{s}File$ $^{s}Form$	Schreibt ein Grid der Auflösung *Resolution* für die *ELF*-Werte $\eta(\mathbf{r}), \forall \mathbf{r} \leq Range$ im Format *Form* (CUB, GNU, POV) in *File*.
`WriteELF` $^{d}XMin$ $^{d}XMax$ $^{d}YMin$ $^{d}YMax$ $^{d}ZMin$ $^{d}ZMax$ $^{s}File$ $^{s}Form$	Wie `WriteELFRange` jedoch im angegebenen Bereich.
`WriteELFCut` $^{v}Vec1$ $^{i}nb1$ $^{v}Vec2$ $^{i}nb2$ $^{v}Vec3$ $^{i}nb3$ $^{s}GnuFile$.gnudat	Schreibt ein Grid entlang der Vektoren *Vec1*, *Vec2*, *Vec3* mit den Punktanzahlen *nb1*, *nb2*, *nb3* im `gnu`-Format nach *GnuFile*.
`WriteELFRangePLT`	Schreibt ein Grid der Auflösung *Resolution* für die *ELF*-Werte $\eta(\mathbf{r}), \forall \mathbf{r} \leq Range$ im `plt`-Format ins Scratch-Verzeichnis.

`WriteELFPLT` $^{d}XMin\ ^{d}XMax$ $^{d}YMin\ ^{d}YMax$ $^{d}ZMin\ ^{d}ZMax$	Wie `WriteELFRangePLT` jedoch im angegebenen Bereich.
`WritePLT`	Schreibt das Graphik-Grid im `plt`-Format ins Scratch-Verzeichnis.
`WriteRho` $^{d}XMin\ ^{d}XMax$ $^{d}YMin\ ^{d}YMax$ $^{d}ZMin\ ^{d}ZMax$ $^{s}File$ $^{s}Form$	Wie `WriteELF` jedoch mit $\rho(\mathbf{r})$.

C.6 Graphics

`DrawSurface` ^{s}*SrfFile*.srf	Zeichnet die Grenzfläche(n) in *SrfFile* in das Graphik-Grid. Ist das Flag *VIS* gesetzt, werden die Flächen mit *ZBuffer* im DisLin-Fester angezeigt.
`InitGridRange` [`Res`|`Points`] [^{d}*Resolution*|^{i}*nbPoints*]	Initialisiert das dreidimensionale Graphik-Grid für den Bereich $\eta(\mathbf{r}) \leq$*Range*. Mit dem Kommando `Res` legt man die Auflösung des Grids zu *Resolution* fest, mit `Points` setzt man die maximale Punktzahl pro Dimension auf *nbPoints* fest. Ist nicht genügend Speicher vorhanden wird die Auflösung solange verringert, bis das Grid in den Speicher passt.
`InitGrid` $^{d}XMin$ $^{d}XMax$ $^{d}YMin$ $^{d}YMax$ $^{d}ZMin$ $^{d}ZMax$ [`Res`|`Points`] [^{d}*Resolution*|^{i}*nbPoints*]	Wie `InitGridRange` jedoch mit den angegebenen Griddimensionen.
`PlotSurface` ^{s}*SrfFile*.srf	Zeichnet die Grenzfläche(n) in *SrfFile* in das Graphik-Grid. Ist das Flag *VIS* gesetzt, werden die Flächen *ohne ZBuffer* im DisLin-Fester angezeigt.
`PlotCPs` ^{s}*CPFile*.xlf	Plottet einen 2D-Konturplot um jeden CP in *CPFile* in seiner Eigenebene.

C.7 Integration

`IntDomain`	Integriert vom Aufpunkt (X, Y, Z)
${}^{d}X\ {}^{d}Y\ {}^{d}Z$	mit den Integrationsparametern.
${}^{i}Mode$	$Mode \neq 0$ ist der Debugging-Mode.

Literaturverzeichnis

[1] R. F. W. Bader, *Atoms in Molecules*, Oxford University Press, Oxford 1990.

[2] A. D. Beckem, E. Edgecombe, *J. Chem. Phys.* **92** (1990) 5397.

[3] P. L. A. Popelier, F. M. Aicken, S. E. O'Brien, in *Chemical Modelling: Applications and Theory, Volume 1*, Kap. 3 – Atoms in Molecules, The Royal Society of Chemistry 2000.

[4] P. L. A. Popelier, *Sci. Comp. World.* **45** (1999) 26.

[5] R. F. W. Bader, T. T. Nguyen-Dang, Y. Tal, *Rep. Prog. Phys.* **44** (1981) 893.

[6] Schwinger, *J. Phys. Rev.* **82** (1951) 914.

[7] R. F. W. Bader, *Chem. Rev.* **91** (1991) 893.

[8] R. F. W. Bader, P. L. A. Popelier, *Int. J. Quantum Chem.* **45** (1993) 189.

[9] A. Savin, R. Nesper, S. Wengert, T. F. Fässler, *Angew. Chem.* **109** (1997) 1892.

[10] A. Savin, B. Silvi, F. Colonna, *Int. J. Quantum Chem.* **74** (1996) 1088.

[11] F. W. Biegler-König, R. W. F. Bader, T. Tang, *J. Comp. Chem* **3** (1982) 317.

[12] S. Noery, X. Krokidis, F. Fuster, B. Silvi, *Computers & Chemistry* **23** (1999) 597.

[13] G. Jansen, M. Schubart, B. Findeis, L. H. Gade, I. J. Scowen, M. McPartlin, *J. Am. Chem. Soc.* **120** (1998) 7239.

[14] S. Raub, G. Jansen, *Theor. Chem. Acc.* **106** (2001) 223.

[15] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes in FORTRAN, 2nd Edition*, Cambridge University Press, Cambridge 1992.

[16] P. L. A. Popelier, *Computer Physics Communication* **108** (1998) 180.

[17] H. R. Schwarz, *Numerische Mathematik*, B. G. Teubner Stuttgart 1997.

[18] P. L. A. Popelier, *Molecular Physics* **87** (1996) 1169.

[19] V. I. Lebedev, D. N. Laikov, *Doklady Akademii Naik* **366** (1999) 741.

[20] S. L. Sobolev, *Sib. Math. J.* **3** (1962) 769.

[21] V. I. Lebedev, *Vych. Mat. Mat. Fiz* **16** (1976) 293.

[22] V. I. Lebedev, *Sib. Mat. Zh.* **18** (1977) 132.

[23] V. I. Lebedev, A. L. Skorokhodov, *Dokl. Akad. Nauk* **324** (1992) 519.

[24] V. I. Lebedev, *Dokl. Akad. Nauk* **338** (1994) 454.

[25] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, V. G. Zakrzewski, J. A. Montgomery, Jr., R. E. Stratmann, J. C. Burant, S. Dapprich, J. M. Millam, A. D. Daniels, K. N. Kudin, M. C. Strain, O. Farkas, J. Tomasi, V. Barone, M. Cossi, R. Cammi, B. Mennucci, C. Pomelli, C. Adamo, S. Clifford, J. Ochterski, G. A. Petersson, P. Y. Ayala, Q. Cui, K. Morokuma, D. K. Malick, A. D. Rabuck, K. Raghavachari, J. B. Foresman, J. Cioslowski, J. V. Ortiz, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. Gomperts, R. L. Martin, D. J. Fox, T. Keith, M. A. Al-Laham, C. Y. Peng, A. Nanayakkara, C. Gonzalez, M. Challacombe, P. M. W. Gill, B. Johnson, W. Chen, M. W. Wong, J. L. Andres, C. Gonzalez, M. Head-Gordon, E. S. Replogle, J. A. Pople, *Gaussian 98, Revision A.4*, Gaussian, Inc., Pittsburgh PA 1998.

[26] A. L. Allred, E. G. Rochow, *J. inorg. nucl. Chem* **5** (1957) 264.

[27] W. Kohn, L. Sham, *Phys. Rev. A* **140** (1965) 1133.