

Universität Bielefeld

Fakultät für Chemie

Masterarbeit

**Eine graphische Benutzeroberfläche
für hochdimensionale
Quantendynamiksimulationen**

Bearbeiter: Peter Protassow

Prüfer: Prof. Dr. Uwe Manthe

Zweitprüfer: Prof. Dr. Wolfgang Eisfeld

Abgabedatum: 18. Juni 2018



Hiermit versichere ich, die vorgelegte Masterarbeit selbstständig und ohne unzulässige Hilfe angefertigt zu haben. Die verwendeten Quellen und Hilfstexte sind vollständig angegeben und die Stellen der Arbeit, einschließlich Tabellen und Abbildungen, die anderen Werken im Sinn und Wortlaut entnommen wurden, als Entlehnung kenntlich gemacht. Die Bestimmungen der Masterprüfungsordnung sind mir bekannt. Die von mir vorgelegte Masterarbeit wurde in der Zeit vom 29. November 2017 bis 18. Juni 2018 im Arbeitskreis von Prof. Dr. Uwe Manthe an der Fakultät für Chemie der Universität Bielefeld unter der wissenschaftlichen Anleitung von Roman Ellerbrock durchgeführt.

Bielefeld, den 18. Juni 2018 .

.....
(Peter Protassow)

Inhaltsverzeichnis

1 Einleitung	1
2 Theorie	3
2.1 Ansatz der Multilayer-MCTDH-Wellenfunktion	5
3 Technische Details	11
4 Ergebnisse	12
4.1 Python-Interface für MCTDH	12
4.2 Graphische Benutzeroberfläche für MCTDH	18
5 Zusammenfassung und Ausblick	25
Anhang	26
Literaturverzeichnis	82

1 Einleitung

Das zeitabhängige Multikonfiguration-Hartree-Verfahren (MCTDH)^[1,2] und seine Multilayererweiterung (ml-MCTDH)^[3,4] sind effiziente Verfahren zur genauen Simulation mehrdimensionaler Quantendynamik, die von mehreren Forschungsgruppen verwendet werden^[5–23]. Beispiele für hochdimensionale Benchmark-Anwendungen sind die 21-dimensionalen Rechnungen, in denen die Tunnelaufspaltung des Grundzustands^[24–29] und der angeregten^[25–29] Schwingungszustände von Malonaldehyd erforscht wurden. Es wurden in 15-dimensionalen Rechnungen die Schwingungszustände von protonierten Wasserdimeren^[30–34] untersucht. Außerdem wurde in 12-dimensionalen Rechnungen die thermischen Geschwindigkeitskonstanten^[35–40], anfangszustandsausgewählte Reaktionswahrscheinlichkeiten^[41–45] und die state-to-state Reaktionswahrscheinlichkeiten^[46] für die Reaktion von Methan mit Wasserstoff berechnet. In diesen Rechnungen wurden detaillierte *ab initio* Potentialflächen verwendet. Signifikant höhere Dimensionen wurden in MCTDH-Rechnungen mit Modelhamiltonoperatoren untersucht. So wurde in wegweisenden Rechnungen die nichtadiabatischen Dynamiken von Pyrazin erforscht, in denen ein 24-modenschwingungsgekoppelter Hamiltonoperator^[47–49] verwendet wurde. Multilayer-MCTDH Simulationen von typischen physikalischen Modellen^[3,8,50–53] kondensierter Materie schließen üblich tausende Freiheitsgrade ein. Für die Untersuchung eines Photodissoziationsmodels wurden in einem Wirt-Gast-Komplex 189-dimensionale Multilayer-MCTDH Rechnungen durchgeführt^[54]. Für weiterführende Literatur, in der das MCTDH-Verfahren und seine Anwendungen diskutiert werden, siehe Refs.^[55–60].

Das MCTDH-Programmpaket, welches zur Berechnung und Simulation der oben genannten Systeme verwendet wird, bietet die Möglichkeit, Wellenfunktionen und Dichtematrizen hocheffizient zu propagieren, sowie Eigenzustände zu berechnen. Bisher ist die Bedienung des vorliegenden MCTDH-Programmpakets selbst bei Standardaufgaben Spezialisten vorenthalten, da ein tiefgreifendes Verständnis der Programmstruktur erforderlich ist. Im Rahmen dieser Masterarbeit wurden zwei wesentliche Verbesserungen unternommen: Es wurde eine Benutzeroberfläche (von englisch *graphical user interface*,

1 Einleitung

GUI) erstellt, die Wissenschaftlern ohne Programmierkenntnisse die Bedienung des Programms bei Standardaufgaben ermöglicht. Des Weiteren wurde eine Python-Schnittstelle für MCTDH entwickelt, die es erlaubt, komplizierte Aufgaben mithilfe von Skripten zu bewältigen. Diese Schnittstelle ermöglicht die Nutzung des MCTDH-Programmpakets, ohne ein tiefgreifendes Verständnis der Programmstruktur vorrauszusetzen. Python hat sich als einsteigerfreundliche Programmiersprache erwiesen. Vergleichbare Python-Schnittstellen existieren auch für viele andere bekannte und häufig genutzte numerische Programmpakete (TensorFlow^[61], SciPy^[62] usw.).

Diese Arbeit ist wie folgt gegliedert: In Kapitel 2 wird der Ansatz der MCTDH-Wellenfunktion beschrieben. Es werden die Unterschiede zum Multilayer-MCTDH hervorgehoben. Die technische Beschreibung der GUI erfolgt in Kapitel 3. In Kapitel 4 werden die Python-Schnittstelle und die graphische Benutzeroberfläche vorgestellt. Schließlich wird in Kapitel 5 die Arbeit zusammengefasst und ein Ausblick gegeben.

2 Theorie

Die Effizienz des MCTDH-Verfahrens resultiert aus der doppellagigen Struktur der verwendeten Wellenfunktion. In anderen Wellenpaktdynamikverfahren, wie der Standardmethode^[60], wird die Wellenfunktion direkt in einer zeitunabhängigen Basis oder einem zeitunabhängigen Gitter dargestellt. Anstatt die Wellenfunktion in einer zeitunabhängigen Basis zu entwickeln, erfolgt im MCTDH-Verfahren die Darstellung des korrelierten mehrdimensionalen Wellenpaketes in einem Satz zeitabhängiger Basisfunktionen. Diese zeitabhängigen Basisfunktionen werden Einteilchenfunktionen (von englisch *single-particle function*, SPF) genannt. Die SPFs werden in der primitiven zeitunabhängigen Basis oder dem Gitter ausgedrückt. Das MCTDH-Verfahren kann als eine Zweilagendarstellung angesehen werden: So bilden die Entwicklungskoeffizienten, die genutzt werden, um die korrelierte Wellenfunktion in dem Satz der SPF-Basis auszudrücken, die obere Lage. Die zeitabhängigen Entwicklungskoeffizienten, welche die zeitabhängigen SPFs in der primitiven zeitunabhängigen Basis darstellen, bilden die untere Lage. Die Bewegungsgleichungen, durch die gleichzeitig die optimalen Entwicklungskoeffizienten für beide Lagen bestimmt werden, ergeben sich aus dem Dirac-Frenkel Variationsprinzip.

Dennoch ist auch das MCTDH-Verfahren durch die Anzahl der korrelierten Koordinaten limitiert. Die Effizienz des MCTDH-Verfahrens resultiert aus der Größe der SPF-Basis, welche, verglichen mit der primitiven Basis, signifikant kleiner gewählt werden kann. Allerdings skaliert der numerische Aufwand des MCTDH-Verfahrens noch immer exponentiell mit der Anzahl der korrelierten Koordinaten. Um Korrelationseffekte beschreiben zu können, sind mindestens zwei SPFs in jeder dieser Koordinaten notwendig. Die Anzahl der Konfigurationen, die in der MCTDH-Wellenfunktion enthalten ist, beträgt bei f korrelierten Koordinaten mindestens 2^f Konfigurationen. Aufgrund dieser Limitierung konnten mit dem MCTDH-Verfahren Systeme mit maximal 12-14 korrelierten Koordinaten berechnet werden.^[35–37,47,63–66]

Im von Meyer eingeführten Moden-Kombinationsverfahren kann die Anzahl der Konfigurationen reduziert werden. In diesem Verfahren werden die „logischen“ Koordina-

ten, die in der MCTDH-Darstellung verwendet werden, von physikalischen Koordinaten unterschieden. Es werden verschiedene physikalische Koordinaten zu einzelnen logischen Koordinaten kombiniert. Analog zur Theorie der Elektronenstruktur werden diese mehrdimensionalen logischen Koordinaten Partikel genannt. Folglich wird die MCTDH-Rechnung statt durch die korrelierten Koordinaten durch die Partikel limitiert. So konnten Systeme mit 15-24 korrelierten Freiheitsgraden^[31,32,49,67] und System-Bad-Modelle^[68-70] durch das Moden-Kombinationsverfahren behandelt werden. Zwar konnte durch dieses Verfahren die Grenze zu höherer Dimensionalität verschoben werden, dennoch bleibt die grundlegende Einschränkung: Der numerische Aufwand skaliert mindestens mit 2^p , wobei p die Anzahl der logischen Koordinaten bzw. Partikel wiedergibt. Die Anzahl an physikalischen Koordinaten, die zu logische Koordinaten zusammengefasst werden können, ist begrenzt, da die SPFs nun mehrdimensionale Wellenfunktionen darstellen. Für molekulare Systeme stellte sich heraus, dass die Kombination von mehr als drei bis vier Koordinaten in einem Partikel ineffizient ist.

Die Begrenzung durch die Anzahl der korrelierten Koordinaten bzw. Partikel konnte durch das Multilayer-MCTDH-Verfahren^[3] überwunden werden. Die SPF_s des MCTDH-Verfahrens können ebenfalls als MCTDH-Wellenfunktion dargestellt werden. So kann z.B. eine MCTDH-Wellenfunktion um eine Lage erweitert werden, sodass die Wellenfunktion in drei Lagen ausgedrückt wird: Die oberste Lage wird durch die zeitabhängigen Entwicklungskoeffizienten gebildet. Die Wellenfunktion wird in der SPF-Basis der ersten Lage dargestellt, d.h. den SPF_s des einfachen MCTDHs. Die mittlere Lage wird durch die zeitabhängigen Entwicklungskoeffizienten gebildet, welche die SPF_s der ersten Lage in der SPF-Basis der zweiten Lage darstellen. D.h. die SPF_s der ersten Lage werden in der SPF-Basis einer zusätzlichen Lage ausgedrückt, die im Multilayer-MCTDH-Verfahren hinzugekommen ist. Schließlich werden in der untersten Lage die SPF_s der zweiten Lage in der primitiven zeitunabhängigen Basis dargestellt. Durch eine rekursive Anwendung des MCTDH-Verfahrens können weitere Lagen hinzugefügt werden. Mit dem Multilayer-MCTDH-Verfahren waren quantumdynamische Rechnungen von System-Bad Modellen mit bis zu 1000 korrelierten Koordinaten möglich, in denen Elektronentransferprozesse^[3,50] untersucht wurden.

Die Propagation der MCTDH-Wellenfunktion setzt die effiziente Berechnung der Matrixelemente des Hamiltonoperators voraus. Solange der Hamiltonoperator der Summe von Produkten von Einteilchenoperatoren^[2] entspricht, stellt die Berechnung der Matrixelemente kein Problem dar. Im Gegensatz zu vielen Modelhamiltonoperatoren können *ab*

initio Potentialenergieflächen aber nur selten in dieser Form dargestellt werden. Durch die Verwendung einer spezifischen zeitabhängigen Quadratur, welche die Matrixelemente allgemeiner Potentiale effizient auswertet, können auch Matrixelemente resultierend aus *ab initio* Potentialenergieflächen effizient berechnet werden. Dieses Verfahren wird correlation discrete variable representation (CDVR) [71,72] genannt.

Das ursprüngliche Vorgehen für das CDVR^[71] beruht auf einem zeitabhängigen DVR-Gitter, das einer SPF-Basis entspricht. Somit kann das Standard-CDVR weder für modenkombinierte MCTDH-Rechnungen noch für Berechnungen mit dem Multilayer-MCTDH-Ansatz verwendet werden. Hingegen ist das mehrdimensionale CDVR-Verfahren mit dem modenkombinierten MCTDH-Verfahren kompatibel.^[73] Der numerische Aufwand des CDVRs hängt linear von der Anzahl der verwendeten primitiven Gitterpunkten ab, die für die Darstellung der SPFs benötigt werden. In modenkombinierten MCTDH-Rechnungen wird eine große Anzahl an primitiven Gitterpunkten verwendet, sodass modenkombinierte MCTDH-Rechnungen kombiniert mit der CDVR-Auswertung des Potentials ineffizient sind. Multilayer-MCTDH-Rechnungen benötigen dagegen keine mehrdimensionalen Gitter, um die SPFs darzustellen und bieten sich daher in Kombination mit dem CDVR an.

2.1 Ansatz der Multilayer-MCTDH-Wellenfunktion

Zunächst werden die Ansätze der Wellenfunktionen der Standardmethode, des Zweilagen-MCTDHs und des modenkombinierten MCTDH betrachtet, um anschließend den Ansatz der Multilayer-MCTDH-Wellenfunktion stufenweise vorzustellen.

In der Standardmethode wird die Wellenfunktion in einer zeitunabhängigen Basis bzw. in einem zeitunabhängigen Gitter dargestellt. Die mehrdimensionale Wellenfunktion wird durch das direkte Produkt von eindimensionalen Basisfunktionen $\mathcal{X}_j^\kappa(x_\kappa)$ wie folgt dargestellt:

$$\Psi(x_1, \dots, x_f, t) = \sum_{j_1=1}^{N_1} \dots \sum_{j_f=1}^{N_f} A_{j_1, \dots, j_f}^1(t) \cdot \mathcal{X}_{j_1}^{(1)}(x_1) \cdot \dots \cdot \mathcal{X}_{j_f}^{(f)}(x_f) \quad (2.1)$$

Die zeitabhängigen Koeffizienten $A_{j_1, \dots, j_f}^1(t)$ beschreiben die Bewegung der Wellenpakete. Die Darstellung der Wellenfunktion in Gleichung 2.1 kann auch als einlagige Darstellung angesehen werden und die Hochzahl 1 von $A_{j_1, \dots, j_f}^1(t)$ soll darauf hinweisen,

dass $A_{j_1, \dots, j_f}^1(t)$ ein Entwicklungskoeffizient der ersten (und einzigen) Lage ist.

Im MCTDH-Verfahren wird eine zusätzliche Lage für die Darstellung der Wellenfunktion eingeführt. Die mehrdimensionale Wellenfunktion wird erst in einer orthonormalen Basis der zeitabhängigen SPF's $\phi_j^{1;\kappa}(x_\kappa, t)$ entwickelt.

$$\Psi(x_1, \dots, x_f, t) = \sum_{j_1=1}^{n_1} \dots \sum_{j_f=1}^{n_f} A_{j_1, \dots, j_f}^1(t) \cdot \phi_{j_1}^{1;1}(x_1, t) \cdot \dots \cdot \phi_{j_f}^{1;f}(x_f, t). \quad (2.2)$$

Anschließend werden diese SPF's innerhalb der zeitunabhängigen primitiven Basis dargestellt:

$$\phi_m^{1;\kappa}(x_\kappa, t) = \sum_{j=1}^{N_\kappa} A_{m;j}^{2;\kappa}(t) \cdot \mathcal{X}_j^{(\kappa)}(x_\kappa). \quad (2.3)$$

Gleichung 2.3 beinhaltet einen Satz zusätzlicher Entwicklungskoeffizienten, $A_{m;j}^{2;\kappa}(t)$, der die zeitabhängigen SPF's in der zeitunabhängigen primitiven Basis darstellt. Die hochgestellte Zahl z der Koeffizienten $A^z(t)$ bezieht sich auf die Lagentiefen. In Gleichung 2.3 folgt aus $z = 2$, dass Gleichung 2.3 die zweite Lage darstellt. Das hochgestellte κ und der Index m von $A_{m;j}^{2;\kappa}(t)$ beziehen sich auf die m -te SPF der κ -ten Koordinate.

Zur Visualisierung unterschiedlicher MCTDH-Wellenfunktionen, für die unterschiedlich viele Lagen verwendet wurden, dienen Baumdiagramme, wie sie in Abbildung 2.1.1 gezeigt werden. Als Beispiel soll ein siebendimensionales System dienen. Die Standardwellenpaketdarstellung aus Gleichung 2.1 und die MCTDH-Darstellung sind in Abbildung 2.1.1a und 2.1.1b schematisch dargestellt. In den Diagrammen sind die verschiedenen Sätze der A-Koeffizienten durch die ausgefüllten schwarzen Kreise gekennzeichnet. So kommt in Abbildung 2.1.1a ein Satz von Koeffizienten A_{j_1, \dots, j_7}^1 vor, der durch den einzigen schwarzen Punkt gekennzeichnet ist. Jede Linie, die von solchen Kreisen führt, entspricht einem tiefgestellten Index aus $A_{m;j}^{2;1}$ und die Zahl neben einer Linie gibt den maximalen Wert des Indexes an, d.h. die jeweilige Basisgröße. Die tieferliegende primitive Darstellung wird durch den Koordinatdeskriptor x_n hervorgehoben. Beispielsweise ist der Koeffizient $A_{m;j}^{2;1}$, der durch die SPF's $\phi_{j_1}^{1;1}(x_1, t)$ dargestellt wird, mit dem Koeffizienten A^1 über eine Linie verbunden, die den m -ten Index darstellt. Über eine weitere Linie, die für den j -ten Index steht, ist $A_{m;j}^{2;1}$ mit dem Koordinatdeskriptor x_1 verbunden. In Abbildung 2.1.1a ist der Koeffizient A_{j_1, \dots, j_7}^1 durch sieben Linien mit den entsprechenden Indizes von j_1 bis j_7 direkt mit den sieben Koordinatdeskriptoren x_1, x_2, \dots, x_7 verbunden und in Abbildung 2.1.1b ist A_{j_1, \dots, j_7}^1 mit den sieben Sätzen an A-Koeffizienten

$A^{2;1}, A^{2;2}, \dots, A^{2;7}$ verbunden. Die Wellenfunktion aus Abbildung 2.1.1a ist direkt in der zeitunabhängigen primitiven Basis dargestellt. In Abbildung 2.1.1b gibt es eine intermediäre Lage an zeitabhängigen SPFen.

Während in den Gleichungen 2.2 und 2.3 nur eindimensionale SPFen vorkommen, werden im modenkombinierten MCTDH-Verfahren mehrdimensionale SPFen verwendet. Hierfür werden die f physikalischen Koordinaten x_1, x_2, \dots, x_f d logischen Gruppen zugeordnet, die Partikel genannt werden. Die mehrdimensionalen Koordinaten $q_1^1, q_2^1, \dots, q_d^1$ sind wie folgt definiert:

$$\begin{aligned} q_1^1 &= \{q_1^{2;1}, q_2^{2;1}, q_{d_1}^{2;1}\} \\ &= \{x_1, x_2, \dots, x_{d_1}\} \\ q_2^1 &= \{q_1^{2;2}, q_2^{2;2}, q_{d_2}^{2;2}\} \\ &= \{x_{d_1+1}, x_{d_1+2}, \dots, x_{d_1+d_2}\} \\ &\vdots \\ q_{f^1}^1 &= \{q_1^{2;d}, q_2^{2;d}, q_{d_d}^{2;d}\} \\ &= \{\dots, x_f\} \end{aligned}$$

Die logische mehrdimensionale Koordinate q_κ^1 umfasst d_κ Koordinaten: $q_1^{2;\kappa}, q_2^{2;\kappa}, \dots, q_{d_\kappa}^{2;\kappa}$. Die Hochzahlen 1 und 2 in dieser Notation zeigen, ob die Koordinate eine mehrdimensionale Koordinate der ersten Lage ist oder einer Koordinate der zweiten Lage entspricht. Für die Koordinate der zweiten Lage gibt der zusätzliche hochgestellter Index κ den Index der Koordinate der ersten Lage an. Eine modenkombinierte MCTDH-Wellenfunktion mit d logischen Koordinaten wird wie folgt definiert:

$$\Psi(q_1^1, q_2^1, \dots, q_d^1, t) = \sum_{j_1=1}^{n_1} \dots \sum_{j_d=1}^{n_d} A_{j_1, \dots, j_d}^1(t) \cdot \phi_{j_1}^{1;1}(q_1^1, t) \cdot \dots \cdot \phi_{j_d}^{1;d}(q_d^1, t) \quad (2.4)$$

$$\begin{aligned} \phi_m^{1;\kappa}(q_\kappa^1, t) &= \sum_{j=1}^{N_\alpha} \dots \sum_{j_{d_\kappa}=1}^{N_\beta} A_{m; j_1, \dots, j_{d_\kappa}}^{2;\kappa}(t) \cdot \mathcal{X}_{j_1}^{(\alpha)}(q_1^{2;\kappa}) \cdot \dots \cdot \mathcal{X}_{j_{d_\kappa}}^{(\beta)}(q_{d_\kappa}^{2;\kappa}) \\ &\left(\text{mit } \alpha = 1 + \sum_{i=1}^{\kappa-1} d_i \text{ und } \beta = 1 + \sum_{i=1}^\kappa d_i \right) \end{aligned} \quad (2.5)$$

In Abbildung 2.1.1c ist das entsprechende Diagramm der Wellenfunktion eines siebendimensionalen Systems dargestellt, dessen Koordinaten in logischen Koordinaten zusam-

mengefasst wurden. Die physikalischen Koordinaten bilden in Abbildung 2.1.1c drei mehrdimensionale logische Koordinaten: $q_1^1 = \{x_1, x_2\}$, $q_2^1 = \{x_3, x_4\}$ und $q_3^1 = \{x_5, x_6, x_7\}$.

Im einfachsten Fall kommen im Multilayer-MCTDH zwei Lagen von SPF vor. Anstelle die SPF aus Gleichung 2.4 in der primitiven zeitunabhängigen Basis zu entwickeln, können die mehrdimensionalen SPF ebenfalls durch das MCTDH-Verfahren dargestellt werden. Diese Entwicklung resultiert in einer Multilayer-MCTDH-Wellenfunktion. Der Ansatz der Multilayer-MCTDH-Wellenfunktion ist gegeben durch

$$\Psi(q_1^1, q_2^1, \dots, q_d^1, t) = \sum_{j_1=1}^{n_1} \dots \sum_{j_d=1}^{n_d} A_{j_1, \dots, j_d}^1(t) \cdot \phi_{j_1}^{1;1}(q_1^1, t) \cdot \dots \cdot \phi_{j_d}^{1;d}(q_d^1, t) \quad (2.6)$$

$$\begin{aligned} \phi_m^{1;\kappa}(q_\kappa^1, t) &= \phi_m^{1;\kappa}(q_1^{2;\kappa}, \dots, q_{d_\kappa}^{2;\kappa}, t) \\ &= \sum_{j=1}^{n_{\kappa,1}} \dots \sum_{j_{d_\kappa}=1}^{n_{\kappa,d_\kappa}} A_{m;j_1, \dots, j_{d_\kappa}}^{2;\kappa}(t) \\ &\quad \times \phi_{j_1}^{2;\kappa,1}(q_1^{2;\kappa}, t) \cdot \dots \cdot \phi_{j_{d_\kappa}}^{2;\kappa,d_\kappa}(q_{d_\kappa}^{2;\kappa}, t), \end{aligned} \quad (2.7)$$

$$\begin{aligned} \phi_m^{2;\kappa,\lambda}(q_\lambda^{2;\kappa}, t) &= \sum_{j=1}^{N_\alpha} A_{m;j}^{3;\kappa,\lambda}(t) \mathcal{X}_j^{(\alpha)}(q_\lambda^{2;\kappa}) \\ &\quad \left(\text{mit } \alpha = \lambda + \sum_{i=1}^{\kappa-1} d_i \right). \end{aligned} \quad (2.8)$$

In Gleichung 2.7 ist $\phi_m^{2;\kappa,\lambda}$ mit der dazugehörigen Koordinate $q_\lambda^{2;\kappa}$ eine SPF der zweiten Lage. Die Hochzahl 2 bezieht sich auf die Lagentiefe, sodass die SPF zur zweiten Lage gehört, wobei κ und λ die dazugehörigen Koordinaten kennzeichnen. Die Entwicklungskoeffizienten $A_{m;j}^{3;\kappa,\lambda}(t)$ werden verwendet, um diese SPF darzustellen und die Entwicklungskoeffizienten $A_{m;j_1, \dots, j_{d_\kappa}}^{2;\kappa}(t)$ definieren die Entwicklung der SPF der ersten Lage, die in der SPF-Basis der zweiten Lage entwickelt werden. In Gleichung 2.3 definierten diese Entwicklungskoeffizienten die SPF der ersten Lage in der primitiven zeitunabhängigen Basis. Abbildung 2.1.1d zeigt das entsprechende Diagramm für ein siebendimensionales System in Multilayer-MCTDH-Darstellung.

Da die Gleichungen der Multilayer-MCTDH-Wellenfunktion schnell unhandlich werden, ist es einfacher statt der Gleichungen die Wellenfunktionen wie in Abbildung 2.1.1 als Diagramm anzugeben. Jedes Diagramm definiert die Wellenfunktionen eindeutig, so dass aus den Diagrammen die entsprechenden Gleichungen der Wellenfunktionen abgeleitet werden können. Die Notation für die SPF, A-Koeffizienten und (mehrdimensionalen)

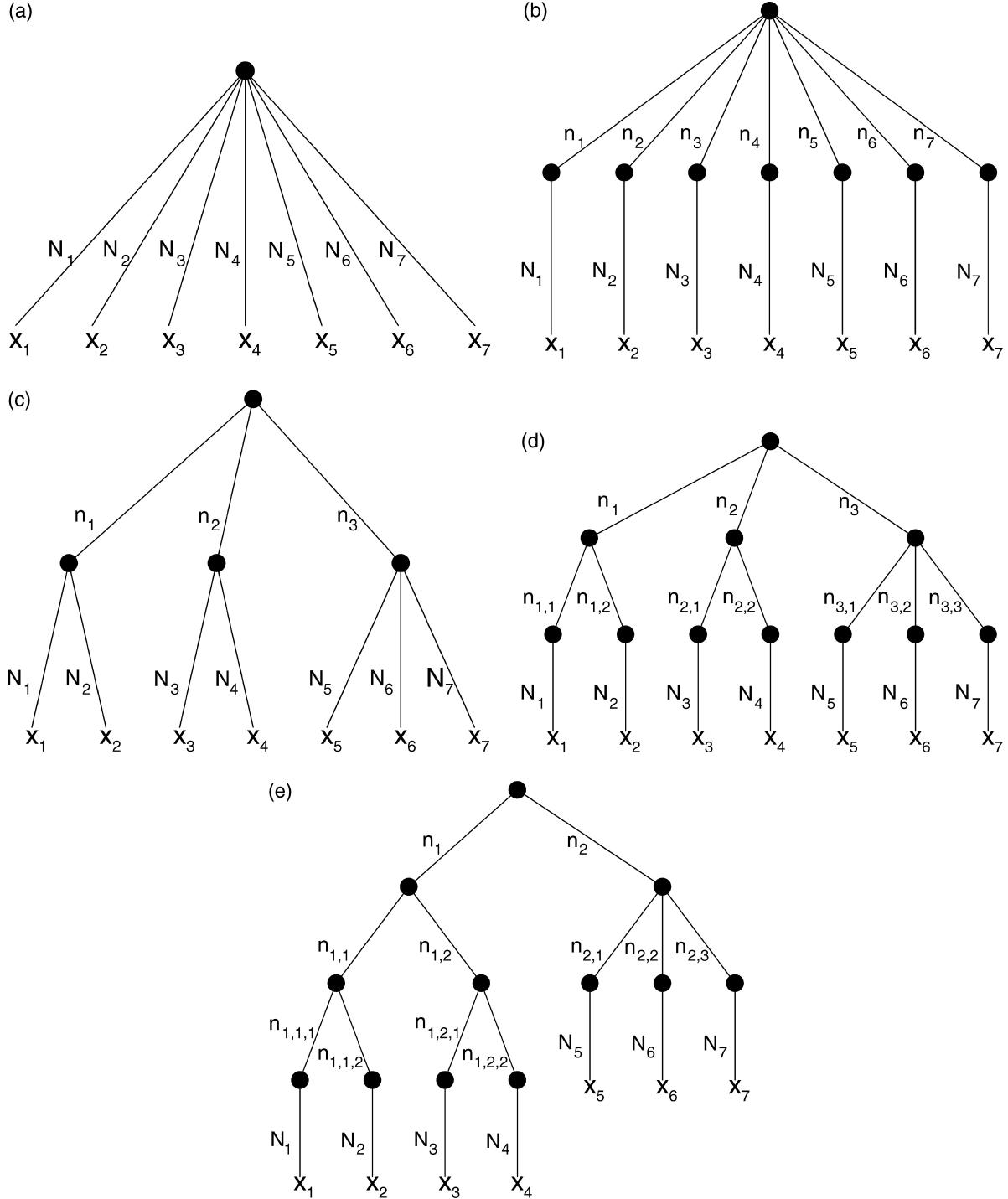


Abbildung 2.1.1: Unterschiedliche Darstellungen von Wellenfunktionen eines siebendimensionalen Systems. Dargestellt sind: (a) Eine Darstellung eines Standardwellenpakets, (b) eine MCTDH-Wellenfunktion, (c) eine moden-kombinierte MCTDH-Wellenfunktion, [(d) und (e)] eine Multilayer-MCTDH-Wellenfunktion. Die Diagramme wurden aus Ref. [4] entnommen.

Koordinaten, die oben in den Gleichungen angegeben wurde, kann einfach für beliebig mehrlagige Darstellungen erweitert werden. Ein Beispiel dieser Strukturen ist in Abbildung 2.1.1e dargestellt. Die Anzahl der Lagen für die jeweiligen Koordinaten kann bis zur primitiven Darstellung von Koordinate zu Koordinate variieren. So wird in Abbildung 2.1.1e das Baumdiagramm einer Multilayer-MCTDH-Wellfunktion gezeigt, in der für die Koordinaten $x_1 - x_4$ drei MCTDH-Lagen verwendet werden (d.h. insgesamt eine Vier-Lagen-Darstellung) und für die Koordinaten $x_5 - x_7$ werden zwei MCTDH-Lagen verwendet (d.h. eine Drei-Lagen-Darstellung).

3 Technische Details

Die Python-Schnittstelle für MCTDH wurde in der Programmiersprache Cython erstellt. Cython stellt eine Hybridprogrammiersprache dar, die Python und C\C++ kombiniert. Im Gegensatz zu Python wird Cython kompiliert, wobei die Cythonsyntax der Python-syntax ähnlich ist. In Cython kann C++ Code verwendet werden, der beim Erstellen der Cython-Programme ebenfalls kompiliert wird. Auf diese Weise wurden mehrere *mctdh*-Klassen auf Basis von C++ Klassen erstellt.

Die GUI für die MCTDH-Rechnungen wurde in Python und Qt implementiert. Der Zugriff auf die Qt-Bibliothek erfolgt über die Python-Bibliothek PyQt4. Das Design der einzelnen GUI-Fenster erfolgte in Qt-Designer. Die Informationen über die jeweiligen Fenster werden in *ui*-Dateien gespeichert. Mit PyQt können diese Dateien eingelesen werden und die entsprechenden Klassen erstellt und beliebig erweitert werden. Die in Qt-Designer erstellten Fenster werden in PyQt miteinander verknüpft.

Neben PyQt wurden die Python-Module *networkx* und *matplotlib* verwendet. Mithilfe von Klassen aus *networkx* und *matplotlib* konnten Baumdiagramme erstellt und gespeichert werden, die in der GUI zur Visualisierung des MCTDH-Baums verwendet wurden.

4 Ergebnisse

4.1 Python-Interface für MCTDH

Es wurde eine Programmierschnittstelle (von englisch *application programming interface*, API) für das MCTDH-Programmpaket erstellt, welche es erlaubt Klassen und Funktionen des in C++ verfassten MCTDH-Codes in Python-Skripten zu verwenden. Die Schnittstelle umfasst Klassen und Methoden, die zur Verwaltung der MCTDH-Basis dienen. Die Erweiterung auf andere Klassen kann nach einem einfachen Schema erfolgen. Abbildung 4.1.1 zeigt ein Klassendiagramm aller Klassen, welche über die API aufgerufen werden können. Jede Klasse der API wird in Abbildung 4.1.1 durch ein Rechteck repräsentiert. Im oberen Teil des Rechtecks ist der Klassenname angegeben und im unteren Teil sind die Methoden der Klasse aufgelistet. Die baumartige Anordnung der Rechtecke repräsentiert die Abhängigkeit der Klassen zueinander. Eine Klasse hängt von allen Klassen ab, die sich im Diagramm unterhalb der betrachteten Klasse befinden und zu denen eine Verbindung besteht. In der Klasse *Controlparameters* werden die Genauigkeitsparameter der MCTDH-Rechnung verwaltet. Die Klasse *physCoor* verwaltet Informationen einer Koordinate und enthält das primitive Gitter oder die primitive Basis. Sie ist ein Memberobjekt der Klasse *mctdhNode*, welche einen Knoten im MCTDH-Baum repräsentiert. In der Klasse *mctdhNode* wird die lokale Konnektivität des Knoten, sowie dessen Satz von Entwicklungskoeffizienten verwaltet. Die Dimensionen der Entwicklungskoeffizienten sind in Objekten der Klasse *Tdim* gespeichert. Die Klasse *mctdhBasis* enthält alle Objekte, die dem Management der MCTDH-Basis dienen.

Zur Demostration der API wird im folgenden ein Python-Skript vorgestellt, mithilfe dessen Dimensionsinformationen einer Multilayer-MCTDH-Wellenfunktion berechnet werden können. Zuerst wird das Skript vorgestellt und anschließend wird es abschnittsweise erklärt.

```
import mctdh

config = mctdh.controlParameters()
config.initialize('mctdh.config')
basis = mctdh.MctdhBasis()
basis.initialize('basis.txt', config)

maxNodes = basis.NmctdhNodes()

nodes_spf = []
for i in range(maxNodes):
    node = basis.MCTDHnode(i)
    tdim = node.t_dim()
    nodes_spf[i] = tdim.GetnTensor()

primitivB = {i: basis.MCTDHnode(i).t_dim().active(0) for i in \
             range(maxNodes) if \
             basis.MCTDHnode(i).Bottomlayer() == True}
NCoefBottomNode = {}
for key in primitivB:
    NCoefBottomNode[key] = primitivB[key] * nodes_spf[key]
NCoefBottom = sum([l_[1] for l_ in NCoefBottomNode.items()])

print NCoefBottom

NCoefTopNode = 0
for i in range(maxNodes):
    if basis.MCTDHnode(i).Toplayer() == True:
        children = basis.MCTDHnode(i).NChildren()
        for j in range(children):
            NCoefTopNode *= basis.MCTDHnode(i).down(j).t_dim().GetnTensor()
            NCoefTopNode *= basis.MCTDHnode(i).t_dim().GetnTensor()

print NCoefTopNode
```

```
remnantNodeList = []
remnant = 0
for i in range(maxNodes):
    if basis.MCTDHnode(i).Toplayer() == False and \
       basis.MCTDHnode(i).Bottomlayer() == False:
        children = basis.MCTDHnode(i).NChildren()
        parent = basis.MCTDHnode(i).t_dim().GetnTensor()
        for j in range(children):
            remnant *= basis.MCTDHnode(i).down(j).t_dim().GetnTensor()
        remnant *= parent
        remnantNodeList.append(remnant)
        remnant = 0
NCoefRemnant = sum(remnantNodeList)

print NCoefRemnant
```

Die in Abbildung 4.1.1 dargestellten Klassen werden mit folgenden Befehl in Python importiert:

```
import mctdh
```

Objekte der Klassen *ControlParameters* und *mctdhBasis* werden instanziert durch:

```
config = mctdh.controlParameters()
basis = mctdh.MctdhBasis()
```

Die erstellten Objekte werden mittels

```
config.initialize('mctdh.config')
basis.initialize('basis.txt', config)
```

initialisiert, wobei *mctdh.config* und *basis.txt* Dateinamen von Dateien sind, welche die Genauigkeitsparameter, bzw. die Basis-Definition enthalten.

Die Anzahl der Knoten im MCTDH-Baum wird mit dem folgenden Befehl bestimmt:

```
maxNodes = basis.NmctdhNodes()
```

Danach wird die Anzahl der Tensoren für jeden Knoten bestimmt. Die entsprechenden Dimensionen sind Member der Klasse *Tdim*, welche in den repräsentativen Knotenobjekten gespeichert sind. Sie werden ermittelt und in dem Dictionary *nodes_spf* gespeichert:

```
for i in range(maxNodes):
    node = basis.MCTDHnode(i)
    tdim = node.t_dim()
    nodes_spf[i] = tdim.GetnTensor()
```

Eine Liste der Größen aller primitiven Basissätze wird durch die folgenden Befehle erzeugt:

```
primitivB = {i: basis.MCTDHnode(i).t_dim().active(0) for i in \
             range(maxNodes) if \
             basis.MCTDHnode(i).Bottomlayer() == True}
```

Daraufhin wird die Anzahl aller Entwicklungskoeffizienten der untersten Lage berechnet durch

```
for key in primitivB:
    NCoefBottomNode[key] = primitivB[key] * nodes_spf[key]
NCoefBottom = sum([l_[1] for l_ in NCoefBottomNode.items()])
```

Die Anzahl der Entwicklungskoeffizienten der obersten Lage wird durch den folgenden Code-Abschnitt berechnet:

```
for i in range(maxNodes):
    if basis.MCTDHnode(i).Toplayer() == True:
        children = basis.MCTDHnode(i).NChildren()
        for j in range(children):
            NCoefTopNode *= basis.MCTDHnode(i).down(j).t_dim().GetnTensor()
    NCoefTopNode *= basis.MCTDHnode(i).t_dim().GetnTensor()
```

Im letzten Code-Block wird die Summe der Anzahl der Entwicklungskoeffizienten aus den übrigen Lagen gebildet:

```
for i in range(maxNodes):
    if basis.MCTDHnode(i).Toplayer() == False and \
       basis.MCTDHnode(i).Bottomlayer() == False:
        children = basis.MCTDHnode(i).NChildren()
        parent = basis.MCTDHnode(i).t_dim().GetnTensor()
        for j in range(children):
            remnant *= basis.MCTDHnode(i).down(j).t_dim().GetnTensor()
        remnant *= parent
        remnantNodeList.append(remnant)
        remnant = 0
NCoeffRemnant = sum(remnantNodeList)
```

Schließlich wird die Anzahl aller Entwicklungskoeffizienten der unteren Knoten, des oberen Knoten und der restlichen Knoten berechnet und ausgegeben:

```
print NCoeffBottom
print NCoeffTopNode
print NCoeffRemnant
```

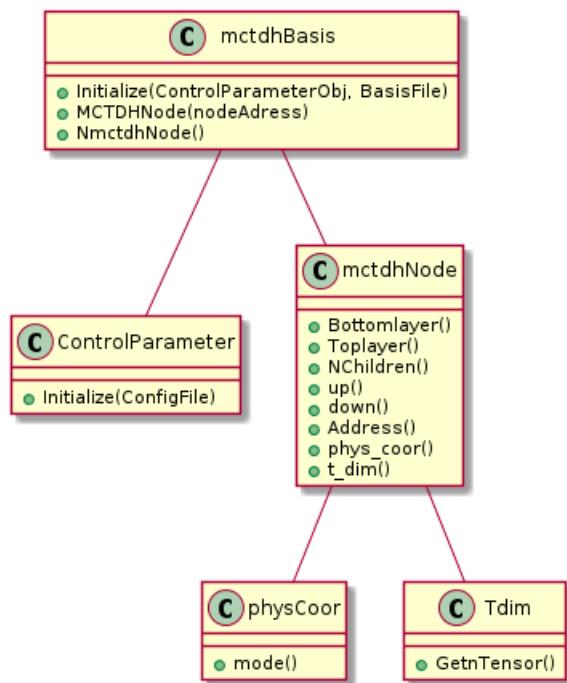


Abbildung 4.1.1: Klassendiagramm der Programmierschnittstelle (API). Die Rechtecke repräsentieren jeweils eine Klasse. Im oberen Bereich des Rechtecks befindet sich der Name der Klasse und im unteren Bereich eine Liste aller Memberfunktionen. Für eine detaillierte Beschreibung siehe Text.

4.2 Graphische Benutzeroberfläche für MCTDH

Zur Demonstration der graphischen Benutzeroberfläche wird im folgenden das Vorgehen zum Starten einer MCTDH Simulation vorgeführt. Als Beispiel dient die Realzeitpropagation eines Wellenpakets auf dem ersten elektronisch angeregten Zustand (S_1) von NOCl.

Beim Start der GUI erscheint das Hauptfenster (siehe Abbildung 4.2.1). In Liste **1** sind alle vorhandenen Projekte aufgeführt. Nach Auswahl eines Projekts erscheinen in Liste **2** alle dem Projekt zugeordneten Rechnungen. Durch die Schaltflächen „+“ und „-“ können Projekte sowie Rechnungen jeweils hinzugefügt bzw. entfernt werden. Im Reiter „File“ (**3**) befinden sich zusätzliche Optionen zur Projektverwaltung (siehe Abbildung 4.2.2). Durch Klicken der Schaltfläche **5** wird ein neues Projekt erstellt. Externe Projekte werden durch die Schaltfläche **6** geladen. Betätigung der Schaltfläche **7** beendet die Benutzeroberfläche.

Zum Starten einer neuen Rechnung wird zuerst der Reiter „Project“ (Abbildung 4.2.1 **4**) ausgewählt. Zwei neue Buttons erscheinen (Abbildung 4.2.3). Durch Klicken von **8** erscheint ein neues Fenster mit dem Namen „MCTDH calculation“ (siehe Abbildung 4.2.4). In diesem Fenster wird die neue MCTDH Rechnung spezifiziert.

In dem Eingabefenster „MCTDH calculation“ kann im Feld **10** der Name der Rechnung angegeben werden. Sollte beim Speichern (Abbildung 4.2.4 **18**) des MCTDH-Baums und der Einstellungsparameter kein Name angegeben sein, wird der Benutzer aufgefordert einen Name für die Rechnung zu wählen. Beim Speichern der Rechnung wird überprüft, ob eine Rechnung mit dem ausgewählten Namen existiert, um ein unabsichtliches Überschreiben zu verhindern. In Liste **11** wird der entsprechende Summe-von-Produkten Operator für das gegebene System ausgewählt. Über die Knöpfe „on“ und „off“ (**12**) wird gesteuert, ob eine Potentialflächenauswertung mittels CDVR erfolgt. Ist der „off“ Knopf aktiviert, werden keine Potentiale in der Liste **13** angezeigt. Bei der Standardeinstellung der GUI ist der Knopf „on“ aktiviert und es können Potentiale durch Klicken auf die Elemente der Liste **13** ausgewählt werden. Im vorliegenden Beispiel wird die Option „off“ ausgewählt. Die Erweiterung der Summe-von-Produkten Operatoren und Potentialflächen kann schematisch erfolgen.

Mithilfe der MCTDH GUI können vier verschiedene Operationen durchgeführt werden (**14**): „real-time propagation“, „imaginary-time propagation“, „Eigenstate calculation“ und „Thermal flux eigenstate calculation“. Hier wird die Option „real-time propagation“

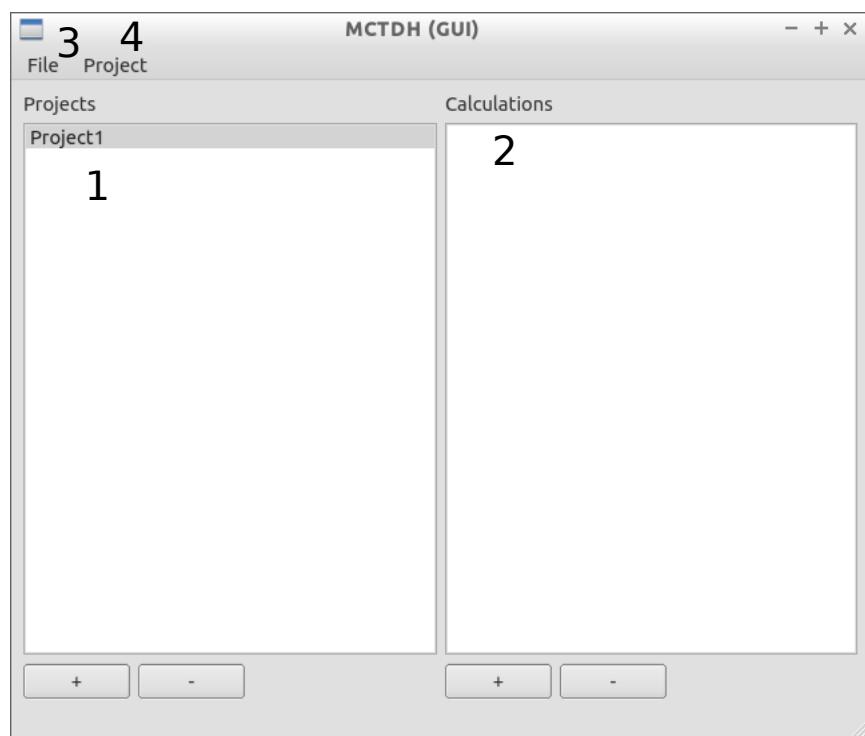


Abbildung 4.2.1: Hauptfenster der MCTDH-GUI. Es zeigt aktuell vorhandene Projekte (1), sowie zugehörige Rechnungen (2).

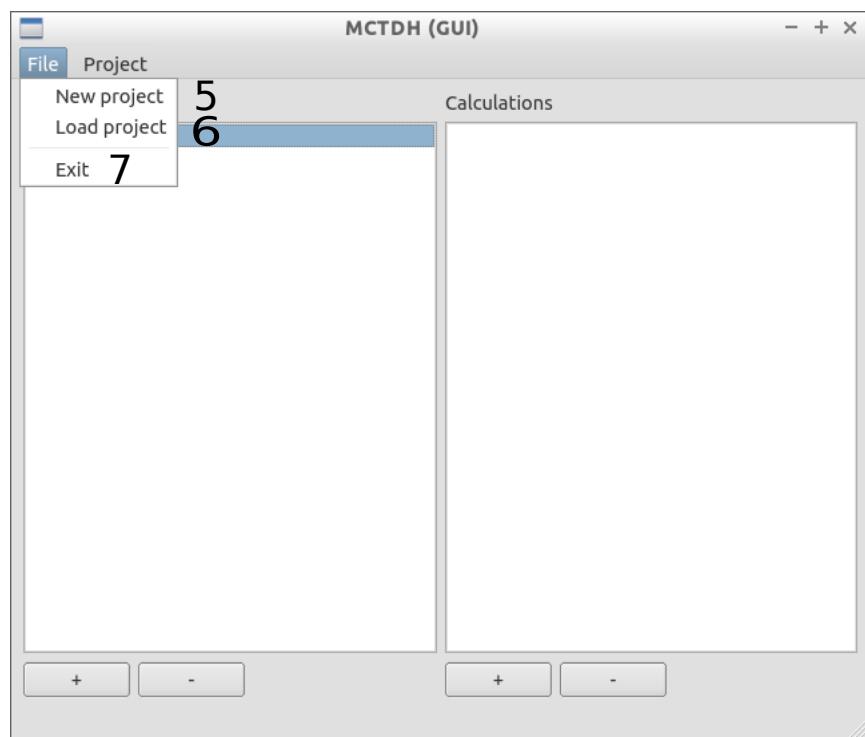


Abbildung 4.2.2: Hauptfenster der MCTDH-GUI. Nach Betätigung des Reiters „File“ stehen Optionen zur Projektverwaltung, sowie zum Beenden der GUI zur Verfügung.

4 Ergebnisse

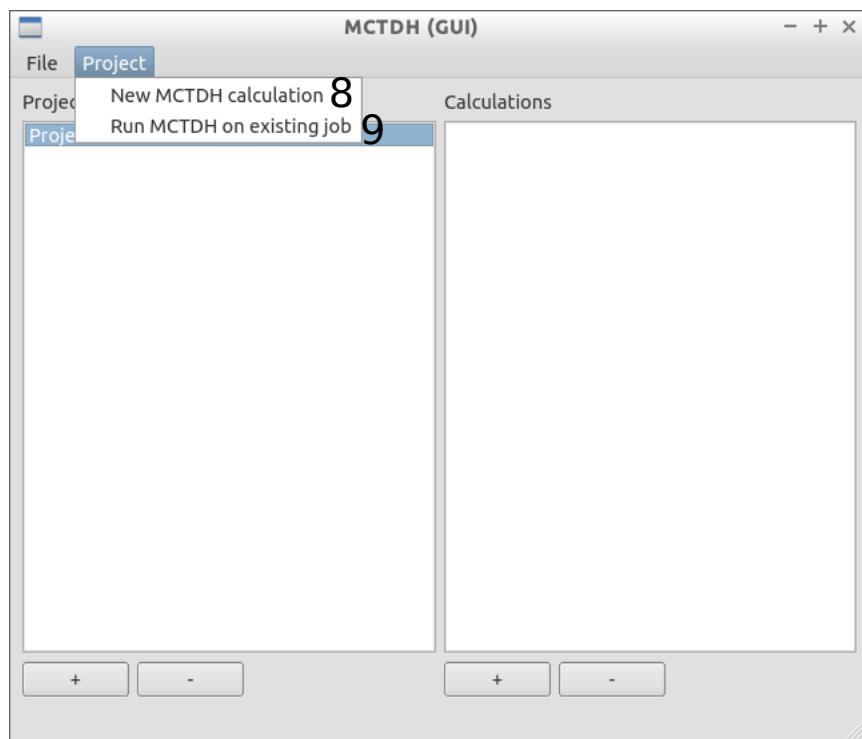


Abbildung 4.2.3: Hauptfenster der MCTDH-GUI nach Betätigung des Reiters „Project”.
Es stehen Optionen zum Erstellen und Starten einer MCTDH-Rechnung zur Verfügung.

ausgewählt. Des Weiteren werden unter **15** Steuerungsparameter für den Integrator ausgewählt. Dazu gehören die Anfangszeit, die Endzeit, der initiale Zeitschritt und (für die Operationen „Eigenstate calculation“ und „Thermal flux eigenstate calculation“) die Anzahl an Iterationen. Alle Parameter werden automatisch geladen, indem ein System aus Liste **11** ausgewählt wird oder eine Eingabe-Datei durch die „Load“ Schaltfläche **19** eingelesen wird. Der MCTDH-Baum, welcher die aktuell gewählte Basis repräsentiert, wird unter **16** als Listendiagramm angezeigt. Hier kann durch Doppelklick die Anzahl der SPFs, der primitiven Basis und der Moden verändert werden. Zusätzlich wird der MCTDH-Baum in Form eines Diagramms in **17** angezeigt. Die Schaltfläche „Cancel“ **21** beendet das Eingabefenster und die Schaltfläche „Start calculation“ (**20**) beginnt die MCTDH Simulation. Alternativ kann die MCTDH Simulation auch aus dem Hauptfenster gestartet werden, indem Schaltfläche „Run MCTDH on existing job“ (**9**) betätigt wird (siehe Abbildung [4.2.3](#)).

In Abbildung [4.2.5](#) wurde das System NOCl mit sinnvollen Eingabeparametern geladen, indem auf „NOCl“ aus Liste **11** geklickt wurde.

4 Ergebnisse

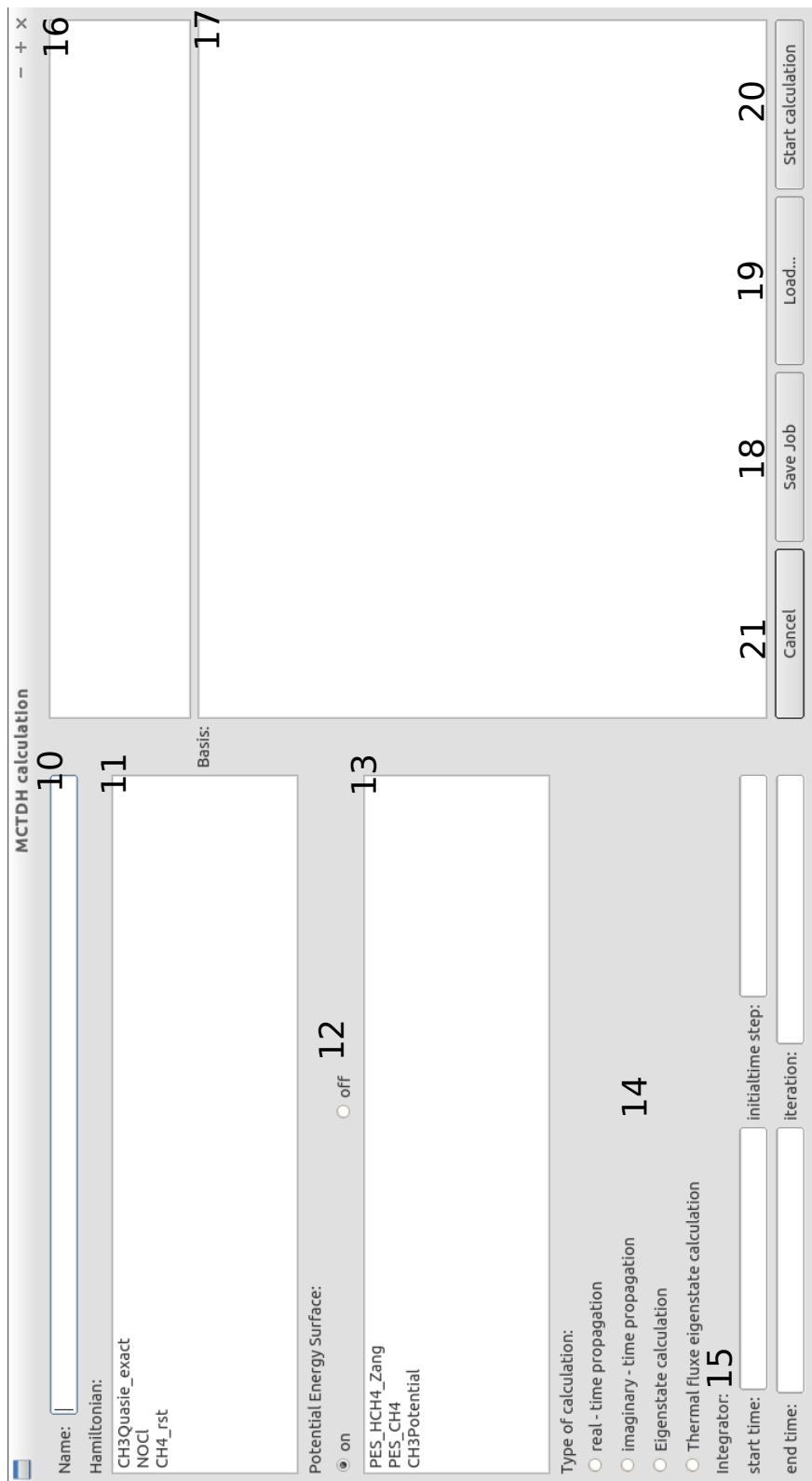


Abbildung 4.2.4: Spezifikation einer neuen MCTDH-Rechnung durch die MCTDH-GUI.

4 Ergebnisse

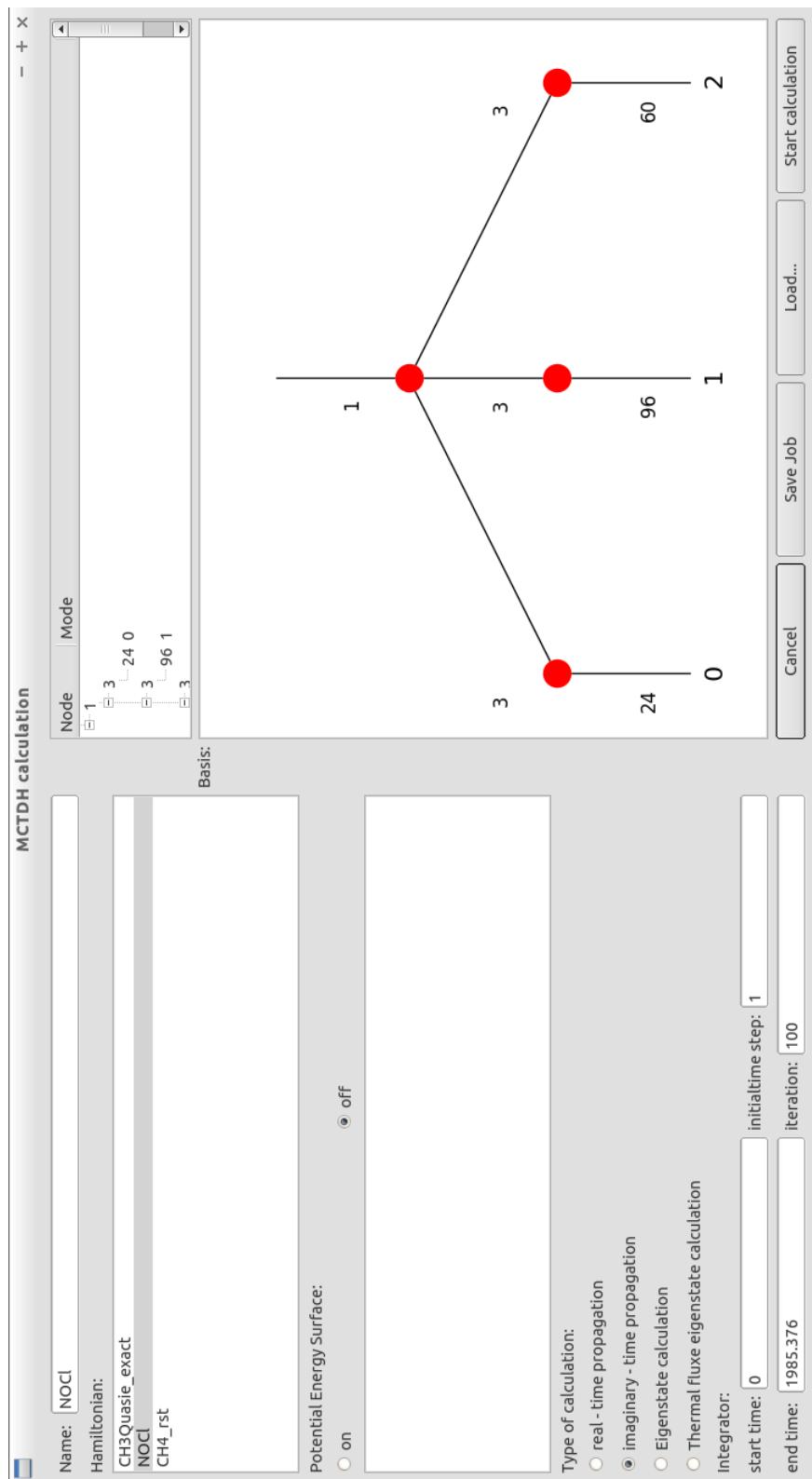


Abbildung 4.2.5: Eingabefenster der MCTDH-GUI am Beispiel der Photodissoziation von NOCl.

todo: 12. Juni 2018 - 17:08 Uhr

5 Zusammenfassung und Ausblick

In dieser Arbeit wurde eine Pythonschnittstelle für den in C++ verfassten MCTDH-Code erstellt, welche es erlaubt Klassen in Python-Skripten zu nutzen, die der Verwaltung der MCTDH-Basis dienen. In Zukunft kann die Schnittstelle um weitere Klassen des MCTDH-Codes ergänzt werden, sodass sie vielseitig eingesetzt werden kann.

Eine graphische Benutzeroberfläche wurde erstellt, welche es Nutzern ohne Programmierkenntnisse erlaubt MCTDH-Rechnungen zu starten. Im Rahmen dieser Arbeit wurde in der GUI eine Auswahl von Summe-von-Produkten Operatoren, sowie von *ab initio* Potentialflächen bereitgestellt. Diese Auswahl kann in Zukunft erweitert werden. Zudem wäre eine Funktion zum Erstellen von benutzerdefinierten Operatoren mittels Drag&Drop hilfreich. Die Struktur der MCTDH-Basis wird aktuell über externe Dateien eingelesen; auch hier wäre es sinnvoll eine Funktion zum Erstellen und Modifizieren der Multilayer-MCTDH Basisstruktur mittels Drag&Drop bereitzustellen.

Die Funktionen der GUI konzentriert sich bisher auf das Erstellen und Verwalten von projektbezogenen Rechnungen. Es wäre nützlich die Funktionen auf das Verfolgen und Visualisieren laufender oder beendeter Rechnungen auszuweiten. Dazu ist es sinnvoll für die Ausgabe des C++ MCTDH-Programms ein standardisiertes Protokollformat zu entwickeln, welches von externen Programmen eingelesen und interpretiert werden kann. Somit wäre beispielsweise ein Einbinden von Smartphones und Tabletts denkbar, um laufende Rechnungen unkompliziert und leicht verfolgen zu können.

Anhang

dialogC.py:

```
from PyQt4 import QtCore, QtGui, uic
from PyQt4 import *
import sys, os
from widgetA import WidgetA

base, form = uic.loadUiType("dialogNewName.ui")

class DialogC(base, form):
    def __init__(self, parent=None):
        super(base, self).__init__(parent)
        self.setupUi(self)

        self._FolderName = ""
        self._cancel = None

        self.uiWarning.setText('')
        self.uiFolderName.setText("Give name for new Folder!")
        self.uiFolderName.selectAll()
        self.uiFolderName.textChanged.connect(self.change0)
        self.uiSaveBu.clicked.connect(self.save)
        self.uiCancelBu.clicked.connect(self.esc)

    def setWarning(self, warningUi):
        print warningUi
        self.uiWarning.setText(warningUi)
```

```
def save(self):
    self._FolderName = self.uiFolderName.text()
    if ' ' not in self._FolderName:
        self.close()
    else:
        self.setWarning(str('Name contains whitespace'))
        print 'Name contains whitespace'

def esc(self):
    self._cancel = "Cancel"
    self.close()

def change0(self):
    self._FolderName = self.uiFolderName.text()
```

InputPro.py:

```
from PyQt4 import QtCore, QtGui
import sys, os

class ListAbstrModel(QtCore.QAbstractListModel):
    def __init__(self, data=[], parent=None):
        super(ListAbstrModel, self).__init__(parent)
        self.__data = data

    def rowCount(self, parent):
        return len(self.__data)

    def data(self, index, role):
        if role == QtCore.Qt.EditRole:
            row = index.row()
            return self.__data[row]

        if role == QtCore.Qt.DisplayRole:
```

```
row = index.row()
value = self._data[row]
return value

def flags(self, index):
    return QtCore.Qt.ItemIsEditable | QtCore.Qt.ItemIsEnabled | QtCore.Qt.ItemIsSelectable

def setData(self, index, value, role=QtCore.Qt.EditRole):
    if role == QtCore.Qt.EditRole:
        row = index.row()
        self._data[row] = str(value.toString())
        return True
    return False

class ListModel(ListAbstrModel):
    def __init__(self, data=[], parent=None):
        super(ListModel, self).__init__(data, parent)
        self._data = data
        self._dataBefore = list(data)
        self._messageBu = False

    def setData(self, index, value, role=QtCore.Qt.EditRole):
        if value not in self._data:
            if role == QtCore.Qt.EditRole:
                row = index.row()
                self._data[row] = str(value.toString())
                self.getValue()
                return True
        else:
            return False

    def getValue(self):
        matches = list(set(self._data).intersection(self._dataBefore))
        new = [l_ for l_ in self._data if l_ not in matches]
```

```
old = [l_ for l_ in self.__dataBefore if l_ not in matches]
print new, old
print os.getcwd()
self.__dataBefore = list(self.__data)
try:
    os.rename(old[0], new[0])
except OSError:
    raise

def removeRows(self, position, rows, parent=QtCore.QModelIndex()):
    self.beginRemoveRows(parent, position, position+rows-1)

    value = self.__data[position]
    self.showdialog(value)
    if self._messageBu == 'OK':
        for i in range(rows):
            value = self.__data[position]
            self.__data.remove(value)
        self.endRemoveRows()
        return True
    else:
        self.endRemoveRows()
        return False

def showdialog(self, value):
    msg = QtGui.QMessageBox()
    msg.setIcon(QtGui.QMessageBox.Warning)

    msg.setText("Are sure you want to delete Folder %s?" %value)
    msg.setStandardButtons(QtGui.QMessageBox.Ok | QtGui.QMessageBox.Cancel)

    msg.buttonClicked.connect(self.msgbtn)
    msg.exec_()
```

```
def msgbtn(self, i):
    self._messageBu = str(i.text())

class ListModel2(ListModel):
    def __init__(self, project, data=[], parent=None):
        super(ListModel2, self).__init__(data, parent)
        self.__data = data
        self.__dataBefore = list(data)
        self._changePath = os.getcwd() + '/' + project

    def setData(self, index, value, role=QtCore.Qt.EditRole):
        if value not in self.__data:
            if role == QtCore.Qt.EditRole:
                row = index.row()
                self.__data[row] = str(value.toString())
                self.getValue(row)
                return True
            else:
                return False

    def getValue(self, row):
        matches = list(set(self.__data).intersection(self.__dataBefore))
        new = [l_ for l_ in self.__data if l_ not in matches]
        old = [l_ for l_ in self.__dataBefore if l_ not in matches]
        self.__dataBefore = list(self.__data)
        os.chdir(self._changePath)
        try:
            os.rename(old[0], new[0])
        except Exception:
            self.showdialog(row)

    def removeRows(self, position, rows, parent=QtCore.QModelIndex()):
        self.beginRemoveRows(parent, position, position+rows-1)
```

```
        value = self._data[position]
        for i in range(rows):
            value = self._data[position]
            self._data.remove(value)
        self.endRemoveRows()
        return True
```

InputTree.py:

```
#####
#      Title: PyQt4 Model View Tutorial Part 04
#      Author: Yasin Uludag
#      Date: 2011
#      Availability: https://www.youtube.com/watch?v=pr1M3mP7qfI
#####

from PyQt4 import QtCore, QtGui, uic
import sys
from Node import Node, BottomNode, Tree

class SceneGraphModel(QtCore.QAbstractItemModel):
    def __init__(self, root, parent=None):
        super(SceneGraphModel, self).__init__(parent)
        self._rootNode = root
        self._child = None
        self._childIndex = QtCore.QModelIndex()
        self._dictNodes = {}

    def parent(self, index):
        node = index.internalPointer()
        parentNode = node.parent()

        if parentNode == self._rootNode:
            return QtCore.QModelIndex()
        return self.createIndex(parentNode.row(), 0, parentNode)
```

```
def index(self, row, column, parent):  
  
    childItem = parentNode.child(row)  
  
    if childItem:  
        return self.createIndex(row, column, childItem)  
    else:  
        return QtCore.QModelIndex()  
  
def rowCount(self, parent):  
    if not parent.isValid():  
        parentNode = self._rootNode  
    else:  
        parentNode = parent.internalPointer()  
  
    return parentNode.childcount()  
  
def columnCount(self, parent):  
    return 2  
  
  
def flags(self, index):  
    return QtCore.Qt.ItemIsEnabled | QtCore.Qt.ItemIsSelectable | \  
          QtCore.Qt.ItemIsEditable | QtCore.Qt.ItemIsDragEnabled | \  
          QtCore.Qt.ItemIsDropEnabled  
  
def headerData(self, section, orientation, role):  
    if role == QtCore.Qt.DisplayRole:  
        if section == 0:  
            return "Node"  
        else:  
            return "Mode"
```

```
def data(self, index, role):

    if not index.isValid():
        return None

    node = index.internalPointer()

    if role == QtCore.Qt.DisplayRole or role == QtCore.Qt.EditRole:
        if index.column() == 0:
            return node.name()
        else:
            if node.typeInfo() == "Bottom":
                return node._physcoor

    if role == QtCore.Qt.DecorationRole:
        if index.column() == 0:

            typeInfo = node.typeInfo()

def setData(self, index, value, role = QtCore.Qt.EditRole):
    if index.isValid():

        if role == QtCore.Qt.EditRole:
            node = index.internalPointer()
            if index.column() == 0:
                node.setName(str(value.toString()))
                return True
            elif index.column() == 1:
                node.setPhyscoor(str(value.toString()))
                return True
            return False

def getNode(self, index):
    if index.isValid():
```

```
node = index.internalPointer()
if node:
    return node
return self._rootNode

def getNode2(self, index):
    return self._rootNode

def insertRows(self, position, rows, parent=QtCore.QModelIndex()):
    parentNode = self.getNode(parent)
    oldNode = self._child
    if oldNode.typeInfo() == 'NODE':
        self.addNode(oldNode.name(), oldNode.name(), None)
    if oldNode.typeInfo() == 'Bottom':
        self.addBottomNode(oldNode.name(), oldNode.name(), None, oldNode.physcoor)
    self.copyNode(self._child)

    self.beginInsertRows(parent, position, position+rows-1)

    for row in range(rows):
        success = parentNode.insertChild(position, self._dictNodes[oldNode.name()])

    self.endInsertRows()

    return success

def removeRows(self, position, rows, parent=QtCore.QModelIndex()):
    parentNode = self.getNode(parent)

    self.beginRemoveRows(parent, position, position+rows-1)

    for row in range(rows):
        success = parentNode.removeChild(position)

    self.endRemoveRows()
```

```
    self.endRemoveRows()

    return success

#####Drag and Drop#####

def supportedDropActions(self):
    return QtCore.Qt.CopyAction | QtCore.Qt.MoveAction

def mimeTypes(self):
    types = QtCore.QStringList()
    types.append('text/plain')
    return types

def mimeData(self, index):
    rc = ''
    self._child = index[0].internalPointer()
    self._childIndex = index[0]
    mimeData = QtCore.QMimeData()
    mimeData.setText(rc)
    return mimeData

def dropMimeData(self, data, action, row, column, parentIndex):
    if action == QtCore.Qt.IgnoreAction:
        return True

    self.insertRows(0,1, parentIndex)
    index = self._childIndex
    self.removeRows(0,1, index)
    print self._rootNode.child(0)
    return True

def copyNode(self, oldNode):
```

```
children = oldNode.childAll()
if children:
    for oldchild in children:
        if oldchild.typeInfo() == "NODE":
            self.addNode(oldchild.name(), oldchild.name(), self._dictNodes[oldchild])
        if oldchild.typeInfo() == "Bottom":
            self.addBottomNode(oldchild.name(), oldchild.name(), self._dictNodes[oldchild])
    self.copyNode(oldchild)

def addNode(self, obj, SPF, parent):
    self._dictNodes[obj] = Node(SPF, parent)

def addBottomNode(self, obj, SPF, parent, physcoor):
    self._dictNodes[obj] = BottomNode(SPF, parent, physcoor)
```

LogicalNodes.py:

```
from ModelTree import ModelTree
import networkx as nx
from networkx.drawing.nx_agraph import write_dot, graphviz_layout
import matplotlib.pyplot as plt

class LogicalNodes():
    def __init__(self, layer_matr, config_file, sys_file):
        self.G = nx.DiGraph()
        self.layer_matr = layer_matr
        self.ModelTree = ModelTree(config_file, sys_file)
        self.label_mode = self.ModelTree.label_mode
        self.nodes_spf = self.ModelTree.nodes_spf
        self.Networkx()
        self.augDiGraph("SPF", self.nodes_spf)
        self.augDiGraph("Mode", self.label_mode)

    def Networkx(self):
        for c_ in self.layer_matr:
```

```
for b_ in c_:
    self.G.add_node(b_)
    l = len(c_)
    for index, b_ in enumerate(c_):
        if index < (l-1):
            self.G.add_edge(b_, c_[index + 1], weight=1)

def augDiGraph(self, str_kind, str_dict):
    for key, ele_ in str_dict.items():
        self.G.nodes[key][str_kind] = ele
```

main.py:

```
from PyQt4 import QtGui, QtCore, uic
import sys, os, shutil
from widgetA import WidgetA
from InputPro import ListModel, ListModel2
from dialogC import DialogC
from dialogD import DialogD

base, form = uic.loadUiType("main.ui")

class Main(base, form):
    def __init__(self, parent=None):
        super(base, self).__init__(parent)
        self.setupUi(self)

        self._HamiltonianDir = os.getcwd() + '/' + 'Hamiltonians'
        os.chdir('Projects')
        self._startingPath = os.getcwd()
        self._ProjectName = None
        self._path2 = None
        self._dir_list = None
        self._proContent = []
```

```
self._model1 = None
self._model2 = None
self._itemIndex1 = None
self._itemIndex2 = None

self.getdirs()
self._WidgetA = WidgetA(self)
self._WidgetA._HamiltonianDir = self._HamiltonianDir
self._WidgetA._startingPath = self._startingPath

self.setList()
self.uiNew.triggered.connect(self.openA)
self.uiLoad.triggered.connect(self.openB)
self.uiMCTDcalc.triggered.connect(self.openD)
self.uiMCTDHexisting.triggered.connect(self.runJob)
self.uiPlusBu.clicked.connect(self.openA)
self.uiPlusBu2.clicked.connect(self.openO)
self.uiMinusBu.clicked.connect(self.removeA)
self.uiMinusBu2.clicked.connect(self.removeO)

self.uiProjects.clicked.connect(self.on_item_select)
self.uiProjects.customContextMenuRequested.connect(self.openMenu)

self.setList2()
self.uiSessions.clicked.connect(self.on_item_select0)
self.uiSessions.customContextMenuRequested.connect(self.openMenu0)

self._dialogD = DialogD()

self._messageBu = None

def openMenu0(self, position):
    """Context menu"""

```

```
menu = QtGui.QMenu()
renameAction = menu.addAction("Rename")
action = menu.exec_(self.uiSessions.mapToGlobal(position))
if action == renameAction:
    self.uiSessions.edit(self._itemIndex2)

def openMenu(self, position):
    """Context menu"""
    menu = QtGui.QMenu()
    renameAction = menu.addAction("Rename")
    action = menu.exec_(self.uiProjects.mapToGlobal(position))
    if action == renameAction:
        self.uiProjects.edit(self._itemIndex1)

def remove0(self):
    """Removes Rows from ListModel2()"""
    rowNum = self._itemIndex2.row()
    session = str(self._itemIndex2.data().toString())
    self.showdialog(session)
    if 'OK' in self._messageBu:
        self._model2.removeRows(rowNum,1, self._itemIndex2)

    shutil.rmtree(self._startingPath+'/'+self._ProjectName+'/'+session)

def removeA(self):
    """Removes Rows from ListModel()"""
    rowNum = self._itemIndex1.row()
    key = str(self._itemIndex1.data().toString())
    startingpath2 = self._startingPath + '/'
    delFolder = startingpath2 + key

    self._model1.removeRows(rowNum,1, self._itemIndex1)

    if self._model1._messageBu == 'OK' and delFolder != self._startingPath and
```

```
shutil.rmtree(self._startingPath+'/'+key)

maxRow = self._model2.rowCount(self._itemIndex2)
if maxRow != 0:
    self._model2.removeRows(0, maxRow)

def showdialog(self, value):
    msg = QtGui.QMessageBox()
    msg.setIcon(QtGui.QMessageBox.Warning)

    msg.setText("Are sure you want to delete Folder %s?" %value)
    msg.setStandardButtons(QtGui.QMessageBox.Ok| QtGui.QMessageBox.Cancel)

    msg.buttonClicked.connect(self.msgbtn)
    msg.exec_()

def msgbtn(self, i):
    self._messageBu = str(i.text())

def getContent(self):
    os.chdir(self._startingPath)
    if os.path.exists(self._ProjectName):
        directories = os.walk('.'+'/'+self._ProjectName).next()[1]
        try:
            directories.remove('tmp')
        except ValueError:
            pass
        self._proContent = sorted(directories)
    else:
        print("path doesn't exists")

def on_item_select0(self, index):
    """clicked Event on Items belonging to ListModel2()"""
    os.chdir(self._startingPath)
```

```
projectFolder = str(self._itemIndex1.data().toString())
sessionFolder = str(index.data().toString())
self._itemIndex2 = index
self._ProjectName = projectFolder

os.chdir(self._ProjectName)
self._WidgetA._ProjectName = projectFolder
self._WidgetA.makedirs()
self._WidgetA.editSession(sessionFolder)
self._WidgetA.setSessionName(sessionFolder)
self._WidgetA.start()
self._WidgetA.removeContent()

SESfiles = os.walk(self._startingPath+'/'+self._ProjectName+'/'+sessionFolder)

if SESfiles:
    for f_ in SESfiles:
        if 'txt' in f_:
            self._WidgetA.fromSESToTMP(f_)

self.openC()
os.chdir('..')

def on_item_select(self, index):
    """Clicked Event on Items belonging to ListModel()"""
    self._itemIndex1 = index
    self._ProjectName = str(index.data().toString())
    self._WidgetA._ProjectName = self._ProjectName
    self._WidgetA._SessionName = None
    self._WidgetA.clearSession()
    self.getdirs()
    self.setList2()

def setList2(self):
    #####ListModelPES#####
    self.getContent()
    self._model2 = ListModel2(self._ProjectName, self._proContent)
```

```
self.uiSessions.setModel(self._model2)
indices = self.uiSessions.selectionModel().selectedIndexes()
if not indices:
    index = self._model2.index(0,0)
    self._itemIndex2 = index
    self.uiSessions.selectionModel().select(index, QtGui.QItemSelectionModel.Select)
    self._path2 = str(self._itemIndex2.data().toString())

def setList(self):
    #####ListModelPES#####
    self.getdirs()
    self._model1 = ListModel(self._dir_list)
    self.uiProjects.setModel(self._model1)
    indices = self.uiProjects.selectionModel().selectedIndexes()
    if not indices:
        index = self._model1.index(0,0)
        self._ProjectName = str(index.data().toString())
        self._WidgetA._ProjectName = self._ProjectName
        self._itemIndex1 = index
        self._itemProxyIndex1 = index
        self.uiProjects.selectionModel().select(index, QtGui.QItemSelectionModel.Select)

def getdirs(self):
    directories = os.walk(self._startingPath).next()[1]
    self._dir_list = [dirs for dirs in directories]
    self._dir_list = sorted(self._dir_list)

def open0(self, warn):
    if warn == False:
        warn = ''
    dialogC = DialogC()
    dialogC.setWarning(str(warn))
    dialogC.exec_()
    self._path2 = str(dialogC._FolderName)
```

```
print self._path2

if str(dialogC._cancel) != 'Cancel':
    path = self._startingPath + '/' + self._ProjectName + '/' + self._pat
    if not os.path.exists(path):
        try:
            os.chdir(self._startingPath+'/'+self._ProjectName)
            os.makedirs(self._path2)
            os.chdir(self._startingPath)
        except IOError as identifier:
            print (dir(identifier))
        self.getContent()
        self.setList2()
    else:
        print('Folder already exists!')
        self.open0('Folder already exists!')

def openA(self, warnings):
    if warnings == False:
        warnings = ''
    dialogC = DialogC()
    dialogC.setWarning(str(warnings))
    dialogC.exec_()
    self._ProjectName = str(dialogC._FolderName)

    if self._ProjectName != 'Cancel':
        path = self._startingPath + '/' + self._ProjectName

        if not os.path.exists(path):
            try:
                os.makedirs(path)
                os.makedirs(path+'/tmp')
            except IOError as identifier:
                print (identifier)
```

```
        self.setList()

    else:
        print('Folder already exists!')
        self.openA('Folder already exists!')

def openB(self):
    self._ProjectName = str(QtGui.QFileDialog.getExistingDirectory(self))
    self._ProjectName = self._ProjectName.split("/")[-1]
    self.getContent()
    self.setList2()

def openC(self):
    dialog = self._WidgetA
    dialog.exec_()
    self.setList2()

def openD(self):
    self._WidgetA.clearSession()
    self._WidgetA.setSessionName(None)
    self._WidgetA.removeContent()

    self.openC()

def Finder(self, path, app):
    fileList = os.walk(path).next()[2]
    return [f_ for f_ in fileList if app in f_][0]

def getRunInput(self):
    filePathList = []
    self.getContent()
    self._proContent
    for root, dirs, files in os.walk(self._startingPath+'/'+self._ProjectName):
        for name in files:
```

```
        filePathList.append(os.path.join(root, name))
filePathList = [p_ for p_ in filePathList if '.in' in p_]
filePathList = [f_.split('/')[-2:] for f_ in filePathList]
filePathList = [f_ for f_ in filePathList if f_[0] in self._proContent]
return ['/'.join(f_) for f_ in filePathList]

def runJob(self):
    '''Try QProcess from Qt'''
    dialog = self._dialogD
    self._dialogD._model = QtGui.QStandardItemModel(self._dialogD.uiRunList)
    self._dialogD.uiRunList.setModel(self._dialogD._model)
    self._dialogD.uiRunList.setEditTriggers(QtGui.QAbstractItemView.NoEditTriggers)
    inputList = self.getRunInput()

    for calc in inputList:
        item = QtGui.QStandardItem(calc)
        self._dialogD._model.appendRow(item)

    self._dialogD.uiRunList.clicked.connect(self.on_item_select2)
    dialog.exec_()

def on_item_select2(self, index):
    inputFile = self._startingPath+'/' +self._ProjectName+'/' +str(index.data())
    self.process = QtCore.QProcess()
    self.process.setProcessChannelMode(QtCore.QProcess.MergedChannels)
    self.process.readyReadStandardOutput.connect(self.mctdhOut)
    self.results(inputFile)

def results(self, inputFile):
    os.chdir('../Results')
```

```
self._WidgetA.genereInput(inputFile)
try:
    os.mkdir(self._WidgetA._mainfolder)
except OSError:
    pass
mctdh = '/home/piet/newRepo/QuantumDynamics/build/bin/mctdh'
self.process.start(mctdh+' '+inputFile)
os.chdir(self._startingPath)
self._dialogD.close()

def mctdhOut(self):
    output = str(self.process.readAllStandardOutput())
    print output

if __name__ == '__main__':
    app = QtGui.QApplication(sys.argv)
    app.setStyle("cleanlooks")
    wnd =Main()
    wnd.show()
    sys.exit(app.exec_())
```

ModelTree.py:

```
import mctdh
import sys

class ModelTree(object):
    def __init__(self, config_file='mctdh.config', sys_file='CH3g1.txt'):
        self.config_file = config_file
        self.sys_file = sys_file
        self.bottom_list = []
        self.mode_list = []
        self.layer_list = []
        self.layer_matr = []
```

```
self.lay_matr_mode = []
self.label_mode = {}
self.nodes_spf = {}

print self.config_file, 'from ModelTree'
print self.sys_file, 'from ModelTree'
self.config = mctdh.controlParameters()
self.config.initialize(self.config_file)
self.basis = mctdh.MctdhBasis()
self.basis.initialize(self.sys_file, self.config)
self.node = mctdh.MctdhNode()
self.phys = mctdh.PhysCoor()
self.tdim = mctdh.Tdim()

self.getLayerMatr()
self.getPhysCoord()
self.modeToGetLayer()
self.get_SPFs()

def getBottomlayer(self):
    """Get the bottom nodes"""
    for i in range(self.basis.NmctdhNodes()):
        self.node = self.basis.MCTDHnode(i)
        if self.node.Bottomlayer() == True:
            self.bottom_list.append(i)
    return self.bottom_list #List of the i-th bottom node

def getPhysCoord(self):
    """get the Modes of the pys. Coordinates"""
    for i in range(self.basis.NmctdhNodes()):
        self.node = self.basis.MCTDHnode(i)
        if self.node.Bottomlayer() == True:
            self.phys = self.node.phys_coor()
```

```
        self.mode_list.append(self.phys.mode()) #append Modes to list

    def nlayer(self, i):
        """Recursion function that goes from the i-th node up to the top layer of
        self.node = self.basis.MCTDHnode(i) #i-th Node
        self.layer_list.append(i)
        if self.node.Toplayer() == False:
            return self.nlayer(self.node.up().address())
        new_list = list(self.layer_list) #copy instead of reference
        del self.layer_list[:]
        return new_list

    def getLayerMatr(self):
        """Takes all bottom nodes and returns for each a list of the path to the
        self.layer_matr = [list(reversed(self.nlayer(b_))) for b_ in self.getBottomNodes()]

    def modeToGetLayer(self):
        """Combines layer_matr with mode_list"""
        self.lay_matr_mode = [l_ + [100 + i] for i,l_ in enumerate(self.layer_matr)]
        self.lay_matr_mode = [["Top"] + l_ for l_ in self.lay_matr_mode]
        #concatinates two lists
        self.label_mode = [100 + i for i in range(len(self.mode_list))]
        self.label_mode = dict(zip(self.label_mode, self.mode_list))

    def get_SPFs(self):
        """get the SPFs of each Node"""
        for i in range(self.basis.NmctdhNodes()):
            self.node = self.basis.MCTDHnode(i)
            self.tdim = self.node.t_dim()
            self.nodes_spf[i] = self.tdim.GetnTensor() #dict

        mode_spf = [self.basis.MCTDHnode(i).t_dim().active(0) for i in \
                    range(self.basis.NmctdhNodes()) if \
                    self.basis.MCTDHnode(i).Bottomlayer() == True ]
```

```
mode_spf_dict = dict(zip(self.label_mode, mode_spf))
for k, i in zip(self.label_mode.keys(), mode_spf):
    self.nodes_spf[k] = i
```

Node.py:

```
from LogicalNodes import LogicalNodes
from ModelTree import ModelTree
import numpy as np
import networkx as nx
import re, sys, os

class Parameters(object):
    def __init__(self):
        self._eps_general = None
        self._eps_1 = None
        self._eps_2 = None
        self._start = None
        self._end = None
        self._dt = None
        self._iteration = None
        self._hamiltonian = None
        self._potential = None
        self._job = None
        self._parameters = None
        self._treeData = None

class InPut(Parameters):
    def __init__(self, filename='example.in'):
        super(InPut, self).__init__()
        self._filename = filename
        self._filenameOld = None
        self._paradict = {}
        self._paralist = []
```

```
self._treelist = []
self._treeString = ''
self._commDict = {}
self.rmCommen()
self.readFile()

def readFile(self):
    self.getPara("mainfolder")
    self.getPara("Hamiltonian")
    self.getPara("Potential")
    self.getPara("job")
    self.getPara("start")
    self.getPara("end")
    self.getPara("dt")
    self.getPara("iteration")
    self.getPara("out")
    self.getPara2()
    self._paradict['para'] = self._paralist
    self._paradict['Comm'] = self._commDict
    self.getTree()

def file_len(self, fname):
    with open(fname) as f:
        for i, l in enumerate(f):
            pass
    return i+1

def getPara2(self):
    lineNumber = self.file_len(self._filename)
    with open(self._filename, "rb") as text:
        for line in text:
            if "parameters" in line:
                for i in range(lineNumber):
                    try:
```

```
para = text.next()
if bool(re.search(r'\d', para)):
    para = para.split()      #if para contains numbers
    self._paralist.append(para)
except StopIteration:
    pass

lastLine = [x for x in self._paralist[-1] if x != ',']
del self._paralist[-1]
self._paralist.append(lastLine)

def rmCommen(self):
    with open(self._filename, "r") as in_put:
        with open('new_InPut.in', 'wb') as output:
            i = 0
            for line in in_put:
                i += 1
                if '//' in line:
                    self._commDict[i-1] = line
                else:
                    output.write(line)

    self._filename = 'new_InPut.in'

def getPara(self, para):
    with open(self._filename, "rb") as text:
        for line in text:
            if para in line:
                try:
                    pos = line.index('=')      # get Index
                    self._paradict[para] = line[pos+1:].strip()  # removes wh
                except ValueError:
                    pass
```

```
def getTree(self):
    lineNumber = self._file_len(self._filename)
    with open(self._filename, "rb") as text:
        for line in text:
            if 'tree' in line:
                for i in range(lineNumber):
                    try:
                        tree = text.next()
                        if ']' in tree:
                            break
                        self._treelist.append(tree)
                    except StopIteration:
                        pass
    self._treeString = ''.join(self._treelist)

class OutPut(Parameters):
    def __init__(self, tree, paradict, sysFile, filename="InPut.in"):
        self._mainfolder = paradict['mainfolder']
        self._start = paradict['start']
        self._end = paradict['end']
        self._dt = paradict['dt']
        self._out = paradict['out']
        self._iteration = paradict['iteration']
        self._hamiltonian = paradict['Hamiltonian']
        try:
            self._potential = paradict['Potential']
        except KeyError:
            self._potential = 'no Potential'
        self._job = paradict['job']
        self._parameters = paradict['para']
        self._formated = self.formatparameter()
        self._treeData = tree._treeData
        self._filename = filename
        self._sysFile = sysFile
```

```
def savefile(self):
    with open(self._filename, "w") as text_file:
        text_file.write("{0}".format(self.bringAllTogether()))

def savefile2(self):
    with open(self._sysFile, "w") as text_file:
        text_file.write("{0}".format(self.bringTreePara()))

def formatparameter(self):
    output = ""
    A = self._parameters
    output = '\n'.join(['' .join(['{:4}'.format(item) for item in row]) for
    return output

def bringAllTogether(self):

    output = "mainfolder = " + self._mainfolder + "\n" \
"Hamiltonian = " + self._hamiltonian + "\n" \
"Potential = " + self._potential + "\n" \
"\n" \
"job = " + self._job + "\n" \
"\n" \
'integrator = { \n" \
"start = " + self._start + "\n" \
"end = " + self._end + "\n" \
"dt = " + self._dt + "\n" \
"iteration = " + self._iteration + "\n" \
"out = " + self._out + "\n" \
"} \n" \
"\n" \
"basis = \n" \
"\n" \
"tree = [ \n" \
```

```
+ self._treeData + \
"]\n" \
"\n" \
"\n" \
"parameters = [\n" \
+ self._formated + \
"]\n" \
"}"
return output

def bringTreePara(self):
    output = self._treeData + \
"\n" \
+ self._formated
return output

def __repr__(self):
    return self.bringAllTogether()

class OutPut2(object):
    def __init__(self, paradict, treeString, filename, pathTMP):
        self._parameters = paradict['para']
        self._formated = self.formatparameter()
        self._treeString = treeString
        self._sysFile = pathTMP + '/Load.txt'
        self._TMPmctdhConfig = pathTMP + '/mctdh.config'
        self._path = pathTMP

    def savefile(self):
        with open(self._TMPmctdhConfig, "w") as text_file:
            text_file.write("{0}".format(self.bringAllEPS()))

    def savefile2(self):
        with open(self._sysFile, "w") as text_file:
```

```
text_file.write("{0}".format(self.bringTreePara()))  
  
def bringAllEPS(self):  
    epsList = ['1E-6', '8E-5', '5E-5', '5E-5', '0', '0']  
    output = '\n'.join(epsList)  
    return output  
  
def bringTreePara(self):  
    output = self._treeString + \  
    "\n" \  
    + self._formated  
    return output  
  
def formatparameter(self):  
    output = ""  
    A = self._parameters  
    output = '\n'.join(['' .join(['{:4}'.format(item) for item in row]) for  
                      row in A])  
    return output  
  
  
class Tree(object):  
    def __init__(self, mctdhConfig, sys_file):  
        self._rootNode0 = Node("TOP")  
        self._dictNodes = {}  
  
        model = ModelTree(mctdhConfig, sys_file)  
        logical = LogicalNodes(model.lay_matr_mode, mctdhConfig, sys_file)  
        self._G = logical.G  
        self._elder = None  
        self.getElder()  
        self.addNode(self._elder, str(self._G.nodes[self._elder]['SPF']), self._r  
        self.readTree()
```

```
self._rootNode = self._dictNodes[self._elder]
self._treeData = self._dictNodes[self._elder].log()

def getElder(self):
    for ele_ in self._G.nodes():
        if self._G.pred[ele_] == {}:
            self._elder = ele_
    self._elder = self._G.successors(self._elder).next()

def readTree(self):
    for suc_ in nx.bfs_successors(self._G, self._elder):
        for brothers in suc_[1]:
            if 'Mode' in self._G.nodes[brothers].keys():
                self.addBottomNode(brothers, str(self._G.nodes[brothers]['SPF']))
            else:
                self.addNode(brothers, str(self._G.nodes[brothers]['SPF']), s)

def setRootNode(self, rootNode):
    self._rootNode = rootNode
    self.setLog()

def setLog(self):
    self._treeData = self._rootNode.log()

def addNode(self, obj, SPF, parent):
    self._dictNodes[obj] = Node(SPF, parent)

def addBottomNode(self, obj, SPF, parent, physcoor):
    self._dictNodes[obj] = BottomNode(SPF, parent, physcoor)
```

```
#####
#   Title: PyQt4 Model View Tutorial Part 04
#   Author: Yasin Uludag
#   Date: 2011
#   Availability: https://www.youtube.com/watch?v=pr1M3mP7qfI
#####

class Node(object):
    def __init__(self, name, parent=None):
        self._name = name
        self._children = []
        self._parent = parent

        if parent is not None:
            parent.addChild(self)

    def addChild(self, child):
        self._children.append(child)

    def insertChild(self, position, child):
        if position < 0 or position > len(self._children):
            return False

        self._children.insert(position, child)
        child._parent = self
        return True

    def removeChild(self, position):
        if position < 0 or position > len(self._children):
            return False

    try:
        child = self._children.pop(position)
```

```
        child._parent = None
        return True
    except IndexError as e:
        print e.message, ': from Node, 237'
        return False

    def name(self):
        return self._name

    def setName(self, name):
        self._name = name

    def child(self, row):
        return self._children[row]

    def childAll(self):
        return self._children

    def childcount(self):
        return len(self._children)

    def parent(self):
        return self._parent

    def row(self):
        if self._parent is not None:
            return self._parent._children.index(self)

    def log(self, tabLevel=-1):
        output = ""
        tabLevel += 1

        for i in range(tabLevel):
            output += "    "
```

```
if self.childcount() == 0:  
    output += self._name + " " + str(self.childcount()) + "\n"  
else:  
    output += self._name + " -" + str(self.childcount()) + "\n"  
  
for child in self._children:  
    output += child.log(tabLevel)  
  
tabLevel -= 1  
  
return output  
  
def __repr__(self):  
    return self.log()  
  
def typeInfo(self):  
    return "NODE"  
  
class BottomNode(Node):  
    def __init__(self, name, parent, physcoor):  
        super(BottomNode, self).__init__(name, parent)  
        self._physcoor = physcoor  
  
    def typeInfo(self):  
        return "Bottom"  
  
    def physcoor(self):  
        return self._physcoor  
  
    def setPhyscoor(self, physcoor):  
        self._physcoor = physcoor  
  
    def log(self, tabLevel=-1):  
        output = ""
```

```
tabLevel += 1

for i in range(tabLevel):
    output += "    "
if self.childcount() == 0:
    output += self._name + " " + str(self.childcount()) + " " + self.-
else:
    output += self._name + " -" + str(self.childcount()) + "\n"

for child in self._children:
    output += child.log(tabLevel)

tabLevel -= 1

return output
```

View.py:

```
import networkx as nx
from networkx.drawing.nx_agraph import write_dot, graphviz_layout
import matplotlib.pyplot as plt

import sys
from PyQt4 import QtGui

class View(object):
    def __init__(self, label_mode, nodes_spf):
        self.label_mode = label_mode
        self.nodes_spf = nodes_spf
        self._G = nx.DiGraph()

    def Display(self, G):
        top = ['Top']
        rest_nodes = [l_ for l_ in self.nodes_spf.keys() if l_ not in self.label_
pos = graphviz_layout(G, prog='dot')
```

```
nx.draw(G, pos, with_labels=False, arrows=False, node_color='w')
nx.draw_networkx_nodes(G, pos, nodelist=top, node_color='w', alpha=1)
nx.draw_networkx_nodes(G, pos, nodelist=rest_nodes, node_color='r', alpha=1)

pos_lower = {}
x_off = 0
y_off = -5
for k, v in pos.items():
    pos_lower[k] = (v[0] + x_off, v[1] + y_off)

nx.draw_networkx_labels(G, pos_lower, self.label_mode, font_size=16)

pos_higher = {}
x_off = -7
y_off = 27
for k, v in pos.items():
    pos_higher[k] = (v[0] + x_off, v[1] + y_off)

nx.draw_networkx_labels(G, pos_higher, self.nodes_spf, font_size=12)
plt.savefig('nx_test.png')
plt.clf()

for key, ele_ in self.nodes_spf.items():
    G.nodes[key]["SPF"] = ele_
for key, ele_ in self.label_mode.items():
    G.nodes[key]["Mode"] = ele_
self._G = G

widgetA.py:

from PyQt4 import QtCore, QtGui, uic
import sys
from Node import OutPut, OutPut2, Tree
from Node import InPut
from InputTree import SceneGraphModel
```

```
from ModelTree import ModelTree
from LogicalNodes import LogicalNodes
from View import View
import os, shutil

base, form = uic.loadUiType("dialogA.ui")

class WidgetA(base, form):
    def __init__(self, parent=None):
        super(WidgetA, self).__init__(parent)
        self.setupUi(self)

        self._HamiltonianDir = None

        #####Attributes#####
        self._paradict = {}
        self._integrator = []
        self._tree = None
        self._treeFromLoad = None

        self._dictHamil = {'CH3Quasie_exact': '1', 'CH4_rst': '2', 'NOC1': '9'}
        self._dictPES = {'CH3Potential': '1', 'PES_CH4': '2', 'PES_HCH4_Zang': '4'}
        self._potential = 'no Potential'

        self._mctdhConfig = None
        self._sysTreeFile = None
        self._inputFile = None
        self._SESmctdhConfig = None
        self._SESSysTreeFile = None
        self._SESinputFile = None
        self._TMPmctdhConfig = None
        self._TMPPsysTreeFile = None
        self._TMPP inputFile = None
```

```
self._dest = None

self._startingPath = None
self._ProjectName = None
self._SessionName = None
self._temporarySES = None
self._messagebut = None

#####ListModelHamilton#####
self._model = QtGui.QStandardItemModel(self.listHamilton)
for key in self._dictHamil:
    item = QtGui.QStandardItem(key)
    self._model.appendRow(item)
self.listHamilton.setModel(self._model)
self.listHamilton.setEditTriggers(QtGui.QAbstractItemView.NoEditTriggers)
self.listHamilton.clicked.connect(self.on_item_select1)

#####RadioButtonsPES#####
self.onRadio.setChecked(True)
self.onRadio.toggled.connect(self.setPES)
self.offRadio.toggled.connect(self.unsetPES)
self.offRadio.toggled.connect(self.noPotenial)

#####ListView of PES is built###
self.setPES()

#####RadioButtonsJob#####
self.RealRadio.toggled.connect(self.setJob1)
self.RealRadio.setChecked(False)
self.ImaginaryRadio.toggled.connect(self.setJob2)
self.EigenstateRadio.toggled.connect(self.setJob3)
self.fluxEigenstateRadio.toggled.connect(self.setJob4)

self._job = None
```

```
self._dictJob = {'integrate': self.RealRadio,
                 'integrate': self.ImaginaryRadio,
                 'eigenstates': self.EigenstateRadio,
                 'flux eigenstates': self.fluxEigenstateRadio}

#####PushBottoms#####
self.uiCancel.clicked.connect(self.cancel)
self.uiSaveJob.clicked.connect(self.saveProject)
self.uiLoad.clicked.connect(self.FromLoadToTMP)
self.uiStartCal.clicked.connect(self.runJob)

#####Line Edits#####
self.uiStartTime.textChanged.connect(self.change1)
self.uiEndTime.textChanged.connect(self.change2)
self.uiInit.textChanged.connect(self.change3)
self.uiIter.textChanged.connect(self.change4)

#####Networkx and MCTDH#####
self.setConfig = None
self.setSystem = None
self.modelTree = None
self.scene = None

def noPotenial(self):
    self._potential = 'no Potential'

def genereInput(self, inputFile):
    #####Get all Parameters from InPut.in#####
    inobj = InPut(inputFile)
    paradict = inobj._paradict
    self._treeFromLoad = inobj._treeString
    self._integrator = []
    try:
        self._integrator.append(paradict['start'])
```

```
        self._integrator.append(paradict['end'])
        self._integrator.append(paradict['dt'])
        self._integrator.append(paradict['iteration'])
        self._integrator.append(paradict['out'])
        self._mainfolder = paradict['mainfolder']
        self._hamiltonian = paradict['Hamiltonian']
        self._job = paradict['job']
        self._parameters = paradict['para']
        self._Comm = paradict['Comm']
        self._potential = paradict['Potential']
    except KeyError as e:
        pass
    if 'no Potential' in self._potential:
        #####RadioButtonsPES#####
        self.offRadio.setChecked(True)
        self.noPotenial()
    ####LineEdit#####
    self.uiStartTime.setText(self._integrator[0])
    self.uiEndTime.setText(self._integrator[1])
    self.uiInit.setText(self._integrator[2])
    self.uiIter.setText(self._integrator[3])

    self._dictJob[self._job].setChecked(True)

def getInput(self, key):
    ###Files for default Hamiltonians#####
    path = self._HamiltonianDir + '/' + key
    DotIn = self.Finder(path, 'in')
    self._inputFile = path + '/' + DotIn

    path = self._HamiltonianDir+'/'+str(key)

    sysTreeFile = self.Finder(path, 'txt')
```

```
self._mctdhConfig = self._HamiltonianDir + '/' + key + '/' + 'mctdh.config'
self._sysTreeFile = self._HamiltonianDir + '/' + key + '/' + sysTreeFile

#####Files for SES calculations#####
if self._ProjectName != None:
    if self._SessionName != None:
        self._SESmctdhConfig = self._startingPath + '/' + self._ProjectName
        self._SESSysTreeFile = self._startingPath + '/' + self._ProjectName
        self._SESinputFile = self._startingPath + '/' + self._ProjectName

def editSession(self, name):
    self.uiProjectName.blockSignals(True)
    self.uiProjectName.setText(str(name))
    self.uiProjectName.blockSignals(False)

def clearSession(self):
    self.uiProjectName.clear()

def makeParaDict(self):
    self._paradict['mainfolder'] = str(self._SessionName) + '/'
    self._paradict['start'] = self._integrator[0]
    self._paradict['end'] = self._integrator[1]
    self._paradict['dt'] = self._integrator[2]
    self._paradict['iteration'] = self._integrator[3]
    self._paradict['out'] = self._integrator[4]
    self._paradict['Hamiltonian'] = self._hamiltonian

    try:
        self._paradict['Potential'] = self._potential
    except AttributeError:
        pass

    self._paradict['job'] = self._job
```

```
self._paradict['para']      = self._parameters
self._paradict['Comm']       = self._Comm

def closeEvent(self, event):
    os.chdir("../")
    event.accept()

def showdialog3(self, Stringmes):
    msg = QtGui.QMessageBox()
    msg.setIcon(QtGui.QMessageBox.Information)

    msg.setText(Stringmes)
    msg.setStandardButtons(QtGui.QMessageBox.Save | QtGui.QMessageBox.Cancel)

    msg.buttonClicked.connect(self.msgbtn)
    msg.exec_()

def showdialog2(self, Stringmes):
    msg = QtGui.QMessageBox()
    msg.setIcon(QtGui.QMessageBox.Warning)

    msg.setText(Stringmes)
    msg.setStandardButtons(QtGui.QMessageBox.Yes | QtGui.QMessageBox.No)

    msg.buttonClicked.connect(self.msgbtn)
    msg.exec_()

def msgbtn(self, i):
    self._messagebut = str(i.text())

def showdialog(self, stringMes):
    msg = QtGui.QMessageBox()
```

```
msg.setIcon(QtGui.QMessageBox.Information)

msg.setText(stringMes)

msg.exec_()

def managefolder(self):

    if self._inputFile != None or self._sysTreeFile != None or self._mctdhCon

        shutil.copy2(self._mctdhConfig, self._TMPmctdhConfig)
        shutil.copy2(self._sysTreeFile, self._TMPsysTreeFile)
        shutil.copy2(self._inputFile, self._TMPP inputFile)
    else:
        pass

    def changeNode(self, my_index):
        topNode = self.modelTree.getNode2(my_index).child(0) #modelTree from
        self._tree.setRootNode(topNode)

        self.PicGenerate()

    def PicGenerate(self):
        #####Generate Outputfiles for new Pic#####
        self.output()

        #####Pic with MCTDH Code and Networkx#####
        if os.path.exists(self._TMPmctdhConfig):
            self.ModelTree = ModelTree(self._TMPmctdhConfig, self._TMPsysTreeFile)
        else:
            print 'Error'
        self.LogicalNodes = LogicalNodes(self.ModelTree.lay_matr_mode, self._TMPm
        self.View = View(self.ModelTree.label_mode, self.ModelTree.nodes_spf) #ob
        self.View.Display(self.LogicalNodes.G) #View method Display() generated .
```

```
#####QGraphicsView#####
pixmap = QtGui.QPixmap('nx_test.png')
self.scene = QtGui.QGraphicsScene(self)
self.scene.addPixmap(pixmap)
self.uiDisplayTree.setScene(self.scene)

def New_Session(self):
    name = str(self.uiProjectName.text())
    if self._SessionName == None:
        print name

def SESfiles(self):

    path = self._startingPath + '/' + self._ProjectName + '/tmp'
    DotIn = self.Finder(path, 'in')
    sysTreeFile = self.Finder(path, 'txt')

    if self._ProjectName != None:
        if self._SessionName != None:
            self._SESmctdhConfig = self._startingPath + '/' + self._ProjectName
            self._SESSysTreeFile = self._startingPath + '/' + self._ProjectName
            self._SESinputFile = self._startingPath + '/' + self._ProjectName

def saveProject(self):
    name = str(self.uiProjectName.text())
    self._SessionName = name
    self.SESfiles()
    Profiles = os.walk(self._startingPath+'/'+self._ProjectName+'/').next()[1]
    if name in Profiles:
        SESfiles = os.walk(self._startingPath+'/'+self._ProjectName+'/'+name)

    #####Checks if SES contains files#####
    if SESfiles:
```

```
        self.showdialog2('Overwriting %s?' %name)
        if 'Yes' in self._messagebut:
            self.output()
            self.fromTMPToSES()
            ###Saves all Parameter and Tree to *.in and tree only to *.tx
            self.esc()
        else:
            pass
    else:
        TMPfiles = os.walk(self._startingPath+'/'+self._ProjectName+'/tmp')
        if TMPfiles:
            self.output()
            self.fromTMPToSES()
            self.esc()
        else:
            self.showdialog('Nothing to save?')

    else:
        if name == '':
            print name
            self.showdialog('Please give Session name')
        else:
            os.chdir(self._startingPath+'/'+self._ProjectName)
            os.mkdir(name)
            os.chdir(self._startingPath)
            self.output()
            self.fromTMPToSES()
            self.esc()

    if self._potential == 'no Potential':
        with open(self._SESinputFile, 'r') as f:
            lines = f.readlines()
        with open(self._SESinputFile, 'w') as f:
            for line in lines:
                if 'Potential' not in line:
```

```
f.write(line)

def cancel(self):
    self.removeContent()
    self.esc()

def removeContent(self):
    TMPpath = self._startingPath +'/' + self._ProjectName + '/tmp'

    try:
        shutil.rmtree(TMPpath)
    except OSError:
        raise

    sysPath = self._startingPath +'/' + self._ProjectName
    os.chdir(sysPath)
    os.mkdir('tmp')
    os.chdir(self._startingPath)

def copyLoad(self):
    LOADinputFile = str(QtGui.QFileDialog.getOpenFileName())
    try:
        shutil.copy2(LOADinputFile, self._TMPinputFile)
    except Exception:
        raise

def FromLoadToTMP(self):

    self.clearTree()

    self._TMPinputFile = self._startingPath + '/' \
+ self._ProjectName + \
'/tmp/InPut.in'
```

```
self._SESSysTreeFile = self._startingPath + '/' \
+ self._ProjectName + '/' \
+ self._SessionName + '/Load.txt'

self._SESinputFile = self._startingPath + '/' \
+ self._ProjectName + '/' \
+ self._SessionName + '/InPut.in'

####removes tmp folder's content####
self.removeContent()

####copies *.in file to tmp folder####
self.copyLoad()

####generates Parameter from *.in file####
self.genereInput(self._TMPinputFile)
self.makeParaDict()
pathTMP = self._startingPath + '/' + self._ProjectName + '/tmp'
outobj = OutPut2(self._paradict, self._treeFromLoad, self._TMPinputFile,
outobj.savefile()
outobj.savefile2()
####Tree will be constructed from parameters####
self._TMPmctdhConfig = self._startingPath + '/' \
+ self._ProjectName + \
'/tmp/Load.txt'

self._TMPSysTreeFile = self._startingPath + '/' \
+ self._ProjectName + \
'/tmp/mctdh.config'

self.TreeOnly()
```

```
def checkTMP(self):
    sysPathTMP = self._startingPath +'/' + self._ProjectName + '/tmp'
    files = os.walk(sysPathTMP).next()[2]
    if files:
        return True
    return False

def fromHToTMP(self, item):
    if self.checkTMP():

        self.showdialog2('Overwriting temporary Settings?')
        if 'Yes' in self._messagebut:
            self.fromHToTMPPinner(item)
        else:
            pass
    else:
        self.fromHToTMPPinner(item)

def fromHToTMPPinner(self, item):
    self.clearTree()
    sysPath = self._HamiltonianDir+'/' +item

    sysFile = self.Finder(sysPath, '.txt')
    DotIn = self.Finder(sysPath, '.in')

    self._mctdhConfig = sysPath+'/'+'mctdh.config'
    self._sysTreeFile = sysPath+'/' +sysFile
    self._inputFile    = sysPath+'/' +DotIn

    self._TMPmctdhConfig = self._startingPath + ' /' \
+ self._ProjectName + \
'/tmp/mctdh.config'

    self._TMPSysTreeFile = self._startingPath + ' /' \

```

```
+ self._ProjectName + \
'/tmp/' + sysFile

self._TMPPin inputFile = self._startingPath + '/' \
+ self._ProjectName + \
'/tmp/' + DotIn

try:
    shutil.copy2(self._mctdhConfig, self._TMPPmctdhConfig)
    shutil.copy2(self._sysTreeFile, self._TMPPsysTreeFile)
    shutil.copy2(self._inputFile, self._TMPPin inputFile)
except Exception:
    raise

def fromSESToTMP(self, sysFile):

    self._TMPPmctdhConfig = self._startingPath + '/' \
+ self._ProjectName + \
'/tmp/mctdh.config'

    self._TMPPsysTreeFile = self._startingPath + '/' \
+ self._ProjectName + \
'/tmp/' + sysFile

    SESpath = self._startingPath + '/' \
+ self._ProjectName + '/' + \
self._SessionName

    DotIn = self.Finder(SESpath, '.in')

    self._TMPPin inputFile = self._startingPath + '/' \
+ self._ProjectName + \
'/tmp/' + DotIn
```

```
try:
    shutil.copy2(self._SESmctdhConfig, self._TMPmctdhConfig)
    shutil.copy2(self._SESSysTreeFile, self._TPMsysTreeFile)
    shutil.copy2(self._SESinputFile, self._TPMinputFile)
except Exception:
    raise

def fromTMPToSES(self):
    TMPpath = self._startingPath + '/' \
              + self._ProjectName + '/' + \
              'tmp'

    sysFile = self.Finder(TMPpath, 'txt')
    DotIn = self.Finder(TMPpath, '.in')

    self._TPMmctdhConfig = self._startingPath + '/' \
              + self._ProjectName + \
              '/tmp/mctdh.config'

    self._TPMsysTreeFile = self._startingPath + '/' \
              + self._ProjectName + \
              '/tmp/' + sysFile

    self._TPMinputFile = self._startingPath + '/' \
              + self._ProjectName + \
              '/tmp/' + DotIn

try:
    shutil.copy2(self._TPMmctdhConfig, self._SESmctdhConfig)
    shutil.copy2(self._TPMsysTreeFile, self._SESSysTreeFile)
    shutil.copy2(self._TPMinputFile, self._SESinputFile)
except Exception:
    raise
```

```
def esc(self):
    self.close()

def setSessionName(self, name):
    self._SessionName = name

def folderExist(self):
    folders = os.walk(self._startingPath+'/'+self._ProjectName).next()[1]
    if self._SessionName in folders:
        self.showdialog('Folder already exists!')
    return False
return True

def change0(self):
    self._SessionName = str(self.uiProjectName.text())

def change1(self):
    self._integrator[0] = str(self.uiStartTime.text())
def change2(self):
    self._integrator[1] = str(self.uiEndTime.text())
def change3(self):
    self._integrator[2] = str(self.uiInit.text())
def change4(self):
    self._integrator[3] = str(self.uiIter.text())

def setJob1(self):
    self._job = "integrate"
def setJob2(self):
    self._job = "integrate"
def setJob3(self):
    self._job = "eigenstates"
def setJob4(self):
```

```
self._job = "thermalflux"

def unsetPES(self):
    self.modelPES.removeRows(0, len(self._dictPES), QtCore.QModelIndex())

def setPES(self):
    #####ListModelPES#####
    self.modelPES = QtGui.QStandardItemModel(self.listPES)
    for key in self._dictPES:
        item = QtGui.QStandardItem(key)
        self.modelPES.appendRow(item)
    self.listPES.setModel(self.modelPES)
    self.listPES.setEditTriggers(QtGui.QAbstractItemView.NoEditTriggers)
    self.listPES.clicked.connect(self.on_item_select2)

def makedir(self):
    path = self._startingPath + '/' + self._ProjectName + '/tmp'
    try:
        os.makedirs(path)
    except (IOError, OSError) as e:
        pass

def start(self):
    self.clearTree()
    name = str(self.uiProjectName.text())
    self._SessionName = name
    ##### if SES contains files, these files will be copied to TMP
    try:
        filenames = os.walk(self._startingPath+'/'+self._ProjectName+'/'+self._SessionName)
        for val in filenames:
            if 'txt' in val:
                sysTreeFile = val

        self._SESmctdhConfig = self._startingPath + '/' + self._ProjectName + '/' + self._SessionName + '/' + self._ProjectName + '.mctdh' + self._SessionName + '.ini'
```

```
self._SESsysTreeFile = self._startingPath + '/' + self._ProjectName
self._SESinputFile = self._startingPath + '/' + self._ProjectName + '.'

#####copies files from SES to TMP
self.fromSESToTMP(sysTreeFile)

#####Parameters from InPut.in in TMP will be loaded#####
self.genereInput(self._TMPinputFile)

#####Tree will be constructed from parameters#####
self.TreeOnly()

except (StopIteration, UnboundLocalError):
    #####when UnboundLocalError is raised then there are no files in SES#####
    # -> deleteing tmp
    self.removeContent()
    self.clearTree()

def clearTree(self):
    try:
        self.scene.clear()
        self.uiDisplayTree.setScene(self.scene)
        self.modelTree.removeRow(0)
    except (IndexError, AttributeError) as e:
        pass

def Finder(self, path, app):
    fileList = os.walk(path).next()[2]
    return [f_ for f_ in fileList if app in f_][0]

def TreeOnly(self):
    #####TreeView#####
    TMPpath = self._startingPath+'/' +self._ProjectName+ '/tmp'
```

```
self._TMPmctdhConfig = TMPpath+ '/mctdh.config'
textFile = self.Finder(TMPpath, 'txt')
self._TMPSysTreeFile = TMPpath+ '/' +textFile
self._tree = Tree(self._TMPmctdhConfig, self._TMPSysTreeFile)
self.modelTree = SceneGraphModel(self._tree._rootNode0)
self.uiTree.setModel(self.modelTree)
self.uiTree.expandAll()
self.uiTree.resizeColumnToContents(0)
self.uiTree.resizeColumnToContents(1)
self.uiTree.clicked.connect(self.changeNode)
#####make Pic from tmp#####
self.PicGenerate()

def generateTree(self, item):
    key = item
    self.getInput(key)

    self._hamiltonian = self._dictHamil[str(key)]

    ####generates Tree####
    self.TreeOnly()

def on_item_select1(self, item):
    key = str(item.data().toString())

    #####Copy from default Hamilton to tmp
    self.fromHToTMP(key)

    #####Generate input from *.in to self._paradict#####
    self.genereInput(self._TMPP inputFile)
```

```
#####Building Tree#####
self.generateTree(key)

def on_item_select2(self, item):
    key = item.data().toString()
    self._potential = self._dictPES[str(key)]

def output(self):
    """Class OutPut takes all parameters and saves them in File by creating
the object of this class"""
    self.makeParaDict()
    outobj = OutPut(self._tree, self._paradict, self._TMPsysTreeFile, self._T
    outobj.savefile()
    outobj.savefile2()

def runJob(self):
    self.esc()
    self.process = QtCore.QProcess()
    self.process.setProcessChannelMode(QtCore.QProcess.MergedChannels)
    self.process.readyReadStandardOutput.connect(self.mctdh0Out)
    inputFile = self._startingPath+'/'+self._ProjectName+'/'+self._SessionNam
    DotIn = self.Finder(inputFile, 'in')
    inputFile = inputFile+'/'+DotIn
    self.results(inputFile)

def results(self, inputFile):
    os.chdir(self._startingPath+'../../Results')
    self.genereInput(inputFile)
    try:
        os.mkdir(self._mainfolder)
    except OSError:
        pass
    mctdh = '/home/piet/newRepo2/QuantumDynamics/build/bin/mctdh'
```

```
self.process.start(mctdh+' '+inputFile)
os.chdir(self._startingPath)

def mctdhOut(self):
    output = str(self.process.readAllStandardOutput())
    print output
```

Literaturverzeichnis

- [1] H.-D. Meyer, U. Manthe, and L. S. Cederbaum, Chem. Phys. Lett. **165**, 73 (1990).
- [2] U. Manthe, H.-D. Meyer, and L. S. Cederbaum, J. Chem. Phys. **97**, 3199 (1992).
- [3] H. Wang and M. Thoss, J. Chem. Phys. **119**, 1289 (2003).
- [4] U. Manthe, The Journal of Chemical Physics **128**, 164116 (2008).
- [5] G. A. Worth, H. D. Meyer, H. Koeppel, L. S. Cederbaum, and I. Burghardt, Int. Rev. Phys. Chem. **27**, 569 (2008).
- [6] T. Westermann, J. B. Kim, M. L. Weichman, C. Hock, T. I. Yacovitch, J. Palma, D. M. Neumark, and U. Manthe, Angew. Chem. Int. Ed. **53**, 1122 (2014).
- [7] E. Y. Wilner, H. Wang, M. Thoss, and E. Rabani, Phys. Rev. B **89**, 205129 (2014).
- [8] H. Wang, J. Phys. Chem. A **118**, 9253 (2014).
- [9] K. Balzer, Z. Li, O. Vendrell, and M. Eckstein, Phys. Rev. B **91**, 045136 (2015).
- [10] M. Schroeter and O. Kuehn, J. Phys. Chem. A **117**, 7580 (2013).
- [11] M. Saab, M. Sala, B. Lasorne, F. Gatti, and S. Guerin, J. Chem. Phys. **141**, 134114 (2014).
- [12] S. Lopez-Lopez, R. Martinazzo, and M. Nest, J. Chem. Phys. **134**, 094102 (2011).
- [13] F. Bouakline, F. Lueder, R. Martinazzo, and P. Saalfrank, J. Phys. Chem. A **116**, 11118 (2012).
- [14] L. Uranga-Pina, C. Meier, and J. Rubayo-Soneira, Chem. Phys. Lett. **543**, 12 (2012).

Literaturverzeichnis

- [15] M. Moix Teixidor and F. Huarte-Larranaga, Chem. Phys. **399**, 264 (2012).
- [16] J. Wahl, R. Binder, and I. Burghardt, Comp. Theo. Chem. **1040**, 167 (2014).
- [17] J. M. Schurer, P. Schmelcher, and A. Negretti, Phys. Rev. A **90**, 033601 (2014).
- [18] V. S. Reddy, C. Camacho, J. Xia, R. Jasti, and S. Irle, J. Chem. Theo. Comp. **10**, 4025 (2014).
- [19] W. Eisfeld, O. Vieuxmaire, and A. Viel, J. Chem. Phys. **140**, 224109 (2014).
- [20] A. Valdes and R. Prosmiti, J. Phys. Chem. A **117**, 9518 (2013).
- [21] T. Mondal, S. R. Reddy, and S. Mahapatra, J. Chem. Phys. **137**, 054311 (2012).
- [22] D. Skouteris and A. Lagana, Chem. Phys. Lett. **575**, 18 (2013).
- [23] B. Zhao, D.-H. Zhang, S.-Y. Lee, and Z. Sun, J. Chem. Phys. **140**, 164108 (2014).
- [24] M. D. Coutinho-Neto, A. Viel, and U. Manthe, J. Chem. Phys. **121**, 9207 (2004).
- [25] T. Hammer, M. D. Coutinho-Neto, A. Viel, and U. Manthe, J. Chem. Phys. **131**, 224109 (2009).
- [26] T. Hammer and U. Manthe, J. Chem. Phys. **134**, 224305 (2011).
- [27] M. Schroeder, F. Gatti, and H.-D. Meyer, J. Chem. Phys. **134**, 234307 (2011).
- [28] T. Hammer and U. Manthe, J. Chem. Phys. **136**, 054105 (2012).
- [29] M. Schroeder and H.-D. Meyer, J. Chem. Phys. **141**, 034116 (2014).
- [30] O. Vendrell, F. Gatti, D. Lauvergnat, and H.-D. Meyer, Angew. Chemie Int. Ed. **46**, 6918 (2007).
- [31] O. Vendrell, F. Gatti, D. Lauvergnat, and H.-D. Meyer, J. Chem. Phys. **127**, 184302 (2007).
- [32] O. Vendrell, F. Gatti, and H.-D. Meyer, J. Chem. Phys. **127**, 184303 (2007).
- [33] O. Vendrell, M. Brill, F. Gatti, and H.-D. Meyer, J. Chem. Phys. **130**, 234305 (2009).

Literaturverzeichnis

- [34] O. Vendrell, F. Gatti, and H.-D. Meyer, *J. Chem. Phys.* **131**, 034308 (2009).
- [35] F. Huarte-Larrañaga and U. Manthe, *J. Chem. Phys.* **113**, 5115 (2000).
- [36] F. Huarte-Larrañaga and U. Manthe, *J. Phys. Chem. A* **105**, 2522 (2001).
- [37] T. Wu, H.-J. Werner, and U. Manthe, *Science* **306**, 2227 (2004).
- [38] G. Schiffel and U. Manthe, *J. Chem. Phys.* **132**, 084103 (2010).
- [39] R. van Harreveldt, G. Nyman, and U. Manthe, *J. Chem. Phys.* **126**, 084303 (2007).
- [40] G. Nyman, R. van Harreveldt, and U. Manthe, *J. Phys. Chem. A* **111**, 10331 (2007).
- [41] G. Schiffel and U. Manthe, *J. Chem. Phys.* **132**, 191101 (2010).
- [42] G. Schiffel and U. Manthe, *J. Chem. Phys.* **133**, 174124 (2010).
- [43] R. Welsch and U. Manthe, *J. Chem. Phys.* **141**, 051102 (2014).
- [44] R. Welsch and U. Manthe, *J. Chem. Phys.* **141**, 174313 (2014).
- [45] R. Welsch and U. Manthe, *J. Chem. Phys.* **142**, 064309 (2015).
- [46] R. Welsch and U. Manthe, *J. Phys. Chem. Lett.* **6**, 338 (2015).
- [47] G. A. Worth, H.-D. Meyer, and L. S. Cederbaum, *J. Chem. Phys.* **105**, 4412 (1996).
- [48] G. A. Worth, H.-D. Meyer, and L. S. Cederbaum, *J. Chem. Phys.* **109**, 3518 (1998).
- [49] A. Raab, G. A. Worth, H.-D. Meyer, and L. S. Cederbaum, *J. Chem. Phys.* **110**, 936 (1999).
- [50] H. Wang, D. E. Skinner, and M. Thoss, *J. Chem. Phys.* **125**, 174502 (2006).
- [51] I. Kondov, M. Cizek, C. Benesch, M. Thoss, and H. Wang, *J. Phys. Chem. C* **111**, 11970 (2007).
- [52] I. R. Craig, M. Thoss, and H. Wang, *J. Chem. Phys.* **135**, 064504 (2011).
- [53] H. Wang, I. Pshenichnyuk, R. Härtle, and M. Thoss, *J. Chem. Phys.* **135**, 244506 (2011).

- [54] T. Westermann, R. Brodbeck, A. B. Rozhenko, W. W. Schoeller, and U. Manthe, J. Chem. Phys. **135**, 184102 (2011).
- [55] M. H. Beck, A. Jäckle, G. A. Worth, and H.-D. Meyer, Physics Reports **324**, 1 (2000).
- [56] H.-D. Meyer and G. A. Worth, Theor. Chem. Acc. **109**, 251 (2003).
- [57] F. Huarte-Larrañaga and U. Manthe, Z. Phys. Chem. **221**, 171 (2007).
- [58] H.-D. Meyer, F. Gatti, and G. A. Worth, *Multidimensional Quantum Dynamics: MCTDH Theory and Applications* (Weinheim: Wiley-VCH, 2009).
- [59] U. Manthe, Mol. Phys. **109**, 1415 (2011).
- [60] H.-D. Meyer, Wiley Interdisciplinary Reviews: Computational Molecular Science **2**, 351 (2012).
- [61] *An open source machine learning framework for everyone* (2018), URL <https://www.tensorflow.org/>.
- [62] *Scientific computing tools for python* (2018), URL <https://scipy.org/about.html>.
- [63] F. Huarte-Larrañaga and U. Manthe, J. Chem. Phys. **116**, 2863 (2002).
- [64] T. Wu, H.-J. Werner, and U. Manthe, J. Chem. Phys. **124**, 164307 (2006).
- [65] F. Huarte-Larrañaga and U. Manthe, J. Chem. Phys. **117**, 4635 (2002).
- [66] J. M. Bowman, D. Wang, X. Huang, F. Huarte-Larrañaga, and U. Manthe, J. Chem. Phys. **114**, 9683 (2001).
- [67] C. Cattarius, G. A. Worth, H.-D. Meyer, and L. S. Cederbaum, J. Chem. Phys. **115**, 2088 (2001).
- [68] H. Wang, J. Chem. Phys. **113**, 9948 (2000).
- [69] H. Wang, M. Thoss, and W. Miller, J. Chem. Phys. **115**, 2979 (2001).
- [70] M. Nest and H.-D. Meyer, J. Chem. Phys. **119**, 24 (2003).

Literaturverzeichnis

- [71] U. Manthe, J. Chem. Phys. **105**, 6989 (1996).
- [72] R. van Harreveldt und U. Manthe, J. Chem. Phys. **121**, 5623 (2004).
- [73] R. van Harreveldt and U. Manthe, J. Chem. Phys. **123**, 064106 (2005).