TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

D I P L O M A R B E I T

Effective calculations of fusion-relevant highly charged ion collisions

ausgeführt am

Institut für Allgemeine Physik der
Technischen Universität Wien

unter Anleitung von

Univ.-Prof. Dr. Friedrich Aumayr
Dipl.-Ing. Katharina Igenbergs

durch

Markus Wallerberger
Favoritenstraße 89/1/2
1100 Wien

Wien, September 2011 Markus Wallerberger

Kurzfassung

Die Vision, die Energieumsetzung unserer Sonne in einem *Kernfusionsreaktor* zu imitieren, ist alt aber wird auch heute noch von dem Wunsch befeuert, den Energiehunger unserer globalisierten Wirtschaft langfristig zu stillen. Trotz intensivster Forschungsbemühungen bleibt jedoch die Stabilisierung der extremen Bedingungen in Fusionsplasmen eine große Herausforderung, was Möglichkeiten zur akkuraten Plasmadiagnostik unerlässlich macht. Hier ist die *Umladungsspektroskopie* (CXRS) hervorzuheben, bei der Ladungsaustauschprozesse zwischen hochgeladenen Verunreinigungen des Plasmas und einem Strahl aus neutralen Atomen über charakteristische Emissionslinien beobachtet werden. Diese Spektren geben dann Aufschluss über Plasmadichte und -temperatur.

Die Qualität der dadurch erhaltenen Daten hängt jedoch stark von dem theoretischen Verständnis von Ladungsaustauschprozessen in atomaren Stößen zwischen hochgeladenen Ionen und Neutralteilchen, insbesondere Wasserstoff, ab. Hier hat sich die halbklassische *Methode der gekoppelten Reaktionskanäle* bewährt, bei der das volle quantenmechanische Stoßproblem auf die Wechselwirkung einer endlichen Anzahl definierter Elektronenzustände reduziert wird. Wir verwenden eine Basis aus gebundenen Zuständen auf beiden Stoßpartnern (*Atomorbitale*) und erweitern diese mit so-genannten Pseudozuständen als Modell des Kontinuums.

Die Wechselwirkungen selbst werden mit der von Shakeshaft entwickelten Fourier-Transformationsmethode berechnet, welche eine symbolische Form der zugehörigen Matrixelemente liefert. Für große und komplexe Systeme zeigt sich jedoch, dass die Rechenzeit stark zunimmt und die numerische Genauigkeit gleichzeitig stark leidet. Um diesen Problemen entgegenzuwirken, wurden komplexe Baumstrukturen für die Matrixelemente entwickelt, welche die Performance der symbolischen Manipulation und die numerische Genauigkeit der Auswertung stark erhöhen. Gleichzeitig wurde mit einer Referenzimplementierung der Formel von Kocbach und Liska die Erstellung der Strukturen verbessert.

Diese Verbesserungen erlauben die Behandlung von Stoßprozessen zwischen „nacktem“ Neon (Ne^{10+}) und neutralem Wasserstoff. In der Fusionsforschung ist Neon als Kühlgas für den Plasmarand interessant, um die thermische Belastung der Reaktorwände zu verringern. Weiters geben wir Daten für Kollisionen mit Sauerstoff (O^{8+}) und Fluor (F^{9+}) an, da diese jene für schwächer geladenes Neon repräsentieren. Die so erhaltenen Stoßquerschnitte für Ladungsaustausch und Ionisation stimmen mit Literaturdaten sehr gut überein.

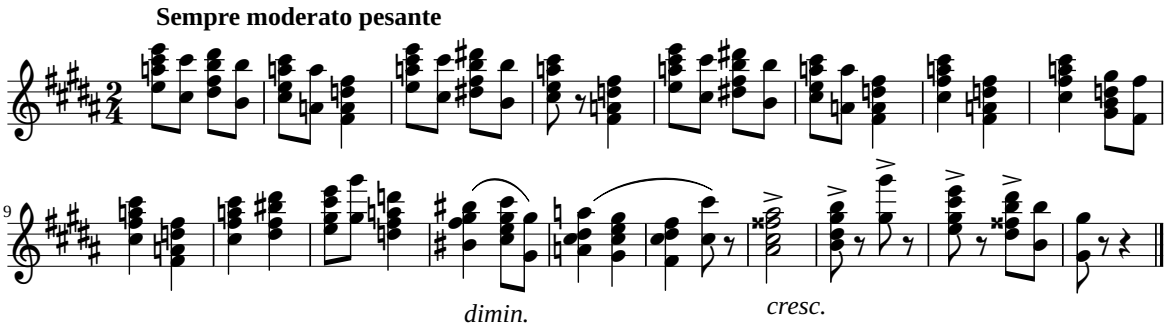
Abstract

Thermonuclear fusion, essentially imitating the way our sun converts its mass to radiation, provides a possible long-term solution for the energy demands of our globalised economy and is therefore a very active field of research. Unfortunately, the extreme thermodynamic conditions required for fusion plasmas are hard to establish and even harder to stabilise, which makes the field of plasma diagnostics a cornerstone of fusion research. An excellent tool for this is *charge exchange recombination spectroscopy* (CXRS), where one observes the radiation spectrum following charge exchange between a beam of neutral atoms and highly charged plasma impurities in order to estimate plasma characteristics such as temperature and density.

For CXRS to yield accurate data, a thorough understanding of charge exchanges processes in atomic collisions between highly charged ions and neutral atoms — hydrogen in particular — is required. The *close-coupling method* in its impact parameter formulation, already developed by Bates and McCarroll in 1958, is a semi-classical tool for the high-precision calculation of atomic collision data: essentially, the full quantum mechanical problem is reduced to the interaction of finite, known set of interacting electron states. In our calculations we use a set of *atomic orbitals* (AO), bound states on either nucleus, and amend them with pseudo-states to model the continuum.

For the computation of the actual channel interactions, we follow the Fourier transform method originally developed by Shakeshaft, which yields a symbolic form for the interaction matrix elements. With growing number of channels, however, the computational complexity and numerical inaccuracies rise very fast. Therefore, we developed sophisticated tree structures which significantly speed up symbolic manipulation and evaluation while also mitigating the risk of numerical errors. We also provide an implementation of Kocbach and Liska's improvements to the creation of the symbolic structures.

Using these methods, we computed total and partial charge exchange and ionisation cross sections for the collisions of fully stripped neon (Ne^{10+}) with neutral hydrogen. Neon is in particular interesting to fusion research, because it will injected into future *tokamak* reactors like ITER to radiatively cool down the outer edge of the plasma, thus reducing the thermal stress on the reactor wall materials. Since other charge states of neon are closely related to fully stripped ions of the same charge, we also provide charge exchange data for oxygen (O^{8+}) and fluorine (F^{9+}) collisions with neutral hydrogen. We find that our data is in excellent agreement both with the experiment and calculations performed by others.



Modest Mussorgsky – Pictures at an Exhibition
"Bydło" in G-sharp minor, measures 21 – 38

Contents

List of Symbols

Symbol	Meaning	Eq.
$\mathcal{H}$	Full Hilbert space for the electron	
$\mathcal{L}$	Linear span	
$\mathcal{S}$	Finite subspace of $\mathcal{H}$ spanned by trial functions (χ_i)	
$\mathcal{T}$	Formal time reversal $t \mapsto -t$	
$\mathcal{S}_\mathrm{T}, \mathcal{S}_\mathrm{P}$	Target and projectile part of $\mathcal{S}$ in two-centre expansions	
$\mathbb{T}$	Circle group, i. e., the set of complex numbers with modulus 1; $\mathbb{T} := \{x \in \mathbb{C} : \lvert x\rvert = 1\}$ (isomorphic to $U(1)$)	
$\mathbb{1}$	Identity operator (or unit matrix) on a space	
A^{q+}	q-fold charged ion A	
B^A	Generalised eigenbasis	
H	(a) Full Hamiltonian $\mathcal{H} \to \mathcal{H}$, (b) Matrix elements of H_CC on $\mathcal{S}$	2.1
H_CC	Close-coupling Hamiltonian	2.13
M	Coupling matrix on $\mathcal{S}$	2.24
M_TP,M_PT	Target-projectile/Projectile-target two-centre coupling matrix	2.38c
M_PP,M_TT	Target/projectile one-centre coupling matrix	2.38b
P_R	Momentum operator to the nuclear movement	
P, P_r	Momentum operator to the electron movement	
Q	Spatial transformation related to time reversal	
S	Overlap matrix on $\mathcal{S}$	2.23
$S_\mathrm{TP}, S_\mathrm{PT}$	Target-projectile/Projectile-target two-centre overlap matrix	2.38a
T	Total kinetic energy	2.1
T_{ij}	Time-dependent part of matrix elements	2.31
V	Total potential energy	
V_T,V_P	Effective Core potentials of the target and the projectile, respectively	
V_N	Inter-nuclear potential	
W	Effective spherically symmetric inter-nuclear potential	

Symbol	Meaning	Eq.
$\hat{K}$	Complex conjugation operator	
$\hat{M}$	Motion reversal operator	
$\hat{T}$	Time reversal operator	
$\vec{a}, a_i$	Amplitude vector of the wave function expansion	
$\vec{b}$	Impact parameter of the collision	
$\vec{R}$	Internuclear distance (between target and projectile)	
$\vec{r}$	Electron coordinate with respect to some origin	
$\vec{r}_{\mathrm{A}}$	Electron coordinate with respect to centre A	
$\vec{v}$	Impact velocity of the projectile nucleus	
σ_{CX},σ_{tot}	Total charge exchange cross section	
χ_i	Basis states of the close-coupling method	
ψ	Electronic wave function	

1 Introduction

Energy – its conversion, consumption, supply and reserves – is a key issue of the modern globalised economy, shifting in and out of public focus since the first global energy crisis during the 1970s. With the 2011 accident [1] at the *Fukushima I* nuclear power plant [2, 3], it again became clear that governments are taking great risks to meet energy demands [4], since energy supply is vital to any economic system [5].

The idea to utilise *thermonuclear fusion*, essentially copying the way our sun converts its mass to energy, is quite old yet tempting: firstly, deuterium fusion has a much higher yield than all fission processes (see Figure 1.1); secondly, a virtually unlimited supply of deuterium lies encapsulated in our water reserves; and finally, there is hardly any radioactive waste or chance of uncontrollable reactions, making it a rather "clean" energy source.

On closer examination, it turned out it was very hard to produce and even harder to stabilise the extreme temperature, pressure and charge density for the fusion plasma. As a result, a tradition established itself in the 1970s that scientists regularly would call fusion technology to be ready some twenty years in the future. Nowadays, predictions are usually less bold because most of the stability problems are still subject to intense research, most prominently at the ITER experimental *tokamak* reactor currently being built in Southern France[1].

In order to understand the stability of the fusion plasma, one needs to know the internal dynamics and distribution of the charged particles. A diagnostic tool proven valuable for this task is *charge exchange recombination spectroscopy* (CXRS). Here, a beam of specific neutral atoms (e. g., hydrogen or lithium) is directed at the plasma and its interactions with the plasma ions are observed through characteristic emission lines from de-excitation [7]. Engineers are also interested in the behaviour of atoms like tungsten, which are torn out of the reactor wall by the passing fuel stream and can "poison" the plasma [8].

These open questions renewed the interest in the field of classic single-electron ion–atom collisions. We will focus on the well-established *close-coupling method*, which approximates the atomic collision process as the interaction of a finite set of *channels* for the electron (Chapter 2). In order to model collisions involving heavy ions, a large number of such channels is required, which unfortunately drastically increases the com-

[1] International Thermonuclear Experimental Reactor (see `http://iter.org`)

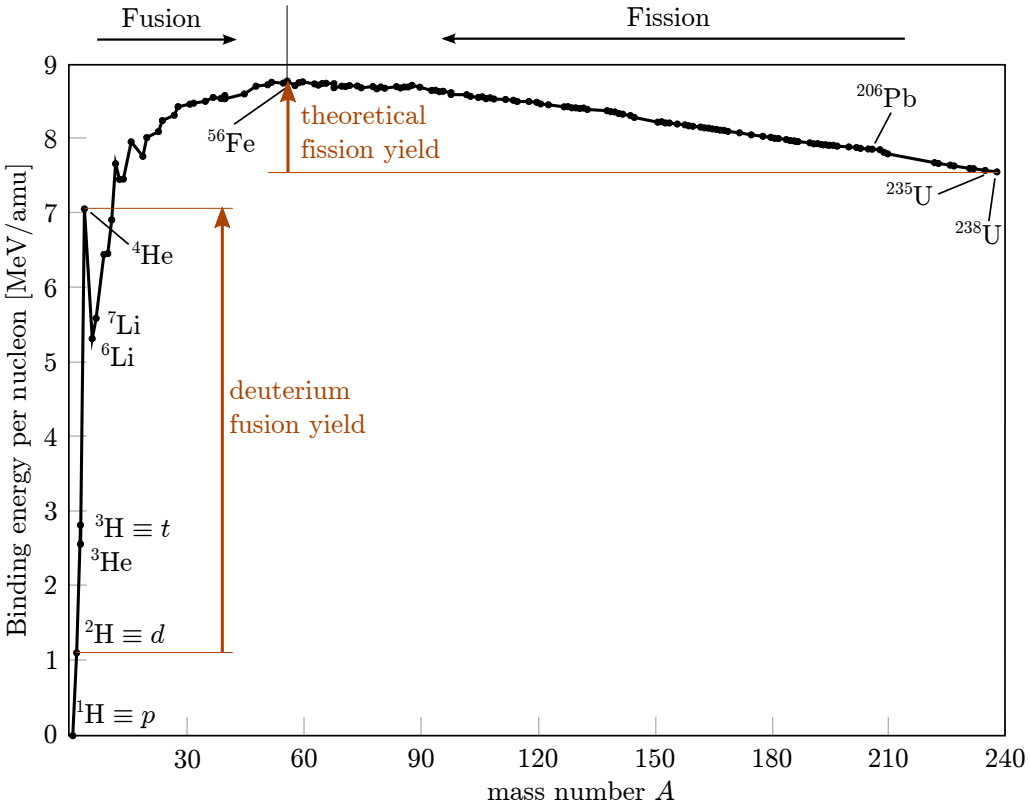


Figure 1.1: Average nuclear binding energy per nucleon for common isotopes [6] (by convention, the nuclear binding energy is positive). Yield for deuterium fusion and maximum yield for fission processes.

putational complexity of the system.

In Chapter 3, therefore, we improve present programs using contemporary numerical and computational strategies. Using our own implementation of these methods, we compute cross sections for excitation, charge exchange and ionisation as well as emission coefficients for collisions involving highly-charged ions. Chapter 4 presents these results and compares them to experimental data from the literature and calculations done by others. Also, fusion-relevant emission coefficients are derived and printed. The thesis concludes with a summary of the results and an outlook in Chapter 5. As appendices, detailed calculations as well as a comprehensive technical description of the code is provided.

2 The close-coupling method

We have seen in the previous chapter that thorough understanding of ion–atom collisions is crucial for the prediction of the dynamics of a plasma and therefore also for its experimental diagnostics. Unfortunately, the fully-fledged scattering set-up constitutes at least a *three-body problem* (two nuclei and the electrons) and is not accessible to analytical treatment.

To overcome this, several numerical and hybrid numerical-analytical methods have been developed to approximate the (inelastic) scattering cross sections. One of the most successful ones is the *close-coupling method* originally developed by Bates and McCaroll [9]. It has been used to predict both charge exchange, excitation and ionisation cross sections with high accuracy (for a collection of important results, refer to [10, 11]). Moreover, several fusion-relevant collision systems have been successfully treated using this method in its *impact parameter formulation* [12, 13, 14], which will also be the primary focus of this thesis.

We will mainly concentrate on the scattering of hydrogen atoms on highly charged heavy ions A (see Chapter 1):

$$\mathrm{A}^{q+} + (\mathrm{H}^{+} + \mathrm{e}^{-}) \rightarrow ?$$

For fully-stripped ions, we get a problem of reasonable scale. For many-electron systems, however, the dynamics complicate rapidly. Therefore we assume that all electrons but the active one are tightly-bound to the core and the active electron moves in a *mean single-particle potential* [15]. This approximation becomes more accurate with higher charges, since the main capture channel is more loosely bound in this case (see Section 4.1).

The Hamiltonian for this system can be written as follows:

$$H = T + V_{\mathrm{T}} + V_{\mathrm{P}} + V_{\mathrm{N}} \tag{2.1}$$

where V_{T} and V_{P} are the mean single-particle potentials of the target and projectile for the active electron, respectively, and V_{N} is the interaction potential of the nuclei. Separating the centre-of-mass movement, the kinetic energy T can be described by two Jacobian coordinates and our problem becomes an effective two-particle problem. The choice of coordinates will be discussed in Section 2.1.1.

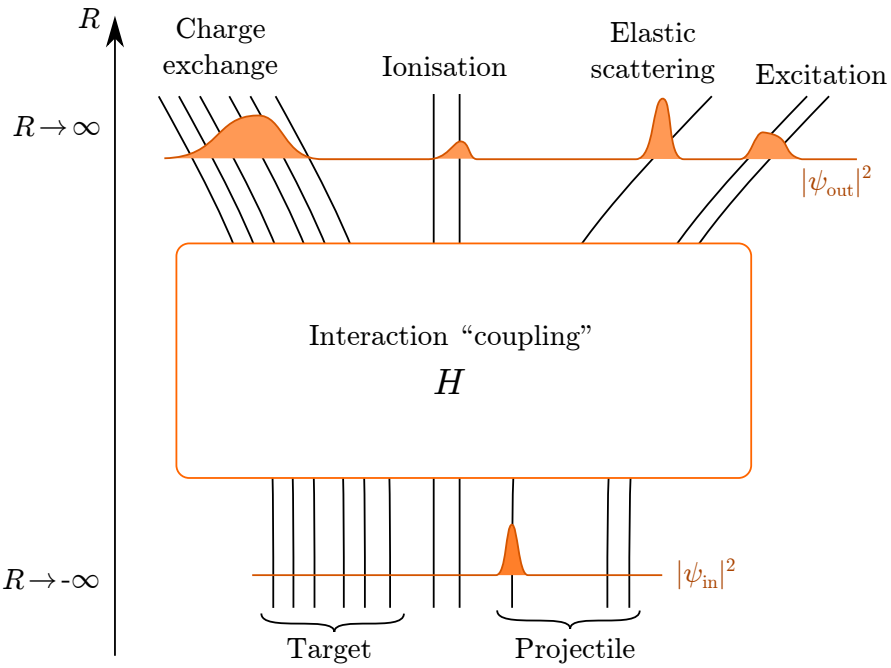


Figure 2.1: The scattering process modelled as the interaction of channels (black lines), parametrized by the internuclear separation. The orange faces refer to the population of the channels before and after the scattering process.

2.1 Coupled channels

The basic idea of the close-coupling method is that the scattering process can be described as the interaction of a set of known *channels*. A channel in our case is a defined state of the electron in the asymptotic region, i. e., a possible "outcome" of the scattering process[1].

In our case, typical channels include:

1. *Elastic scattering*, where the electron remains in the initial state after the collision;
2. *Excitation*, where scattering lifts the electron to an outer shell;
3. *Charge exchange*, where the electron moves to a – usually excited – state on the other nucleus; and

[1] More formally, we define a *set of channels* as a set eigenfunctions $\{\chi_i\} \in \mathcal{H}$ of an observable A which only acts on one part of a separable Hilbert space $\mathcal{H} = \mathcal{H}_1 \oplus \mathcal{H}_2$:

$$A(|\phi\rangle \otimes |\chi\rangle) = |\phi\rangle \otimes A|\chi\rangle, \tag{2.2}$$

which allows us to expand the full wave function in terms of these channels [15]. Note that the eigenfunctions are eigenfunctions in $\mathcal{H}$ rather than just in $\mathcal{H}_2$, which means that channels are also affecting $\mathcal{H}_1$.

4. *Ionisation*, where the electron is separated from both nuclei and ejected into an unbound state.[2]

The dynamics of the system then relates to *coupling* of the channels, which refers to the interaction with respect to the Hamiltonian (see Figure 2.1).

For a complete set of channels, this formulation is equivalent to the Schrödinger equation of the whole system. One, however, usually includes only a finite, very limited set of channels that are most relevant to the process at hand. This choice is a fairly delicate process, since each extra channel adds to the computational effort, but also improves the approximation (see Section 4.1).

2.1.1 The variational method

The idea of coupled channels is best formalised within the variational calculus by Riesz [16]. We recall that within a sub-space $\mathcal{S} \subset \mathcal{H}$ spanned by trial functions $|\psi\rangle = \sum_i \alpha_i |\chi_i\rangle$, minimising the energy functional $E[\psi]$ is equivalent to the weak form of the stationary Schrödinger equation on $\mathcal{S}$:

$$\langle \chi_i | (H - E) | \psi \rangle = 0 \quad \forall \chi_i \tag{2.3}$$

Equivalently, we can say that (2.3) is an approximation of the Schrödinger equation that gets better as the trial space $\mathcal{S}$ expands.

It is fairly obvious that our trial space will be spanned by the electron channels. To proceed, we now need to make some choice of coordinates: we choose the molecular coordinate system $(\vec{R}, \vec{r})$, where $\vec{R}$ is the *internuclear distance* vector from A to B, and $\vec{r}$ is the electron position with respect to the nuclei's centre of mass (see Figure 2.2). These are Jacobi coordinates [17] and the kinetic energy T separates to

$$T = T_R + T_r = \frac{1}{2\mu} P_R^2 + \frac{1}{2} P_r^2, \tag{2.4}$$

with μ being the reduced mass of A and B[3].

Since the electron movement should be "confined" within the channel wave functions χ_i, we make the following ansatz for our trial space [10]:

$$|\psi(\vec{R})\rangle = \sum_j F_j(\vec{R}) |\chi_j(\vec{R})\rangle, \tag{2.5}$$

where we retained the Dirac notation for the electron space ($\vec{r}$) to "conceal" the explicit

[2] In close-coupling calculations, one usually distinguishes direct ionisation and charge exchange into continuum (see Section 2.5.1).

[3] The reduced mass associated with T_r is close to 1 in atomic units because $M_A + M_B \gg m_e$.

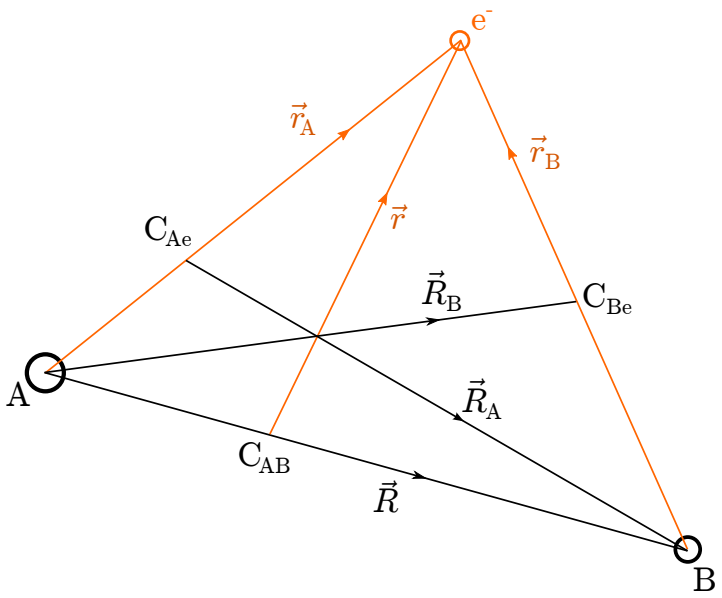


Figure 2.2: Jacobi coordinate systems (up to sign) in di-atomic collisions with one active electron: Asymptotic initial $(\vec{R}_\mathrm{A}, \vec{r}_\mathrm{A})$, asymptotic capture $(\vec{R}_\mathrm{B}, \vec{r}_\mathrm{B})$ and molecular $(\vec{R}, \vec{r})$. C denotes the centres of mass (adopted from [10]).

choice for the electron coordinate[4]. If there was no action T_R on the nuclei's movement, we could think of (2.5) as channels just being parametrised by the internuclear distance, and indeed, this view will become important later on.

Combining equations (2.1) – (2.5), we arrive at a differential equation for the amplitudes $F_j(\vec{R})$ (we omitted the explicit R dependency for clarity):

$$\sum_j \left[\langle \chi_i | \chi_j \rangle (T_R - E) F_j + \frac{1}{\mu} \langle \chi_i | \vec{P}_R | \chi_j \rangle \cdot \vec{P}_R F_j + \langle \chi_i | H | \chi_j \rangle F_j \right] = 0 \tag{2.6}$$

Introducing the channel coupling matrix U, we can rewrite this equation as follows:

$$(T_R - E) F_i = \sum_j U_{ij} F_j \tag{2.7}$$

In the absence of coupling ($U = 0$), this becomes the free particle equation for the amplitudes F_i.

[4] This does not seem to be consistent with the channel definition (2.2) in the case of bound states on either centre. However, a transformation to an asymptotic coordinate system $(\vec{R}_A, \vec{r}_B)$ again separates $\mathcal{S}$ (compare Figure 2.2).

2.1.2 Eikonal approximation and the impact parameter model

Relation (2.6) relates to a large set of coupled second-order differential equations, which are very difficult to solve in its general form. One therefore usually proceeds by exploiting the fact that the nuclei are much heavier than the electron.

One method to do this is by completely separating the core movement from the electron, known as the *Born-Oppenheimer approximation* [15, 17]. Unfortunately, atomic collisions do not qualify as an adiabatic change. At slow collisions, however, this approximation may be used in the impact parameter calculations in the form of molecular channels.

To refine our model, we assume that because of the mass difference we can at least find an effective potential W_j which governs the motion of the nuclei in a certain channel [10]. In first approximation, we can substitute W_j with a *mean potential* $W(R)$. Using this simplification, the movement (2.7) becomes:

$$(T_R + W(R) - E)\bar{F}(\vec{R}) = 0. \tag{2.8}$$

Semi-classical solutions of (2.8) are feasible within the *WKB method*, which is basically the one-dimensional version of the Feynman's path integral formalism. We start by rewriting (2.8) to

$$P_R^2 \bar{F} = 2\mu(E - W)\bar{F} =: p^2 \bar{F}$$

and inserting an exponential ansatz $\bar{F} = \exp(\frac{\imath}{\hbar}S)$. In the classical limit $\hbar \to 0$, we are left with the *eikonal equation*[5]:

$$|\nabla_R S|^2 = p^2 \tag{2.9}$$

which is the square of a Hamilton-Jacobi equation of motion defining all classically possible trajectories at the incident energy E. Interpreting p^2 as classical momentum, we obtain a classical trajectory for the nucleus' movement

$$\nabla_R S =: \vec{p} = \mu \frac{d\vec{R}}{dt}, \tag{2.10}$$

which solves (2.9) trivially. Because we are now considering a single classical trajectory, this is also the point where *time* enters our system: the internuclear separation $\vec{R}$ is no longer part of a stationary scattering problem (compare Figure 2.1), but is "down-graded" to a mere parametrised curve $\vec{R}(t)$ keeping track of the nucleus' location (compare Figure 2.3).

Finally, we assume that the full amplitudes F_i are slowly varying around the mean

[5] From Greek "image", this equation provides a link between classical ray mechanics and wave mechanics, because it models Fermat's principle in terms of waves.

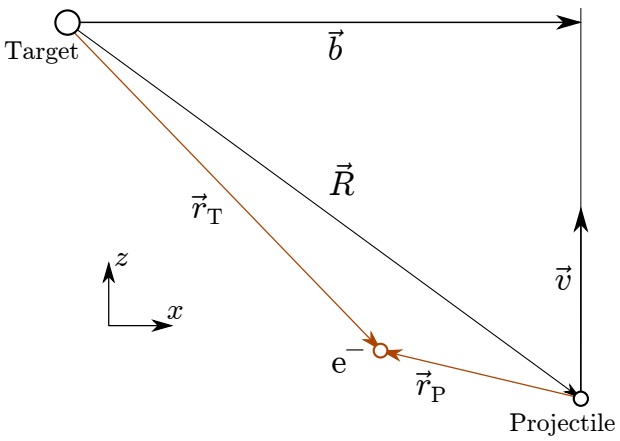


Figure 2.3: Collision set-up in the impact parameter model, where the nuclei are assumed to travel on straight lines (i. e., $\vec{v} = \text{const.}$). For simplification, we assumed that the coordinate system is stationary with respect to the target ($\vec{v}_{\mathrm{T}} = 0$).

amplitude $\bar{F}$ and therefore make the ansatz:

$$F_i = a_i(t)\bar{F}(\vec{R}) \tag{2.11}$$

In order to insert this into (2.6), we neglect P_R^2, because it is of the order $\hbar^2$ and use (2.10) to express ∇_R as time derivatives. After using (2.8) and performing a considerable amount of re-ordering, we obtain [18]

$$\sum_j \langle\chi_i(t)|H_{\mathrm{CC}} + (V_{\mathrm{N}} - W) - \imath\frac{\partial}{\partial t}\left(a_j(t)\,|\chi_j(t)\rangle\right) = 0 \tag{2.12}$$

with the close-coupling Hamiltonian

$$H_{\mathrm{CC}} = T_r + V_{\mathrm{T}} + V_{\mathrm{P}}. \tag{2.13}$$

We immediately see that (2.12) is a weak time-dependent Schrödinger equation on states of the form $|\psi\rangle = a_i|\chi_i\rangle$ (see Section 2.2). This reflects the fact that we changed from a constant flow of particles — the stationary problem (2.6) — to a the classical scattering process of one single atom. With some hand-waiving, therefore, we could have spared ourselves the whole derivation by just stating that the nucleus moves on a classical trajectory and the electron is treated in a quantum-mechanical fashion.

Note that we excluded the inter-nuclear potential $V_{\mathrm{N}}(R)$ and the mean nuclei potential $W(R)$ from the close-coupling Hamiltonian (2.13). This is because as purely time-

dependent contributions, they just prompt a phase factor

$$|\chi_i(t)\rangle = \exp\left(-\imath \int^t d\tau [V_{\mathrm{N}}(R(\tau)) - W(R(\tau))]\right) |\bar{\chi}_i(t)\rangle$$

Introducing the electronic wave function

$$|\psi(t)\rangle = a_i(t)|\bar{\chi}_i(t)\rangle, \tag{2.14}$$

the equations for the transformed amplitudes read:

$$\langle\bar{\chi}_i(t)|(H_{\mathrm{CC}} - \imath\partial_t)|\psi(t)\rangle = 0 \quad \forall\bar{\chi}_i \tag{2.15}$$

2.1.3 Straight line trajectories

While the wave function has "swallowed" all R-dependent potentials in the Hamiltonian, these terms still enter through the form of the trajectory $\vec{R}(t)$. However, for fast or distant collisions, we can simplify these trajectories even further.

Looking at classical scattering of doubly charged ions, one finds an upper bound for the scattering angle of

$$|\theta|[^\circ] \leq \frac{1}{E\,[\mathrm{keV}] \cdot b\,[a_0]}$$

in the observational frame [12]. This means that for $E > 1\,\mathrm{keV}$ and $b > a_0$, the deviation from the straight line is less than 1°. This motivates us to approximate the nucleus' trajectory by a straight line

$$\vec{R}(t) = \vec{b} + \vec{v}t \tag{2.16}$$

where $\vec{v} = \text{const}$ is the incidence velocity and $\vec{b}$ again is the impact parameter (see Figure 2.3). This is equivalent to neglecting the inter-nuclear forces for the heavy-particle movement ($W = 0$).

2.1.4 Boundary conditions and cross sections

In order to compute the partial cross sections σ_i, we need to pay attention to the *asymptotic behaviour* of our system. For our electronic wave function (2.14) this means that the system enters a defined superposition of the channels[6]:

$$\lim_{t\to\pm\infty} a_i(t) =: a_i^{\pm}. \tag{2.17}$$

[6]For potentials of infinite range $\Omega(r^{-1})$, like the Coulomb potential, this definition is problematic, because their long-range scattering induces logarithmic behaviour in the amplitude vector. By the use of a phase transformation [10], however, we can "swallow" this and restore the form (2.17).

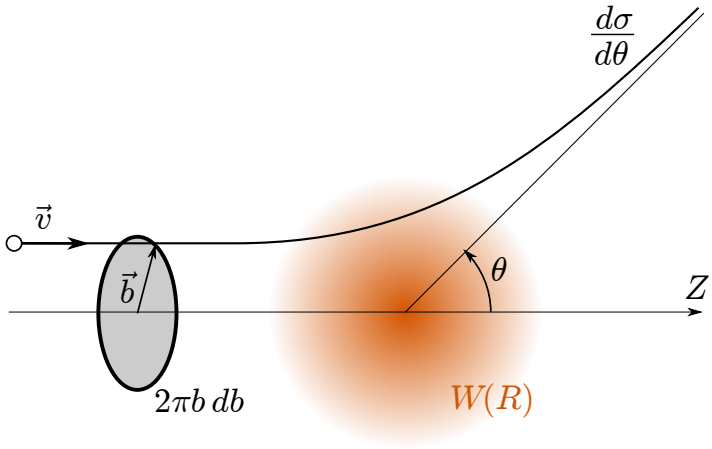


Figure 2.4: Classical scattering of particles off an effective, spherically symmetrical potential $W(R)$ (represented by gradient). We see that the impact velocity v and the impact parameter b determine the scattering angle θ.

Equation 2.17 essentially is a constraint on the channel wave functions $\bar{\chi}_i$. Satisfaction of proper boundary conditions constitutes a critical point in the close-coupling method and is usually established by use of so-called electron translation factors (see Section 2.4.1).

One can now derive explicit expressions by use of scattering theory [19]: at large distances, we can separate the amplitudes of an open channel ($K_i^2 > 0$) into an incidence plane wave and a asymptotically free scattered wave:

$$\lim_{R\to\infty} F_i(\vec{R}) = \delta_{ij}\exp(\imath\vec{K}_j\vec{R}) + f_{ij}(\Omega)\frac{\exp(\imath K_i R)}{R} + o(R^{-1}), \tag{2.18}$$

where j is the initially populated channel, $\vec{K}_j = \mu\vec{v}$ is the corresponding momentum and f_{ij} are the scattering amplitudes depending on the scattering solid angle Ω. The partial cross section $\sigma_{ij} \equiv \sigma_{i\leftarrow j}$ is then given by

$$\sigma_{ij} = \int_\Omega d\Omega\,\frac{d\sigma_{ij}}{d\Omega} = \int_\Omega d\Omega\,|f_{ij}(\Omega)|^2.$$

Instead of doing the maths explicitly [10], we use a classical heuristic: considering the scattering of classical particles off a spherically symmetric potential, we see that impact parameter b and impact velocity v determine the scattering angle Ω (Figure 2.4) [15]. Since the area element dA of the incidence beam is $2\pi b\,db$ and the scattering angle is small, we "guess" the following expression for the partial cross sections (with the incidence channel again being j):

$$\sigma_{ij} = \int_0^\infty 2\pi b\,db\,|a_i^+ - \delta_{ij}|^2, \tag{2.19}$$

which agrees with the explicit computation. The subtraction of δ_{ij} relates to the separation of the incidence wave in (2.18).

Unfortunately, the expansion (2.18) is only valid for potentials of *finite range* $V \in o(1/r)$. This is satisfied for neutral atoms and lowly charged ions: the electrons "shield" the charge and reduce the asymptotic behaviour to the Yukawa potential $\exp(-\alpha r)/r$ [16]. Unshielded charges, i.e., the Coulomb potential of fully stripped ions, on the other hand are of infinite range. As a result, the amplitudes are superimposed the long-range Coulomb scattering amplitude f_C [10]:

$$f_{ij}(\Omega) \mapsto f_{ij}(\Omega) + \delta_{ij} f_\mathrm{C}(\Omega). \tag{2.20}$$

The modifications (2.19) and (2.20) are only relevant for the elastic scattering cross section and can therefore be omitted if we are only interested in charge exchange and ionisation cross sections.

2.2 Schrödinger equation on a finite subspace

We now want to compute solutions of the weak Schrödinger equation (2.15). This section, however, can be applied to any time-dependent problem.

It is not possible to take a complete set of channels and solve the Schrödinger equation in the complete infinite-dimensional Hilbert space $\mathcal{H}$. As a result, we approximate the Hilbert space by a finite-dimensional subspace $\mathcal{S}$ ("truncated/trial Hilbert space"), which is spanned by n basis states $|\chi_i\rangle$:

$$\mathcal{S} = \mathcal{L}\left\{|\chi_i\rangle\right\}_{i \leq n} \subset \mathcal{H}$$

The crux of the matter is to compile a suitable basis for a given problem, and no definitive algorithm for this task has been found yet. Some types of basis states, however, proved especially useful in the calculation of cross sections:

1. Atomic orbitals (AO), which are bound states on either centre, model excitation and charge exchange (see Section 2.4) at higher energies;
2. Molecular orbitals (MO) resemble transiently formed molecules in slow collisions [11];
3. United atoms (UA) pseudo-states are bound states to the united-atom limit and serve as a "bridge" between AO and MO expansions (see Section 2.5); and
4. Sturmian and Hylleraas basis functions [11].

The quality of the results greatly depends on how well the basis states model the physical channels in the scattering problem (see also Section 4.1).

Assuming that the wave function ψ can be expanded into the basis states

$$|\psi(t)\rangle = \sum_{i=1}^{n} a_i(t)\,|\chi_i(t)\rangle, \tag{2.21}$$

we can rewrite the Schrödinger equation $\langle\chi_i|(H - \imath\partial_t)|\psi\rangle = 0$ as first-order ordinary differential matrix equation [11]:

$$M \cdot \vec{a} = \imath S \cdot \frac{d}{dt}\vec{a} \tag{2.22}$$

where we introduced the *overlap matrix* S, which defines an inner product on the space:

$$S_{ij} := \langle\chi_i(t)|\chi_j(t)\rangle. \tag{2.23}$$

S is necessarily a positive definite Hermitian matrix and simplifies to the unity matrix $\mathbb{1}$ for orthonormal systems[7]. It might seem cumbersome at first to work with a non-orthogonal basis, however, it is usually necessary because the basis states should correspond to physical scattering channels (see also Section 2.4). In most cases, however, one demands at least that the states are orthonormal at asymptotic distances

$$\lim_{t\to\pm\infty} S = \mathbb{1}$$

to allow for a clear separation of the channels.

We have also introduced the *coupling matrix* M, which is the action of the Schrödinger equation on the basis states

$$M_{ij} := \langle\chi_i(t)|\left(H - \imath\frac{\partial}{\partial t}\right)|\chi_j(t)\rangle \tag{2.24}$$

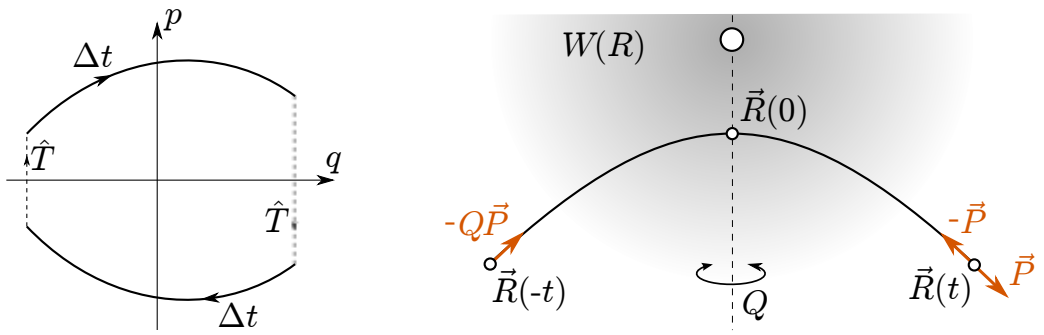
In general, this matrix is *not Hermitian* ($M \neq M^\dagger$): using (2.23) and (2.24) one can easily show that [21]

$$M^\dagger - M = \imath\frac{\partial}{\partial t}S. \tag{2.25}$$

On first glance, this might seem counter-intuitive, on closer examination, however, we see that non-Hermicity (2.25) is due to the fact that we chose an "odd" basis, i. e., we expanded our wave function (2.21) into states related by a time dependent inner product S. This also means that probability conservation relates to [20]

$$\langle\psi|\psi\rangle = \vec{a}^\dagger S\vec{a} = \text{const.} \tag{2.26}$$

[7] The overlap matrix is frequently called N, a convention followed by (e. g.) Fritsch and Lin [11]. Similarly, the coupling matrix is sometimes termed H [20]. We decided against this notation to avoid confusion with normalisation constants N_{nl} and the Hamiltonian matrix H, respectively.



(a) Time reversal as motion reversal $p \mapsto -p$ in a one-dimensional phase space.

(b) Formal time reversal $t \mapsto -t$ of a defined classical trajectory by motion reversal plus reflection Q through the vector of closest approach $\vec{R}(t=0)$.

Figure 2.5: On the concept of time reversal and motion reversal in (a) time independent and (b) explicitly time dependent systems.

as opposed to $\langle\psi|\psi\rangle = |\vec{a}|^2$ in the case of an expansion into a stationary basis set.

2.3 Time reversal and detailed balance

Time symmetry is a fundamental property of classical mechanics[8] and also satisfied by many quantum-mechanical processes, including the collisions of spin-less particles. It actually refers to the fact that under the inversion of the canonical momenta $p \mapsto -p$ (see Figure 2.5a) a given trajectory is followed "backwards" (*motion reversal*), which is equivalent to the "naïve" inversion of the time coordinate $t \mapsto -t$ (*time reversal*) if the Hamiltonian is not explicitly time-dependent.

In analogy to the classical case, motion reversal in quantum mechanics is modelled by the *motion reversal operator* $\hat{M}$, which is defined by its action on the fundamental operators[9]:

$$\hat{M}X_i\hat{M}^\dagger = X, \quad \hat{M}P_j\hat{M}^\dagger = -P_j, \quad \hat{M}J_k\hat{M}^\dagger = -J_k \tag{2.27}$$

In order to preserve the canonical commutator $[X_i, P_j] = \imath\delta_{ij}$, $\hat{M}$ must be semi-linear, i. e., it conjugates scalars when passing through them: $\hat{M}a = a^*\hat{M}$. For time-independent systems ($\dot{H} = 0$) this again links us back to time reversal, because $\hat{M}$ changing the sign of $\imath$ is equivalent to changing the direction of time evolution [22]

$$\hat{M}(\mathbb{1} - \imath H\delta t) = (\mathbb{1} + \imath H\delta t)\hat{M} \tag{2.28}$$

[8] This statement, and time symmetry in general, seems intuitive on the surface but is actually very delicate. I therefore dedicate this somewhat lengthy section to clarifying concepts and subtleties.

[9] Note that most authors call this the *time reversal operator* T or $\mathcal{T}$ because of the correspondence mentioned earlier. I will, however, avoid this terminology because it may confuse the reader when we move to explicitly time-dependent systems.

if and only if H is invariant under motion reversal

$$[\hat{M}, H] = 0 \quad \Leftrightarrow \quad \hat{M} H \hat{M}^\dagger = H.$$

By collecting all operators in (2.28) on one side (neglecting terms of higher order in δt), we indeed arrive at the "closed loop" in Figure 2.5a, thus establishing the link between time reversal and motion reversal.

Unfortunately, this analogy breaks down in explicitly time-dependent systems like the impact parameter formulation. The formal substitution $t \mapsto -t$ is still possible, but it immediately fixes the – usually arbitrary – time origin. As a result, time reversal degenerates to a formal oddity unless $t = 0$ can be assigned physical distinctiveness.[10] If such a point can be found, however, then we can also think of time reversal as motion reversal *plus* the formal "mirroring" of the system so that it advances to $t = 0$ from the "other side". One can now try to amend time reversal with a spatial transformation Q representing this mirroring to restore the relation with motion reversal.

As a highly relevant example, we consider the classical trajectory $\vec{R}(t)$ of a particle scattered off a radial potential $V(R)$ (see Figure 2.5b): we choose the vector of closest approach as the time origin of the trajectory, because it is symmetric around this point [26], a direct consequence of the spherical symmetry of the Hamilton function. In this case time reversal translates to motion reversal ($\frac{d}{dt}R(-t) = -\frac{d}{dt}R(t)$) plus mirroring around the spatial symmetry Q of the trajectory.

To formalise this in quantum mechanical systems, we introduce the semi-linear *time reversal operator*

$$\hat{T} := Q\hat{M}, \tag{2.29}$$

where Q is a time-independent Unitary transformation. The defining relations (2.27) are thus amended to:

$$\hat{T} X_i \hat{T}^\dagger = Q X Q^\dagger, \quad \hat{T} P_j \hat{T}^\dagger = -Q P_j Q^\dagger, \quad \hat{T} J_k \hat{T}^\dagger = -Q J_k Q^\dagger \tag{2.30}$$

In the case of explicitly time-independent systems, we can set $Q = \mathbb{1}$ and recover the correspondence $\hat{T} = \hat{M}$. Studying (2.29), we conclude that time symmetry is a weaker form of motion symmetry, one that requires the concept of time to be specifically crafted to meet the criteria of a symmetry.

It is important to note that the time reversal operator does not actually perform the substitution $t \mapsto -t$: in classical and non-relativistic quantum mechanical systems, time is merely a parameter and is therefore out of the scope of Hilbert space operators. However, since these two are related symmetries, Wigner's theorem [19] tells us that

[10] This fact (along with energy dissipative systems) recently triggered a philosophical discussion on the precise interpretation of time reversal symmetry and the question whether classical mechanics actually has time symmetry. [23, 24, 25]

the formal transformation and its quantum mechanical "model" $\hat{T}$ only differ by a phase factor ($\mathbb{T}$ is the circle group)

$$\eta: \quad \mathcal{H} \to \mathbb{T}; \quad |\psi\rangle \mapsto \exp(\imath\phi(\psi)). \tag{2.31}$$

This additional degree of freedom is due to the fact that phase factors in the transformed amplitudes are unobservable. We also conclude that $\hat{T}$ must be *antiunitary*:[11]

$$\langle\phi|\hat{T}^\dagger\hat{T}|\psi\rangle = \langle\phi|\psi\rangle^* = \langle\psi|\phi\rangle. \tag{2.32}$$

It can easily be shown [27], that any such operator can be written as the product of a unitary operator U and the complex conjugation operator $\hat{K}$:

$$\hat{T} = U\hat{K}. \tag{2.33}$$

For motion reversal on spin-less systems, we choose $\hat{K}$ such that it yields the conjugate complex of the wave function in configuration space: $\hat{K}\psi(\vec{r},t) = \psi^*(\vec{r},t)$.[12] In this convention, $\hat{K}$ already satisfies the classical motion reversal formulae (2.27) and, as a result, already conveys motion reversal. Comparing the explicit forms (2.29) and (2.33), we find that $U = Q$.

2.3.1 Time reversal formulae

We have now enough machinery at our hands to approach the case of our semi-classical description of the collision process. While the full quantum-mechanical problem (2.1) is invariant under motion reversal (and therefore also time reversal, since the problem is not explicitly time-dependent)

$$\hat{M}H\hat{M}^\dagger = H,$$

we want to know if this property carries over to our eikonal approximation.

Similar to the previous section (compare Figure 2.5b), the classical trajectory of the nucleus is generated by a spherically symmetrical potential $W(R)$ (2.8). As a result,

[11] Studying equation (2.32), a more intuitive way might be to perceive time reversal — and semilinear operators in general — as functions that transform a vector into a co-vector, i. e.,

$$\hat{T}: \quad \mathcal{H} \to \mathcal{H}^*; \quad |\psi\rangle \mapsto \langle T\psi|, \quad a|\phi\rangle + b|\psi\rangle \mapsto \langle T\psi|a^* + \langle T\psi|b^*$$

therefore exchanging initial and final state in transition amplitudes [22]. Unfortunately, this notation requires non-trivial alterations to the Dirac notation.

[12] Since complex conjugation cannot be canonically defined on a Hilbert space, one must designate a basis (ϕ_i) together with $\hat{K}$ on which it acts trivially, such that for any $|\psi\rangle = a_i|\phi_i\rangle$ we get

$$|\psi^*\rangle := \hat{K}|\psi\rangle = a_i^*|\phi_i\rangle.$$

Therefore, we can say that $\hat{K}$ refers to a family of "simplest" semilinear operators.

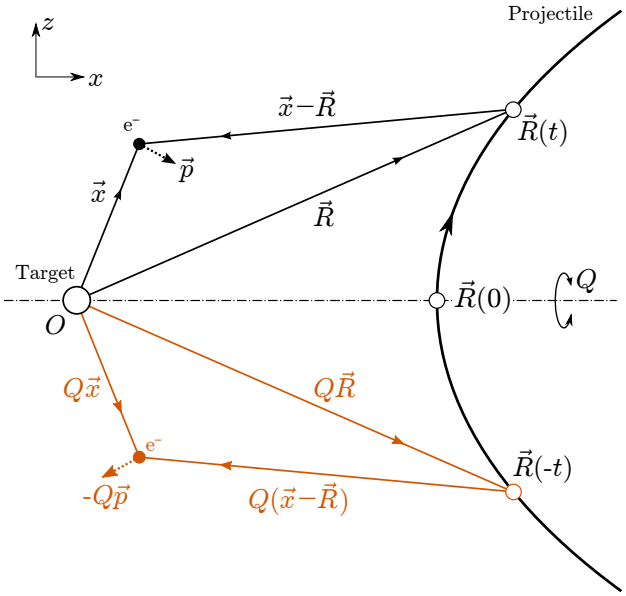


Figure 2.6: The effect of time reversal on semi-classical scattering is a reflection Q (here through the x-y-plane), provided that the trajectory is generated by a Q-symmetric, time independent Hamilton function and $t = 0$ marks the closest approach. We see that this way, all distances are preserved.

time reversal also relates to mirroring the classical trajectory around its line of closest approach. However, we do not mirror the internuclear distance vector $\vec{R}$, since it again just "parametrises" the Hilbert space, but instead mirror the electron coordinate $\vec{x}$, which has just the same effect (see Figure 2.6).[13] Together with motion reversal, this makes the close-coupling Hamiltonian time symmetric (see also [21]):

$$\hat{T} H_{\mathrm{CC}} \hat{T}^\dagger = H_{\mathrm{CC}} \tag{2.34}$$

An interesting conclusion of this is that motion symmetry of the complete system degrades to time symmetry of its eikonal approximation as long as such a symmetry is inherent in the classical trajectory.

To make use of (2.34) in the evaluation of the coupled channels equations, we must first look at the effect of time reversal on wave functions. In the following, we assume that Q is mirroring around the z axis and that the basis states can be separated into a (real) radial part $R_{nl}(r)$ on their respective centres and their angular part are spherical harmonics Y_l^m.[14] The time-reversed wave function can then be expressed as the conjugate complex

[13] This bears some resemblance with active and passive transformations, however note that both the active Q and passive part $Q^\dagger$ act on the electron and do not alter the time parameter, which lies outside the scope of Hilbert space operators.

[14] The same works for molecular orbitals – see [21]. For real spherical harmonics Υ_l^m, the formulae are the same.

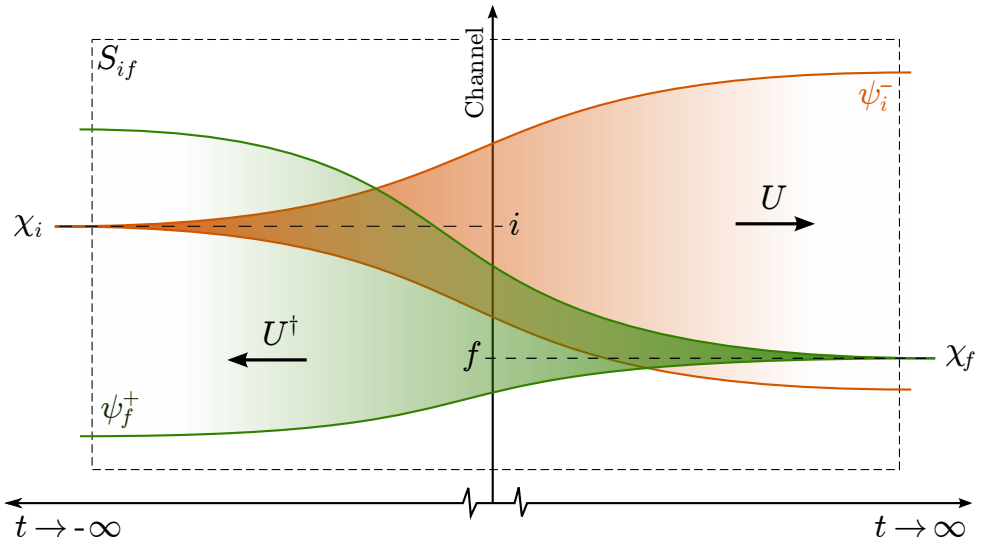


Figure 2.7: Schematic visualisation of the scattering matrix element S_{if} as overlap of ψ_i^- (evolving from the asymptotic initial state χ_i though U) and ψ_f^+ (evolving *to* the asymptotic final state χ_f though U, or, equivalently, evolving *from* the final state though $U^\dagger$) at any given time (vertical line).

of the original, because the mirroring operation just prompts a phase:

$$QY_l^m Q^\dagger = (-1)^{l+m} Y_l^m$$

Straightforward insertion of the time reversal operator $\hat{T} = Q\hat{K}$ (2.29) into the overlap and coupling matrix yields [21]:

$$M_{ij}(-t) = \langle\chi_i|\hat{T}^\dagger(H - \imath\partial_t)\hat{T}|\chi_j\rangle = \epsilon_i\epsilon_j M_{ij}^*(t) \tag{2.35a}$$
$$S_{ij}(-t) = \langle\chi_i|\hat{T}^\dagger\hat{T}|\chi_j\rangle = \epsilon_i\epsilon_j S_{ij}^*(t) \tag{2.35b}$$

where $\epsilon_i := (-1)^{l_i+m_i}$ is the phase factor prompted by the spherical harmonic associated with the i-th basis state and is not to be confused with the phase (2.31) from Wigner's theorem. This essentially means that we only have to compute half of the overlap is and coupling matrices and can deduce the other half by equations (2.35).

2.3.2 Detailed balance

Time reversal can also be used in the computation of the scattering matrix and is related to the concept of detailed balance.

We denote the normal "forward" wave function by ψ^- and its time-reversed "backward" version by $\psi^+ := \hat{T}\psi^-$.[15] Furthermore, we let ψ_i be the state that evolved from the i-th

[15] We use this definition because while for the dynamics of the system, time reversal and the substitution

channel, i. e.,

$$\lim_{t\to\pm\infty} |\psi_i^{\pm}(t)\rangle = |\chi_i\rangle.$$

It is now straightforward to see [22] that the scattering matrix element S_{fi} from channel i to channel f can be expressed as the overlap (see Figure 2.7)

$$S_{fi} := S_{f\leftarrow i} = \langle\psi_f^+(t)|\psi_i^-(t)\rangle$$

for any given time t. This can be used to compute the total scattering matrix by approaching $t = 0$ from "both sides".

One can also show that in the impact parameter model, the transition probabilities $P_{fi} = |S_{fi}|^2$ satisfy *detailed balance* [21]:

$$P_{fi} = P_{if}.$$

This means that the probability for a process $f \leftarrow i$ and its "inverse" $i \leftarrow f$ are the same, which has important consequences in thermodynamics.

2.4 The atomic orbitals basis set

Perhaps the most natural choice for a basis set are bound states on either centre: at asymptotic distances, if no ionization has occurred, the electron will reside in a bound state on the target if the electron has been captured or else remain in a bound state on the projectile.

As a result, we can further separate our finite subspace $\mathcal{S}$ into basis states $|\chi_i\rangle_{\mathrm{T}}$ for the target space $\mathcal{S}_{\mathrm{T}}$ and basis states $|\chi_i\rangle_{\mathrm{P}}$ for the projectile space $\mathcal{S}_P$:

$$\mathcal{S} = \mathcal{S}_{\mathrm{T}} \oplus \mathcal{S}_{\mathrm{P}}; \quad \mathcal{L}\left\{|\chi_i\rangle\right\} = \mathcal{L}\left[\left\{|\chi_i\rangle_{\mathrm{T}}\right\} \cup \left\{|\chi_i\rangle_{\mathrm{P}}\right\}\right].$$

which are called *atomic orbitals* (AO) for obvious reasons. The time-dependence of these states is given canonically by the eigenenergy for the atomic Hamiltonian H_T or H_P on the respective centre:

$$|\chi_i(t)\rangle = \exp(-\imath\varepsilon_i t)|\chi_i\rangle \quad \text{where} \quad (H_{\mathrm{T}} \oplus H_{\mathrm{P}})|\chi_i\rangle = \varepsilon_i|\chi_i\rangle \tag{2.36}$$

This means that we can split the probability amplitude vector $\vec{a} = (\vec{a}_{\mathrm{T}}, \vec{a}_{\mathrm{P}})^{\mathrm{T}}$ which further separates coupling matrix (2.24) and overlap matrix (2.23) into target and pro-

$t \mapsto -t$ are equivalent, for wave functions (as well as for classical particles) this is *not* the case: even though the time-inverse of a wave function $\psi(t)$ is often colloquially denoted by $\psi(-t)$, these two operations yield completely different entities. More specifically, one cannot use time reversal to "predict" the future of a wave function: $\hat{T}\psi(-1) \neq \psi(1)$. One instead gets a wave function that evolved backwards from the same initial conditions at $t = +\infty$ (cf. Figure 2.7).

jectile parts:

$$S = \begin{pmatrix} \mathbb{1}^{\mathrm{T}} & S^{\mathrm{TP}} \\ (S^{\mathrm{TP}})^{\dagger} & \mathbb{1}^{\mathrm{P}} \end{pmatrix} \quad \text{and} \quad M = \begin{pmatrix} M^{\mathrm{TT}} & M^{\mathrm{TP}} \\ M^{\mathrm{PT}} & M^{\mathrm{PP}} \end{pmatrix} \tag{2.37}$$

with the unit matrix $\mathbb{1}^{\mathrm{A}}$ on either centre A, which results from the fact that we assumed that the states are orthonormal on either centre:

$$_{\mathrm{A}}\langle \chi_i | \chi_j \rangle_{\mathrm{A}} = \delta_{ij} =: \mathbb{1}^{\mathrm{A}}$$

Again note that M^{PT} and M^{TP} are not necessarily related by Hermitian conjugation (cf. Equation 2.25).

We classify the parts of M and S in (2.37) as *two-centre*, if they relate states centred at different nuclei, and as *one-centre* otherwise. The explicit expressions of these parts are:

$$S_{ij}^{\mathrm{AB}} = {}_{\mathrm{A}}\langle \chi_i(t) | \chi_j(t) \rangle_{\mathrm{B}} \qquad \ldots \text{ Two-centre overlap} \tag{2.38a}$$

$$M_{ij}^{\mathrm{AA}} = {}_{\mathrm{A}}\langle \chi_i(t) | (H - \imath\partial_t) | \chi_j(t) \rangle_{\mathrm{A}} \qquad \ldots \text{ One-centre coupling} \tag{2.38b}$$

$$M_{ij}^{\mathrm{AB}} = {}_{\mathrm{A}}\langle \chi_i(t) | (H - \imath\partial_t) | \chi_j(t) \rangle_{\mathrm{B}} \qquad \ldots \text{ Two-centre coupling} \tag{2.38c}$$

where the indices i and j run over the respective segments of the matrices.

2.4.1 Electron translation factor

For physically reasonable results, asymptotic states must satisfy the Schrödinger equation at infinite separation. For bound states χ_i on either centre A, this means:

$$\left(\frac{1}{2}\vec{P}_{\mathrm{A}}^{\,2} + V_{\mathrm{A}} - \imath\frac{\partial}{\partial t} \right) |\chi_i(t)\rangle_{\mathrm{A}} = 0. \tag{2.39}$$

It is important to note that the Schrödinger equation is *globally* invariant with respect to translation $\vec{r} \mapsto \vec{r} - \vec{R}$. In our case, however, the presence of the second nucleus breaks this symmetry. We need to fix a coordinate system for the momentum of the electron in equation (2.39) as a result. Because the nuclei travel on classical trajectories $\vec{R}(t)$, this corresponds to a transformation

$$\vec{r}_{\mathrm{A}} = \vec{r} - \vec{R} \quad \text{and} \quad V_{\mathrm{A}} = V + \frac{\vec{v}^{\,2}}{2}, \tag{2.40}$$

where $\vec{v} = \frac{d}{dt}\vec{R}$ is the velocity of the nucleus A with respect to our coordinate system and $\vec{v}^{\,2}/2$ is the kinetic energy of the electron added by this movement. The transformed Schrödinger equation for *moving states* $\tilde{\chi}_i$, which are "dragged" by their centre, now

reads:

$$\left(\frac{1}{2}(\vec{P}-\vec{v})^2+\frac{\vec{v}^2}{2}+V(\vec{r})-\imath\frac{\partial}{\partial t}\right)|\tilde{\chi}_i(t)\rangle=0 \tag{2.41}$$

In order to find a solution for $\tilde{\chi}_i$, we first note that the transformation (2.40) has a similar effect on the Schrödinger equation as Gauge transformation of a electromagnetic potential. We now remember that the Schrödinger equation has a *local* $U(1)$ gauge symmetry: we introduced a phase factor $\exp(\imath\Lambda)$ to "swallow" that gauge freedom [19]. This motivates us to make the following ansatz:

$$|\tilde{\chi}_i(t)\rangle=\exp(\imath\Lambda(\vec{r},t))|\chi_i(t)\rangle \tag{2.42}$$

Inserting this into equations (2.39) and (2.41), we get the following condition for the phase factor:

$$\left(\frac{1}{2}(\vec{P}-\vec{v})^2+\frac{\vec{v}^2}{2}-\imath\frac{\partial}{\partial t}\right)\exp(\imath\Lambda(\vec{r},t))=\exp(\imath\Lambda(\vec{r},t))\left(\frac{1}{2}\vec{P}^2-\imath\frac{\partial}{\partial t}\right)$$

It can easily be shown that this equation is solved by a plane wave of the form[16]

$$\exp(\imath\Lambda(\vec{r},t))=\exp\left(\imath\vec{v}\vec{r}-\imath\int^t d\tau\,\frac{\vec{v}^2(\tau)}{2}\right). \tag{2.43}$$

This phase is commonly called *electron translation factor* (ETF) as it compensates for the movement of the state's origin with respect to our coordinate system. From a geometrical point of view, this factor restores the covariance of the equation with respect to transformations (2.40) [12]. From a physical perspective, on the other hand, the factor fits the solution onto the boundary conditions at infinite separation of the nuclei.

The form of the ETF is not by chance: interpreting the plane wave as wave function, it is a solution to the Schrödinger equation with the kinetic energy potential $V=v^2/2$. As a result, we can think of the moving state $\tilde{\chi}_i$ as a superposition of the asymptotic solution χ_i and a plane wave generated by the kinetic energy of the nucleus moving through the scattering set-up.

Another explanation of the ETFs is more closely linked to why we call it a *translation* factor: the momentum operator $\vec{P}$ is the generator of translations, i. e. it represents an infinitesimal spacial propagation. Finite translations can be obtained by formal exponentiation $\exp(\imath\vec{P}\vec{r})$, which also explains the fact that the ETF is a phase factor.

To simplify our equations and make them independent of the coordinate system, we formalise the ETF as the linear *electron translation operator* $F(\vec{v})$ that satisfies:

[16] The notation $\int^t$ reflects the fact that a change of the time origin, i.e. the fixing of the lower integral boundary, only yields a global phase factor, which is not observable.

$$\left(T + \frac{v^2}{2} - \imath\frac{\partial}{\partial t}\right) \cdot F(\vec{v}) = F(\vec{v}) \cdot \left(T_{\vec{v}} - \imath\frac{\partial}{\partial t}\right) \tag{2.44a}$$

$$F^\dagger(\vec{v}) = F(-\vec{v}) \tag{2.44b}$$

$$F(\vec{v} + \vec{w}) = F(\vec{w})F(\vec{v}), \tag{2.44c}$$

where $T = \vec{P}^2/2$ is the kinetic energy term and $T_{\vec{v}}$ is the corresponding kinetic energy in a system that is moving with a speed $\vec{v}$ against the system of T.

Though being necessary and convenient for most calculations, the inclusion of ETFs amplifies the so-called *sign problem*, a fundamental numerical concern in computational quantum mechanics (see Section 3.4). For this and other reasons, collision systems have been investigated without ETFs, mostly within the Perturbed Stationary State (PSS) model [10, 11].

2.4.2 Coupled-channel equations

The use of electron translation factors (2.44) together with the atomic-orbital basis states allows to find simple expressions for the matrix elements (2.37).

When we insert the close-coupling Hamiltonian (2.13) into the matrix elements (2.38) and use (2.39), we get[17]

$$S_{ij}^{\mathrm{AB}} = {}_{\mathrm{A}}\langle\chi_i|F(\vec{v}_{\mathrm{B}} - \vec{v}_{\mathrm{A}})|\chi_j\rangle_{\mathrm{B}} \cdot T_{ij} \tag{2.45a}$$

$$M_{ij}^{\mathrm{AA}} = {}_{\mathrm{A}}\langle\chi_i|V_B|\chi_j\rangle_{\mathrm{A}} \cdot T_{ij} \tag{2.45b}$$

$$M_{ij}^{\mathrm{AB}} = {}_{\mathrm{A}}\langle\chi_i|V_A F(\vec{v}_{\mathrm{B}} - \vec{v}_{\mathrm{A}})|\chi_j\rangle_{\mathrm{B}} \cdot T_{ij}, \tag{2.45c}$$

where we introduced

$$T_{ij} := \exp(\imath(\varepsilon_i - \varepsilon_j)t) \tag{2.46}$$

as a short-hand notation for the time-dependent part common to all matrix elements.

Note that because we included ETFs into the matrix elements, the elements only depend on velocity difference $\vec{v}_{\mathrm{B}} - \vec{v}_{\mathrm{A}}$ and not on the particular choice of our coordinate system, thus re-establishing translation and Galilean invariance.

Nevertheless, it is convenient to fix our coordinate system to be centred on the target nucleus, which means that the projectile velocity $\vec{v}_{\mathrm{P}}$ equals the impact velocity $\vec{v}$ and $\vec{v}_T = 0$. As a result, only basis states on the projectile are to be multiplied with an ETF:

$$|\tilde{\chi}_i\rangle_{\mathrm{P}} = F(\vec{v})|\chi_i\rangle_{\mathrm{P}}, \quad |\tilde{\chi}_i\rangle_{\mathrm{T}} = |\chi_i\rangle_{\mathrm{T}}$$

[17] Note that due to the fact the $(H - \imath\partial_t)$ is not an Hermitian operator (see Equation 2.25), you cannot plainly act with (2.39) to the left. This fixes V_{A} in the two-centre coupling matrix elements (2.45c).

2.5 Pseudo-state expansions

While elastic scattering at high distances can be modelled well using atomic orbitals [11], these expansions begin to diverge from the experiment when modelling lower impact parameters and intermediate impact energies. Moreover, since bound states are used, the method neglects any ionisation channels. Molecular orbitals are more suited for this task, but unfortunately suffer from some fundamental modelling problems [10, 11].

Upon examining hydrogen – helium scattering, Wiltes and Gallaher looked at the united-atom limit, where both nuclei are located at the same position. They found that even all Hydrogen atomic orbitals only account for 76% of the overlap with the ground state of doubly charged Helium He^{2+} [20]. This is because atomic orbitals by design have another limiting factor: they do not form a complete orthonormal system, even though the underlying associated Laguerre polynomials do[18]. This is a result of the scaling of the argument of the radial coordinate in the wave function

$$\langle r|Znlm\rangle \propto \rho^l e^{\rho/2} L_{n-l-1}^{(2l+1)}(\rho) \quad \text{with} \quad \rho := 2Zr/n$$

by n^{-1}. Therefore, even a complete set of atomic orbitals is insufficient for modelling collisions at a closer distance.

To overcome this specific problem, Cheshire *et al.* [29] constructed orthogonal states to the AO states with the same asymptotic behaviour as the $n = 1, 2$ atomic orbitals[19]. They found that the inclusion of these *pseudo-states* models the helium ground state very well, yielding an overlap of almost 99.7% in the united-atom limit.

Anderson *et al.* [30] took up the idea of the united-atom limit and proposed the use of a third centre, located on the connecting line between both nuclei[20] (see Figure 2.8). For this new centre, they suggested the use of bound states to both nuclei combined $|(Z_1 + Z_2)nlm\rangle$. These *united-atom states* (UA) model the united-atom limit, but can also be seen as a first-order approximation to molecular orbitals. Unfortunately, the introduction of another centre drastically adds to the computational complexity of the interaction matrices.

[18] More precisely, for fixed q, the associated Legendre polynomials $\{L_p^{(q)}\}_{p\geq 0}$ by definition form a orthogonal (but not orthonormal) system on the space of Lebesgue square-integrable functions $L^2[0,\infty)$ with respect to the weighted measure $d\mu(x) = x^q \exp(-x)\, dx$, and can be shown to be complete [28].

[19] The reason for this was mainly to reduce computational complexity of the program, since all matrix elements are computed separately for each power of $\exp r$ in the method by Cheshire (see also Chapter 3).

[20] There are various views on where exactly to put the third centre. Anderson originally suggested to place it on the the centre of charge ($\overline{\mathrm{AC}}/\overline{\mathrm{CB}} = Z_\mathrm{A}/Z_\mathrm{B}$), which seems the most natural choice. Recent considerations, however, favour the equiforce point ($\overline{\mathrm{AC}}/\overline{\mathrm{CB}} = \sqrt{Z_\mathrm{A}/Z_\mathrm{B}}$), because ionisation processes take place in the vicinity of this point [10]. The attempt of Fritsch and Lin (see below) can be seen as a limiting case, where the third centre coincides with one other centre ($\overline{\mathrm{AC}} = 0$).

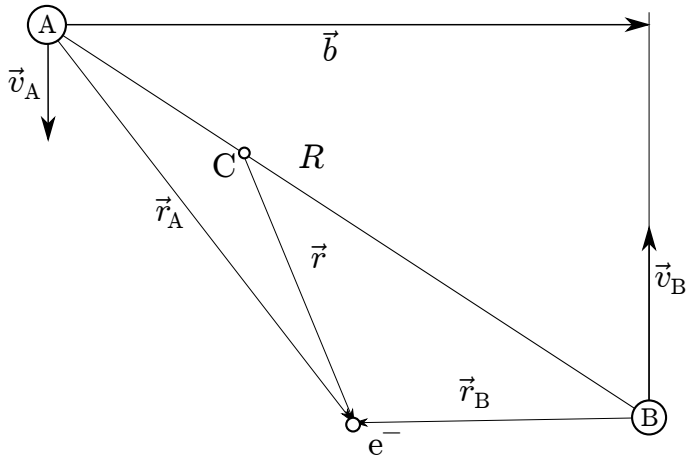


Figure 2.8: Trajectories in the three-centre expansion as proposed by Anderson *et al.* [30]. The origin is placed in the third centre (C), whereas both other centres (A, B) are moving.

2.5.1 Two-centre united atom expansion

Finally, Fritsch and Lin [31] turned these ideas into a powerful modelling scheme [10], basically by combining the attempts by Cheshire and Anderson. They too proposed the use of united atoms states, but instead of placing them on a third centre, they diagonalised these states onto the atomic orbitals centred on the nuclei.

This effort can be justified from both viewpoints discussed earlier: in the united-atom limit, i. e., at small internuclear separations or in the asymptotic region, it does not matter where to put the third centre suggested by Anderson *et al.*, so we may as well choose it to be identical with a nucleus. Secondly, one can argue that these states model the continuum much in the same way as the pseudo-states constructed by Cheshire *et al.* So the main advantage is that this method avoids the computational complexity of a third centre while retaining the physics of the united atoms.

Fritsch and Lin provided another argument for the validity of the approach: they diagonalized the full two-centre Hamiltonian (2.13) in an atomic base including the pseudo-states for various internuclear separations R

$$H_{\mathrm{CC}}(R)|\varepsilon_i(R)\rangle = \varepsilon_i(R)|\varepsilon_i(R)\rangle, \tag{2.47}$$

which yields the electronic energy $\varepsilon_i(R)$ over the internuclear distance[21]. Comparing the eigenenergies of the MO with various expansions into AO, they observed that the

[21] Note that for molecular orbital expansions, this is not necessary because the states are by construction eigenstates of the close-coupling Hamiltonian. Therefore, $|\varepsilon_i(R)\rangle$ already corresponds to the MO states, unlike atomic orbitals and united atoms, which are just eigenstates of the *atomic* and the united-atom Hamiltonian, respectively.

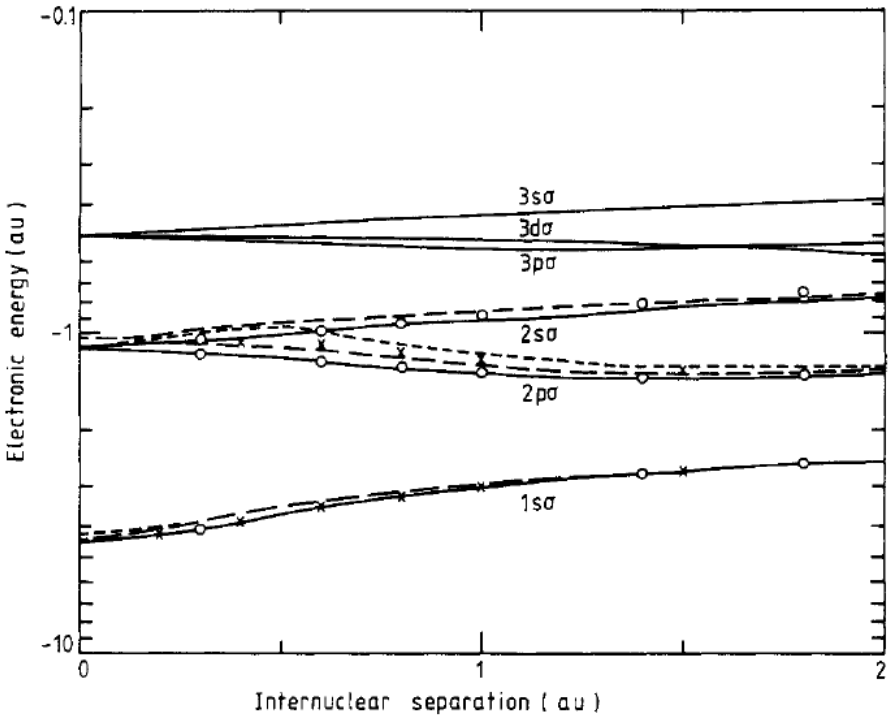


Figure 2.9: Eigenenergies $\varepsilon(R)$ of the close-coupling Hamiltonian over the internuclear distance R for the system $(\mathrm{H}-\mathrm{He})^{2+}$ (taken from [31]). Wee see that the exact MO curves for the lowest σ states (——) are poorly modelled by a plain AO expansions (dashed lines). The inclusion of UA pseudo-states (×,○) significantly improves the approximation.

inclusion of specific UA states significantly improves the agreement with the MO eigenenergies at small internuclear separations (see Figure 2.9). This should not come as a surprise, as united atoms are the limiting case of a molecule when the distance of the nuclei approaches zero ($R \to 0$).

This criterion can be turned around to provide a heuristic for the choice of united atom basis states: we can craft a specific basis to model the eigenenergy of those molecular orbitals we suspect of playing a role in the collision process at hand.

The use of two-centre pseudo-states expansions to model ionisation channels produces two conceptually different types:

- *Direct ionisation* channels are unbound states at the projectile.
- *Capture into continuum* channels are unbound states at the target.

2.5.2 One-centre diagonalisation

There are two facts that complicate the calculation of two-centre expansions with pseudo-states compared to the atomic-orbital set (see Section 2.4):

1. The overlap matrix S has a non-trivial one-centre part ($S^{\mathrm{AA}} \neq \mathbf{1}$) in the case of a non-orthogonal states, e. g., in a combined base of AO and UA states (cf. Equations 2.23, 2.37 and 2.38).

2. Since these states are not eigenstates of the atomic Hamiltonian, they have non-trivial time behaviour (Equation 2.36 does not hold). Moreover, additional terms in the coupling matrix elements (2.45) arise.

To reduce these problems, we diagonalise the *atomic* Hamiltonian H_{A} separately on each centre A (not the full close-coupling Hamiltonian like in the previous section!) [31]:

$$H_{\mathrm{A}}|\psi\rangle_{\mathrm{A}} = \varepsilon|\psi\rangle_{\mathrm{A}} \quad \text{with} \quad |\psi\rangle_{\mathrm{A}} = \sum_{i=1}^{n_{\mathrm{A}}} a_{\mathrm{A}}^{i}|\chi_i\rangle_{\mathrm{A}} \in \mathcal{H}_{\mathrm{A}}, \tag{2.48}$$

where $|\psi\rangle_{\mathrm{A}}$ denotes a wave function in the Hilbert space $\mathcal{H}_{\mathrm{A}}$ on the centre. Note that we use the simple base states $|\chi_i\rangle_{\mathrm{A}}$ without the electron translation factor. This is because in all two-centre expansions, a simple plane-wave ETF of the form (2.43) can be used, which commutes with the eigenbasis transformation.

When multiplied with a base state from the left, equation (2.48) relates to a *generalised eigenvalue problem* for the amplitude vector[22]:

$$(H^{\mathrm{A}} - \varepsilon S^{\mathrm{A}})\vec{a}_{\mathrm{A}} = \vec{0} \tag{2.49}$$

where we have introduced the one-centre Hamilton matrix $H_{ij}^{\mathrm{A}} := {}_{\mathrm{A}}\langle\chi_i|H_{\mathrm{A}}|\chi_j\rangle_{\mathrm{B}}$ and the one-centre overlap matrix $S_{ij}^{\mathrm{A}} \equiv S_{ij}^{\mathrm{AA}} := {}_{\mathrm{A}}\langle\chi_i|\chi_j\rangle_{\mathrm{B}}$. It can be shown [33] that the solution is of the form

$$H^{\mathrm{A}} = S^{\mathrm{A}} B^{\mathrm{A}} E^{\mathrm{A}} (B^{\mathrm{A}})^{-1} \tag{2.50}$$

where B^{A} is the basis[23] of the eigenvectors (which we now denote by $\vec{v}_i^{\mathrm{A}}$ to avoid confusion with the general amplitudes) and E^{A} is the diagonal form of the Hamilton matrix

[22] Equation (2.49) is also called a *positive definite eigenvalue problem*. This arises from the fact that S^{A} is positive definite because it comes from an inner product with vectors that are non-trivial ($\neq |0\rangle$). Any such problem could in principle be transformed to a regular eigenvalue problem by multiplying with $(S^{\mathrm{A}})^{-1}$ from the left (which is guaranteed to exist since S^{A} is Hermitian and all its eigenvalues are positive). On closer examination, however, it turns out that it is faster and numerically more stable to solve a generalized eigenvalue problem directly [32]. As a result, all major linear algebra packages include solver routines for such problems.

[23] Similar to the eigenbasis of a "normal" Hermitian eigenvalue problem, the eigenbasis B^{A} is orthogonal, but with respect to the inner product induced by S^{A}. This relates to the generalised Unitarity relation (compare also with (2.26)):

$$(B^{\mathrm{A}})^{\dagger} S^{\mathrm{A}} B^{\mathrm{A}} = \mathbf{1}^{\mathrm{A}}$$

containing the eigenenergies $\varepsilon_i^{\mathrm{A}}$:

$$B^{\mathrm{A}} := (\vec{v}_1^{\mathrm{A}}\, \vec{v}_2^{\mathrm{A}}\, \cdots\, \vec{v}_n^{\mathrm{A}}) \quad \text{and} \quad E^{\mathrm{A}} := \mathrm{diag}(\varepsilon_1^{\mathrm{A}}, \varepsilon_2^{\mathrm{A}}, \ldots, \varepsilon_n^{\mathrm{A}}) \tag{2.51}$$

Finally, we combine the eigenbasis transformations of both centres to yield transformations for the full amplitude vector $\vec{a}$:

$$B := \begin{pmatrix} B^{\mathrm{T}} & 0 \\ 0 & B^{\mathrm{P}} \end{pmatrix} \quad \text{and} \quad E := \begin{pmatrix} E^{\mathrm{T}} & 0 \\ 0 & E^{\mathrm{P}} \end{pmatrix}$$

In a combined expansion of atomic orbitals and pseudo-states, the atomic orbitals are retained in the diagonalisation process[24], since they are already eigenstates of H^{A}. The eigenenergies of the pseudo-states, on the other hand, depend on the basis set and even on the atomic orbital states included. In particular, if a full bound spectrum of atomic orbitals is used, all united atom have positive energies and therefore correspond to ionisation channels. If only a few atomic orbitals are used, united atom states might be slightly bound.

Therefore one must align the choice of pseudo-states with the atomic basis used when crafting a basis set for the problem at hand.

2.5.3 Generalised coupled-channel equations

To include the eigenbasis decomposition (2.50) into the coupled-channel equations (2.45), one can go two ways, which are equivalent but differ in the computational approach:

1. Apply the transformation to the basis states $|\chi_i\rangle$, or
2. Apply the transformation to the amplitude vector $\vec{a}$.

In the first method, we use the eigenbasis B (2.51) to construct an orthonormal basis $|\chi_i'\rangle$ of the atomic Hamiltonians as a linear combination of our previous basis:

$$|\chi_i'\rangle = B_{ij}|\chi_i\rangle = v_{i,j}|\chi_j\rangle$$

This immediately restores the simple form (2.37) of the overlap matrix S. Moreover, since these states trivially fulfil (2.50), they have a simple time dependence analogous to (2.36):

$$H_{\mathrm{A}}|\chi_i'\rangle_{\mathrm{A}} = \varepsilon_i|\chi_i'\rangle_{\mathrm{A}} \quad \Longrightarrow \quad |\chi_i'(t)\rangle = \exp(-\imath\varepsilon_i t)|\chi_i'\rangle$$

Therefore, the non-trivial parts of S and M can be computed in the same way as in atomic orbitals expansions (2.38).

[24] They may however be – depending on the algorithm – re-ordered by the eigenbasis transformation.

However, from a computational perspective, this might not be ideal: if symbolic structures are used to store the wave functions (see Chapter 3), a linear combination multiplies the number of terms, which adds to the complexity of the matrix elements. To address this, we use the second method, which is conceptually more difficult but none-the-less faster.

Here, we retain the original basis states $|\chi_i\rangle$, but instead transform the amplitude vector $\vec{a}$ into the orthonormal base $|\chi'_i\rangle$. This corresponds to rewriting of (2.22)[25]

$$\left(B^{-1}MB - \imath B^{-1}SB\frac{d}{dt}\right)B^{-1}\vec{a} = 0,$$

which leaves us with transformation rules for the amplitude vector as well as for the matrices S and M.

Since the pseudo-states are not eigenstates of the atomic Hamiltonian, the coupling matrix elements (2.38) must be amended by $H - \imath\partial_t$. The evaluation of the time derivative in the original base proves to be problematic, therefore we extract this part from the coupling matrix element, which leaves us with

$$S_{ij}^{\mathrm{AB}} := {}_{\mathrm{A}}\langle\chi_i|F(\vec{v}_{\mathrm{B}} - \vec{v}_{\mathrm{A}})|\chi_j\rangle_{\mathrm{B}} \tag{2.52a}$$

$$M_{ij}^{\mathrm{AA}} := {}_{\mathrm{A}}\langle\chi_i|(V_{\mathrm{B}} + H_{\mathrm{A}})|\chi_j\rangle_{\mathrm{A}} \tag{2.52b}$$

$$M_{ij}^{\mathrm{AB}} := {}_{\mathrm{A}}\langle\chi_i|F(\vec{v})(V_{\mathrm{A}} + H_{\mathrm{B}})|\chi_j\rangle_{\mathrm{B}}. \tag{2.52c}$$

with $\vec{v} := \vec{v}_{\mathrm{B}} - \vec{v}_{\mathrm{A}}$. In principle, the one-centre overlap matrices S^{AA} are also non-trivial due to the non-orthogonal basis. However, we do not need these parts because they become the unit matrix after the transformation.

Note that we excluded the time-dependent part in M, since it is convenient to evaluate all time dependencies and time derivatives in the orthonormal base, where time evolution is given by

$$U(t) = \exp(\imath Et) = \mathrm{diag}(\exp(\imath\varepsilon_i t)).$$

After transforming the amplitude vector $a' := B^{-1}a$ as well as the matrices $S' := U^\dagger B^{-1}SBU$ and $M' := U^\dagger B^{-1}MBU$, we finally arrive at

$$(M' - S'E)\cdot\vec{a}' = \imath S'\cdot\frac{d}{dt}\vec{a}'. \tag{2.53}$$

which are the coupled channel equations (2.22) in the orthonormal frame. The additional term $S'E\vec{a}'$ corresponds to the time derivative that we excluded from the coupling matrix.

[25] Note that because the basis transformation is stationary, B and d/dt commute.

2.6 Other techniques

Aside from the close-coupling method presented in this chapter, a wide array of other techniques for studying atomic collisions have been developed. In this section, we will briefly present some of these methods, for a comprehensive compilation, please refer to the book by Brandsen [10].

Molecular orbitals At slow collisions, expansions into atomic orbitals no longer model the physical channels. Apart from pseudo-state expansions (see Section 2.5), one can also visualise slow scattering process as transiently formed molecules [11]. The corresponding basis functions are called *molecular orbitals* (MO) and are eigenstates of the Hamiltonian for fixed inter-nuclear separation R.

Numerous studies have been conducted with MO expansions, mostly within the Perturbed Stationary State (PSS) model. However, dynamical models based on MO expansions suffer from the fact that they do not obey proper boundary conditions at infinite separations, or, equivalently, are not Galilein invariant [11]. Therefore, cross sections depend on the choice of the coordinate origin.

Gaussian orbitals In an attempt to reduce the complexity that comes with multi-centre expansions, so-called *Gaussian type orbitals* (GTOs) prove very useful: here, the exact slater-type orbitals (see Section 3.1) are approximated by basis functions of the form $r^q \exp(-\alpha r^2)$. The method now revolves around the theorem that a two-centre exhange matrix element of GTOs can be written as a finite sum of one-centre exchange matrix elements, in turn for which an analytic expression can be found.

GTO expansions are very common in quantum chemistry and are currently being extended to ion–molecular collisions [34]. It scales very well to multi-electron processes and multiple centres because they can be easily reduced to simple one-centre integrals. The crucial part of the method remains the angular part, which must again be processed by a symbolic differentiation method.

S-matrix approaches While our approach is integrating (2.22) explicitly for some initial condition, there are other approaches which try to compute the scattering matrix explicitly. Unlike trying to do a quadrature for the amplitude vector $\vec{a}$, these methods try to find an explicit expression for the time evolution operator U by "integrating" the Hamiltonian H. This quadrature is formally given by the time-ordered exponential $\mathcal{T} \exp\left(-\imath \int^t dt'\, H(t')\right)$ (also called the *Dyson series*), which unfortunately does not have a closed-form solution in general. Nevertheless, an approximate solution is given by the

Magnus series [35, 36]

$$U(t) = \exp\left(-\imath \int^t dt' \left(H(t') + \frac{1}{2}\int^t dt'' \left([H(t'), H(t'')] + \frac{1}{3}\int^t dt''' \cdots\right)\right)\right)$$

One usually first switches to the interaction picture by factorising out the diagonal part of H and then computes U in small time steps [37]. The scattering matrix is then simply given by the time evolution matrices joined by matrix multiplication.

The S-matrix approach is particularly powerful in the case of small interactions, where the expansions reduce to one term, effectively modelling the *Born approximation* [22]. It has been used especially in conjunction with the distorted wave approximation [18] and has the advantage that the complete scattering matrix is computed. However, in the case of heavy interactions, the method suffers from poor convergence properties of the expansions, making a lot of exponentiations necessary.

Classical Trajectory Monte Carlo While most methods for the calculation of atomic collisions are at most semi-classical, the *Classical Trajectory Monte Carlo method* (CTMC) originally introduced by Abrines and Percival [38] does not include any quantum mechanical dynamics. Instead, the full Hamiltonian (2.1) is interpreted as a classical Hamilton function $H(p, q)$. Quantum mechanics enters through the distribution of the initial states, which is the crucial point in the method.

Following the definition of Sawilowsky [39], CTMC is a *Monte Carlo simulation*: first, a random sample of initial conditions is generated following some distribution. Second, Hamilton's equations of motion are solved in order to calculate classical trajectories for the sample. The cross sections are then given by the fraction of the particles in the respective final configurations.

CTMC methods have been repeatedly used to predict total cross sections with good accuracy, provided that a suitable initial distributions are chosen (see Chapter 4). In particular, no channel can be "forgotten". One problem, common to all Monte Carlo simulations, is that weakly populated channels are seldom hit by particles. As a result, a greater number of trajectories must be calculated to get sufficient *statistics* for the estimation of the cross sections [40].

3 Computation

Now that the method has been established, we can in theory compute amplitudes and resulting cross sections for any basis expansion by solving the matrix differential equation we introduces in Section 2.1

$$M \cdot \vec{a} = \imath S \cdot \frac{d}{dt}\vec{a}. \qquad (\&2.22)$$

Conceptually, this task can be split into two parts:

1. The computation of the matrix elements of the coupling matrix M_{ij} (2.24) and the overlap matrix S_{ij} (2.23), and

2. Solving the differential equation for the amplitude vector a by numerical integration ("quadrature") of (2.22).

It turns out that due to the fact that the states are located at two different centres, the computation of the matrix elements is very difficult and exceeds the computational complexity of the integration by several orders of magnitude. A few techniques have been developed for performing this task, and we will follow the approach of Shakeshaft (see Section 3.1), which has later been amended by the work of Kocbach and Liska (see Section 3.3).

This approach is special because it builds a symbolic form of the exchange integral: a sum of one-dimensional finite integrals. In order to compute these sums effectively, new data structures were introduced into the programme (see Section 3.2).

It turns out that even with those optimisations, the computational demands by far exceed the capabilities of PCs, which makes clever parallelisation schemes necessary for the integration of the coupled channels equation.

3.1 Shakeshaft exchange integrals

3.1.1 Obtaining exchange integrals

We again start with the time-dependent Schrödinger equation for a superposition of basis states. Typical base states are hydrogen-like bound states on either target of the form:

$$\langle \vec{r} | Znlm \rangle \quad = \quad N_{nl} \rho^l e^{\rho/2} L_{n-l-1}^{(2l+1)}(\rho) Y_l^m(\Omega)$$

where N_{nl} is a normalisation constant, L_k^q is the q-associated Laguerre polynomial of degree k, Y_l^m are spherical harmonics, and $\rho = 2Zr/n$ is the reduced radius in atomic units.

For generality, we expand the radial wave functions into Slater type orbitals $|\alpha n\rangle$, which are radial-symmetric wave functions of the form

$$\langle r|\alpha n\rangle \equiv S_n^{(\alpha)}(r) := Nr^{n-1}\exp(-\alpha r)$$

where $\alpha := Z/n$ is a positive quantum number related to charge and N again is a normalisation constant. Observing that spherical harmonics in Cartesian coordinates obey (see Appendix A.1)

$$Y_l^m(\vec{r}) \quad = \quad r^{-l}\sum_i C_i x^{\xi_i} y^{\eta_i} z^{\zeta_i} \quad \text{with} \quad \xi_i + \eta_i + \zeta_i = l,$$

we can write a complete "base" of *Cartesian exchange orbitals* $|\alpha n\vec{l}\rangle$, which should prove especially useful in calculating exchange integrals[1]:

$$\langle \vec{r}|\alpha n\vec{l}\rangle := Nr^{n-2}\exp(-\alpha r)x^{l_1}y^{l_2}z^{l_3} \tag{3.1}$$

where n takes the role of the main quantum number and $l_i \in \mathbb{N}_0$ correspond to the magnetic quantum numbers. The usage of $n-2$ is purely for conventional convenience, due to the fact that for $n \geq 1$, these states remain normalisable (see also Appendix A.2).

To shorten our notation, we introduce vector powers by defining [41]

$$(\vec{x})^{\vec{y}} := \prod_{i=1}^{3}(x_i)^{y_i} \quad \text{and} \quad (\nabla_{\vec{x}})^{\vec{n}} := \prod_{i=1}^{3}\left(\frac{\partial}{\partial x_i}\right)^{n_i}. \tag{3.2}$$

Now, by expanding the wave function into exchange orbitals, we can reduce the exchange integrals to terms of the form

$${}_A\langle\psi_1|\psi_2\rangle_B = \sum A_{n_1 l_1 n_2 l_2}\,{}_A\langle\alpha_1 n_1\vec{l_1}|\alpha_2 n_2\vec{l_2}\rangle_B$$

[1] Note, however, that these orbitals do not form an orthogonal system: $\langle\alpha' n'\vec{l'}|\alpha n\vec{l}\rangle = N'^* N/|N''|^2 \neq 0$, where N'' is the normalisation constant of the state $|\frac{1}{2}(\alpha+\alpha'), \frac{1}{2}(n+n'), \frac{1}{2}(\vec{l}+\vec{l'})\rangle$. In fact, they are not even linearly independent, since

$$|\alpha(n+2)l_1l_2l_3\rangle = |\alpha n(l_1+2)l_2l_3\rangle + |\alpha n l_1(l_2+2)l_3\rangle + |\alpha n l_1 l_2(l_3+2)\rangle$$

due to the fact that we dropped the constraint $l_1 + l_2 + l_3 = l$.

where we can express the overlap in configuration space by

$$ {}_A\langle \alpha_1 n_1 \vec{l_1} | \alpha_2 n_2 \vec{l_2} \rangle_B = \int d^3 r_A r_A^{n_1-2} r_B^{n_1-2} (\vec{r}_A)^{\vec{l}_1} (\vec{r}_B)^{\vec{l}_2} \exp(-\alpha_1 r_A - \alpha_2 r_B) $$

Similarly, for the exchange matrix elements, we find that we can expand the potential $V(R, r_A, r_B)$ in powers of r_A and r_B, yielding the exchange integral

$$ {}_A\langle \alpha_1 n_1 \vec{l_1} | V | \alpha_2 n_2 \vec{l_2} \rangle_B = \int d^3 r_A r_A^{n_1'-2} r_B^{n_1'-2} (\vec{r}_A)^{\vec{l}_1'} (\vec{r}_B)^{\vec{l}_2'} \exp(\imath \vec{a} \vec{r}_A + \imath \vec{b} \vec{r}_B - \alpha_1 r_A - \alpha_2 r_B) $$

where $\vec{a}$ and $\vec{b}$ are real vectors arising from the potential and $n', \vec{l'}$ are modifications to n, l due to the addition of the potential. This is the most general integral we will have to solve, also called $I(n_1, \vec{l}_1, n_2, \vec{l}_2)$.

3.1.2 Transformation to one-dimensional integrals

Several techniques have been proposed to solve this integral. We will follow the method of Shakeshaft [42], where the three-dimensional integrals are transformed to a sum over one-dimensional integrals

$$ \begin{aligned} I &= \int d^3 r_A \, r_A^{n_1-2} r_B^{n_2-2} (\vec{r}_A)^{\vec{l}_1} (\vec{r}_B)^{\vec{l}_2} \exp(\imath \vec{a} \vec{r}_A + \imath \vec{b} \vec{r}_B - c r_A - d r_B) \\ &= 2\pi (-\imath)^{l_1+l_2} (\nabla_{\vec{a}})^{\vec{l}_1} (\nabla_{\vec{b}})^{\vec{l}_2} \left(-\frac{\partial}{\partial c} \right)^{n_1-1} \left(-\frac{\partial}{\partial d} \right)^{n_2-1} \int_0^1 dy \frac{\exp(\imath \vec{B} \vec{R} - AR)}{A} \end{aligned} \quad (3.3) $$

where $\vec{R} = \vec{r}_A - \vec{r}_B$ is the internuclear separation and where we introduced the quantities $A^2 := y(1-y)|\vec{a} + \vec{b}|^2 + yc^2 + (1-y)d^2$ and $\vec{B} := y\vec{a} - (1-y)\vec{b}$. We state this formula without its (rather labour-intensive) proof.

For our collision system, we identify $\vec{a} = v\vec{e}_z$ and $\vec{b} = 0$. To perform these differentiations analytically, we write down the form of the most general term that will be arising in the computation as a polynomial multiplied with the exponential factor.

Next, we derive general formulae for the terms that will be arising in the computation

$$ C^{\nu}_{\alpha\rho\zeta} := y^{\nu} A^{\alpha} R^{\rho} c^{\zeta} f; \quad D^{\mu}_{\alpha\rho\delta} := (1-y)^{\mu} A^{\alpha} R^{\rho} d^{\delta} f; \quad A^{(i)\nu\mu}_{\alpha\rho\phi\psi} := y^{\nu} (1-y)^{\mu} A^{\alpha} R^{\rho} V_i^{\phi} R_i^{\psi} f, $$

where $f := \exp(i\vec{B}\vec{R} - AR)$ and $V_i := a_i$ by using the Shakeshaft rules:

$$-\frac{\partial}{\partial c}C^{\nu}_{\alpha\rho\zeta} = -\alpha C^{(\nu+1)}_{(\alpha-2)\rho(\zeta+1)} + C^{(\nu+1)}_{(\alpha-1)(\rho+1)(\zeta+1)} - \zeta C^{\nu}_{\alpha\rho(\zeta-1)} \tag{3.4a}$$

$$-\frac{\partial}{\partial d}D^{\mu}_{\alpha\rho\delta} = -\alpha D^{(\mu+1)}_{(\alpha-2)\rho(\delta+1)} + D^{(\mu+1)}_{(\alpha-1)(\rho+1)(\delta+1)} - \delta D^{\mu}_{\alpha\rho(\delta-1)} \tag{3.4b}$$

$$\begin{aligned}\frac{\partial}{\partial a_i}, \frac{\partial}{\partial b_i}A^{(i)\nu\mu}_{\alpha\rho\phi\psi} &= \alpha A^{(i)(\nu+1)(\mu+1)}_{(\alpha-2)\rho(\phi+1)\psi} \pm \imath A^{(i)(\nu+1)\mu}_{\alpha\rho\phi(\psi+1)} - A^{(i)(\nu+1)(\mu+1)}_{(\alpha-1)(\rho+1)(\phi+1)\psi} \\ &\quad + \phi A^{(i)\nu\mu}_{\alpha\rho(\phi-1)\psi}\end{aligned} \tag{3.4c}$$

So, we get a polynomial in 12 variables multiplied by f. Instead of storing the terms in arrays like done in the subroutine `CRERS` of the program `ALAIN` [43], we use a binary tree to store the elements (see Section 3.2).

3.1.3 Simplification of one-centre exchange integrals

One-centre exchange integrals arise from states on one centre being affected by the potential at the other centre. If we expand the wave functions in terms of exchange orbitals, we get:

$${}_A\langle\psi|V_B|\chi\rangle_A = \sum_p \sum_q \psi^*_p \chi_q \cdot {}_A\langle\alpha_p n_p \vec{l_p}|V_B|\alpha_q n_q \vec{l_q}\rangle_A$$

where ψ_p and χ_q are expansion coefficients of the respective wave functions. If we assume our potential to be radial-symmetrical and of the form

$$V(r) = \sum_i a_i r^{\rho_i} \exp(-f_i r)$$

we can match the requirements of our exchange integrals:

$${}_A\langle\psi|V_B|\chi\rangle_A = \sum_{p,q,i} \psi^*_p \chi_q a_i \cdot I(n_p + n_q, \vec{l_p} + \vec{l_q}, \rho_i + 2, \vec{0})$$

as well as $c = \alpha_p + \alpha_q$, $d = f_i$ and $\vec{a} = \vec{b} = 0$. In first approximation, we assume the shielding represented by the exponential term to be small and therefore set d to zero, yielding

$$A = \sqrt{y}c \quad \text{and} \quad \vec{B} = 0$$

With these constraints, we can identify the terms of the exchange integrals $I1$ involved in calculating one-centre exchange matrix elements:

$$I \equiv I1(n, \vec{l}) = \sum_i a_i R^{\rho_i} R_1^{\phi_i} R_3^{\psi_i} \int_0^1 dy \sqrt{y}^{\sigma_i} \exp(-cR\sqrt{y})$$

These terms are special cases of $A^{\nu\mu}_{i\alpha\rho\phi\psi}$, where we used the binomial expansion of $(1-y)^\mu$ to convert everything to powers of y. Substituting $u := cR\sqrt{y}$, we can give a closed form of these integrals:

$$I1(n,\vec{l}) = \sum_i \frac{2a_i\Gamma(\sigma_i+2)}{c^{\sigma_i+2}} R^{\rho_i-(\sigma_i+2)} R_1^{\phi_i} R_3^{\psi_i} P(\sigma_i+2, cR)$$

where we used the lower incomplete Gamma function

$$P(a,x) = \frac{1}{\Gamma(x)} \int_0^x dt\, \mathrm{e}^{-t} t^{a-1}$$

Note that this approach is guaranteed to converge for $\rho_i \geq -1$, but may fail for potentials of higher order, since the integrals P may diverge.

This yields a polynomial in four variables, which is again stored in the polynomial structure. Unlike the original programme `JANAL` by Hansen and Dubois [44], the coefficients are stored in a partially evaluated form.

3.2 Tree containers for symbolic structures

To store symbolic form of wave functions and the exchange integrals obtained in Section 3.1, we need to model polynomials. These polynomials are primarily used to perform symbolic differentiations, calculus and lots of evaluations.

Symbolic differentiation in particular is problematic because with each differentiation step, the number of terms at least triples (Equations 3.4), but at the same time lots of linearly dependent terms arise. As a result, we need to (i) insert terms in a fast fashion and (ii) detect and add up duplicates early.

Evaluation poses another problem: if we evaluate a polynomial with high powers on a term-by-term basis, we massively impeach performance. Moreover, cancellation errors may occur and, as the number of terms increase, may even become dominant. Therefore, we want to collapse the polynomial using the Horner scheme [45]:

$$\sum_{n=a}^{b} c_n x^n = (\cdots(((c_b \cdot x + c_{b-1})x + c_{b-2})x + \cdots)x + c_a x^a \tag{3.5}$$

This requires (iii) the terms to be in a *lexicographic order* [46]: this essentially means we need to order terms by the power of one variable, then by the powers of the next

powers of x	powers of y	powers of z	Coefficient
1	3	4	1.0
8	2	2	0.3
8	2	3	0.3
5	2	1	-1.0
5	-2	1	-1.0

Table 3.1: Tabular representation of the polynomial (3.6). Typically, this is modelled in the original Fortran 77 code by using three integer arrays `xarr`, `yarr` and `zarr` for the variable powers as well as a complex coefficients array `kfarr`.

variable, and so forth[2]. For example[3], $[x^3y^2z^9] < [x^3y^3z^1]$, because $[y^2] < [y^3]$. By using a descending order, we have the advantage of multiplying by x and not by x^{-1} in the Horner scheme. Finally, to find a fast way to use the Horner scheme, we need to find (iv) a fast way to know up to which variable the powers of two adjacent terms agree.

A first, straight-forward approach to store a polynomial in n variables is simply to store an array of *monomials* (terms), where each monomial is identified by an array of n powers and its coefficient. For example, the following polynomial in three variables x, y, z

$$P(x, y, z) = xy^3z^4 + 0.3 \cdot x^8y^2(z^2 + z^3) - x^5(y^2 + y^{-2})z \tag{3.6}$$

is modelled by Table 3.1. This is actually the way the original code stores all symbolic structures.

However, the requirements (i – iv) are not met by this storage. Provided that we insert n terms into our polynomial,

1. detecting duplicate (at insertion) terms requires $O(n^2)$, since ordered insert into an array is highly inefficient and therefore not feasible.

2. ordering the terms is of complexity $O(n \log n)$. Moreover, comparing two terms means comparing all variables, which is ineffective.

3. evaluating an unordered set, as done in the original program, may easily lead to cancellation and is ineffective.

[2]More formally, we define the order as follows: Let $(A, \leq_A)$ and $(B, \leq_B)$ be partially ordered sets. A partial order $\leq_L$ on the Cartesian product set $A \times B$, where for all $(a_1, b_1), (a_2, b_2) \in A \times B$

$$(a_1, b_1) \leq_L (a_2, b_2) \quad \leftrightarrow \quad \begin{cases} a_1 \leq_A a_2 & a_1 \neq a_2 \\ b_1 \leq_B b_2 & a_1 = a_2 \end{cases}$$

is called *lexicographic order*. By iterating this definition $n - 1$ times, we can easily define a lexicographic order for A^n. If we assume $A = \mathbb{Z}$, this yields a total order for monomials in n variables, provided that we choose an order of variables.

[3]The use of square brackets should clarify that this is an order on the *term powers*, not on the possible *values* of these expressions (e. g., for $y \leq 1$, $y^2 < y^3$ does not hold).

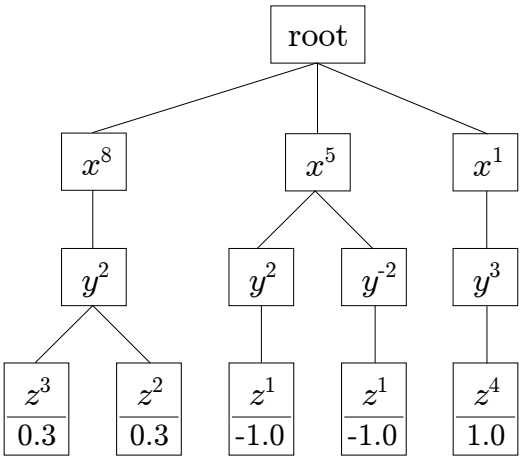


Figure 3.1: Representation of the polynomial (3.6) in monomials stored in an n-ary tree structure, where the ranks distinguish the variables and siblings represent different powers.

3.2.1 Binary tree representation

In a first step towards solving this problem, we observe that a factorised form of a polynomial can be represented by an n-ary tree, provided that we choose an order of variables (see Figure 3.1). Each node of the tree represents a variable, a power thereof. If it is a leaf node, it also contains the coefficient of the term.

If we keep siblings sorted by power, we get a tree suitable for sorted insert and evaluation of the variables. However, to get a well-performing tree for sorted insert, we still need to take care which container to use to store the children: An array storage is inefficient when the number of elements grows too large, since many the elements must be moved when inserting.

To overcome these problems, we use the so-called *natural correspondence* to map n-ary trees to binary trees [47]: for each node in the n-ary tree, an order is chosen for its children. The left branch of the binary result tree is pointed to the first child of the corresponding node in the n-ary tree, whereas the right branch points to the next sibling of this node. Because of the different semantic meanings of the branches, this data structure is also called a *left-child right-sibling (LC-RS) tree*.

Figure 3.2 shows the binary LC-RS representation of the n-ary tree in Figure 3.1. The left branch (represented by horizontal lines) separates the levels of the tree, distinguishing different variables. The right branch (represented by vertical arrows), on the other hand, points to the next sibling of the node, which is associated with the subsequent power of the same variable.

One can easily show that the natural correspondence is a one-to-one mapping. Indeed, you can restore the polynomial (3.6) from the tree in Figure 3.2 as follows: Start from at the root node (top left) and replace each node by the respective variables and powers

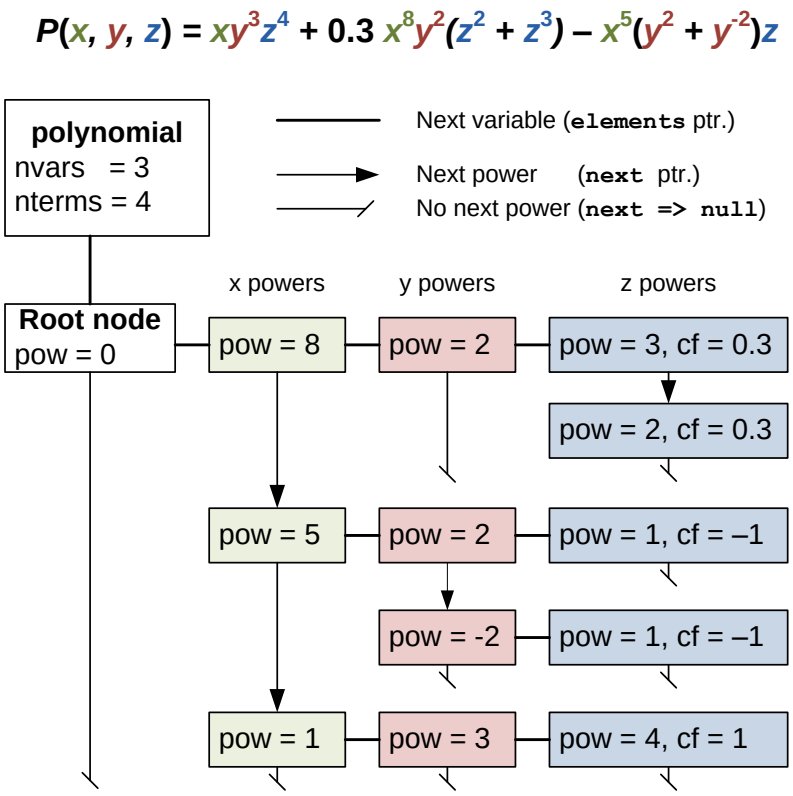


Figure 3.2: Internal representation of an instance of type `polynomial` modelling formula (3.6). The nodes, implemented in type `polyterm`, form a *LC-RS tree* structure where the right branch (`elements` pointer, lines) distinguishes different variables and the down branch (`next` pointer, arrows) points to the next power in the same variable, enforcing a descending order of powers.

$cf \cdot var^{pow}$. First follow the horizontal lines, replacing it with an opening bracket. Then follow the vertical lines, replacing each arrow with a '+' and each dead end ↾ with a closing bracket.

In this representation, many common operations and calculus reduce to simple tree mutation, being considerably faster than operating on (even sorted) arrays. Moreover, using binary LC-RS trees instead of arrays considerably speeds up sorted insert, duplicate detection and evaluation at the expense of finding a specific term (which we do not need to do).

A extensive library (`polynomial.f90`) of evaluation, manipulation and calculus methods was written in Fortran 90 (see Appendix B.1 for a comprehensive API documentation). One of the downsides of this implementation is that Fortran 90 does not provide any destruction mechanism, which leaves the task of managing the object's scope to the user[4].

3.2.2 Performance improvements of structure creation

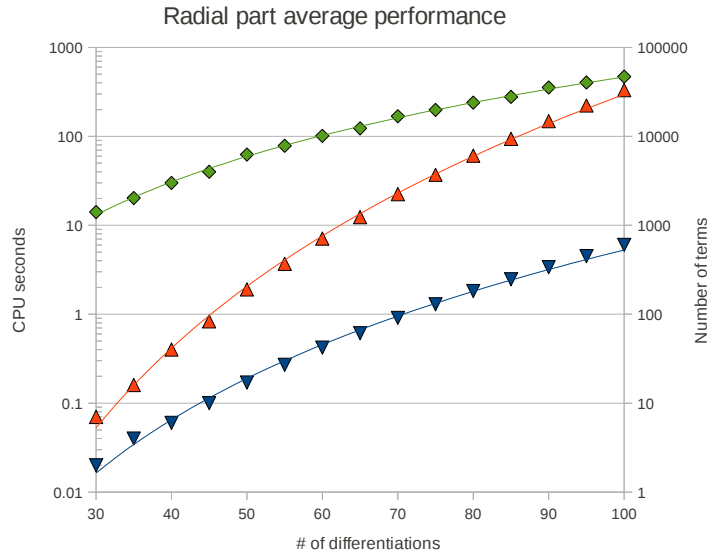
To put the routines to the acid test, we used them as a basic building block for a programme computing the exchange matrices' symbolical structures introduced in Section 3.2 [50].

We closely followed the basic principles of the code by Hansen [43], but the actual symbolic manipulations were delegated to the polynomial library. As with the previous approach – encapsulated in the subroutine `CRERS`[5] – we did not calculate the most general form of the elements, but instead partially evaluated the polynomial for values of c and d. Algorithmically, we improved the detection of terms that vanish because of the symmetries of the collision system in the impact parameter model.

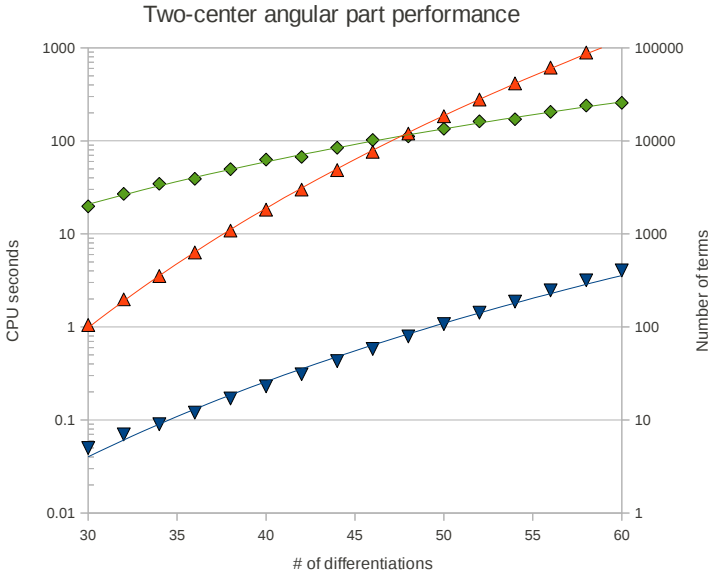
This allowed us to compare the runtime performance of the symbolic differentiations in both codes, illustrated by Figure 3.3: we observe that for all cases, the performance of the new tree method is a lower boundary for the running time of the old `CRERS` routine. Studying the asymptotic behaviour of the radial part, we find that the complexity is reduced from $O(n^{7.17})$ to $O(n^{4.80})$ for the exchange integral $I(\frac{n}{2}, \vec{0}, \frac{n}{2}, \vec{0})$ with $c \neq d$. This improves running time by $O(n^{2.37})$, speeding up large integrals.

For the two-centre angular part, we investigated the exchange integral $I(1, \frac{n}{2}, 1, \frac{n}{2})$. The complexity of the algorithm was improved from $O(n^{10.3})$ to $O(n^{6.48})$, a massive performance gain by $O(n^{3.82})$. All calculations were performed on an Intel i5 750 processor with 2 GB of RAM running Ubuntu 10.04 x64 edition. Both codes were compiled using gfortran 4.4.1 with maximum optimisation levels.

[4] The most recent advancements in the Fortran standard, Fortran 2003 and Fortran 2008, provide such life-cycle facilities by use of type-bound procedures and the `final` keyword [48, 49]. However, these constructs are still poorly supported by mainstream compilers (GNU Fortran compiler from version



(a)



(b)

Figure 3.3: Performance of our tree method (∇) versus the CRERS method ($\triangle$) for the creation of exchange integrals on an Intel i5 750 processor over the number n of differentiations involved: (a) radial exchange integral $I(\frac{n}{2}, 0, \frac{n}{2}, 0)$, (b) two-centre angular exchange integral $I(1, \frac{l}{2}, 1, \frac{l}{2})$ for $c \neq d$. A power regression was laid through the measurement points. The number of terms ($\diamond$, secondary y-axis) is identical for both methods.

Asymptotic behaviour	Radial part		Angular part	
	$O(n^x)$	R^2	$O(n^x)$	R^2
Number of terms	2.97	99.9%	3.66	99.8%
CRERS	7.17	99.8%	10.30	100.0%
Polynomial library	4.80	99.7%	6.48	99.4%

Table 3.2: Asymptotic behaviour of the exchange integral creation in powers of n for angular and radial part as obtained by power regression of data in Figure 3.3 (where R^2 is the fit quality determined by Pearson's χ^2-test [51]).

3.2.3 Sequential representation

We successfully sped up the creation of the exchange matrix elements, which proved a performance bottleneck in the previous code. However, the use of the binary trees exhausts another critical resource more quickly: the main memory (RAM).

Since Fortran 95 does not provide a performant way of type casting[6], we need to store a coefficient (which is unused for branch nodes) for any node in the tree. This means that every tree node occupies at least $2p + 2m + 2$ bytes, where p is the floating-point precision and m is the memory width. For instance, each node demands 34 bytes in a 64-bit system with double precision floating-point arithmetic.

The huge memory demand obstructs our path to the calculation of more complex collision systems: the number of basis states required to model a highly charged ion A^{q+} rises approximately with $q^{9/4}$ [52]. The number of matrix elements rise with $q^{9/2}$ as a result, which is already of the order of 10^6 for argon – hydrogen collisions. The bad scaling property of the memory demand with respect to the ion charge q is a fundamental flaw of the method of storage of symbolic structures. Therefore, to allow calculations in this framework, we need to find a more compact representations of the underlying polynomials after their creation.

The lion's share of the administrative overhead is due to pointers modelling the topology of the tree. To address this problem, we switch to an *implicit representation* of the tree [47]. These representations store the nodes in a sequential structure, dropping all pointers but using a special order. The topology of the tree is – to some degree – now implied by the position of the nodes in the sequence.

Since we do not need to alter the polynomials after their creation, we use an array to store the nodes. With the evaluation in mind, we use *pre-order linearisation* for the order in the array. This means, we first store the node itself, and then their sub-trees in some fixed order. To make this fairly abstract algorithm more concrete, Table 3.3

4.5, Intel Fortran compiler from version 11.1) .

[5]The meaning of the subroutine's name is unknown.

[6]Fortran 95 does include the `transfer` intrinsic, which allows some form of type casting, but it creates a copy of the data in the process and does not provide any type-safety.

provides the pre-order sequential representation of the tree in Figure 3.1.

Sequence number	1	2	3	4	5	6	7	8	9	10	11	12	13
Number of children	3	1	2	0	0	2	1	0	1	0	1	1	0
Power	–	8	2	3	3	5	2	1	-2	1	1	3	4
Coefficient	–	–	–	0.3	0.3	–	–	-1.0	–	-1.0	–	–	1.0

Table 3.3: Pre-order sequential representation (with child count) of the tree in Figure 3.1.

This structure is already far more compact than the tree structure. Still, there are redundancies:

- The root node does not have a power entry.
- Since the tree has a fixed depth, all zero children entries are implied by the representation.
- Similarly, no branch node has a coefficient.

Removing all these unnecessary entries, we reduce the memory for a branch node to 3 bytes and the memory of a leaf node to 18 bytes, yet being fully compatible to the previous structure. The resulting memory layout for the tree in Figure 3.1 is depicted in Table 3.4.

There is still room for improvement: for polynomials with a large number of variables, there will be some "standalone" terms (like xy^3z^4 in our example polynomial), which results in a series of ones. This can be improved by using "0" for the number of terms in such cases, which is shorthand for a series of nodes with one child each. The corresponding layout for our example now shortens to the one in Table 3.5.

Since this adds to complexity of the evaluation routine, this method was not included in the programme.

An extensive library (`lpolynomial.f90`) for evaluation and debugging of these structures was written in Fortran 90 (see Appendix B.2 for a comprehensive API documentation). Array structures are advantageous because Fortran is very good at handling large arrays as opposed to handling large pointer structures. Moreover, the risk of memory fragmentation due to frequent allocation and deallocation of small chunks of memory

Array index	1	2	3	4	5	6	7	8	9	10	11	12
Number of children	3	1	2	2	1	1	1	1				
Power	8	2	3	3	5	2	1	-2	1	1	3	4
Coefficient	0.3	0.3	-1.0	-1.0	1.0							

Table 3.4: Improved pre-order sequential representation (with child count) of the tree in Figure 3.1.

Array index	1	2	3	4	5	6	7	8	9	10	11	12
Number of children	3	1	2	2	1	1	0					
Power	8	2	3	3	5	2	1	-2	1	1	3	4
Coefficient	0.3	0.3	-1.0	-1.0	1.0							

Table 3.5: Improved pre-order sequential representation (with child count) of the tree in Figure 3.1, where standalone entries (consecutive ones in the number of children) are collapsed.

decreases. Finally, the scope of linear polynomials can be managed by Fortran, which minimises the risk for memory leaks.

3.3 Kocbach–Liska symbolic form

In the previous section, we saw that the implementaion of the method of symbolic differentiation must deal with some non-trivial numerical and technical problems. Fortunately, an explicit expression for the differentiations was found by Kocbach and Liska in 1994 [41].[7] Extending the multi-index notation (3.2) to

$$\lfloor (v_i) \rfloor = (\lfloor v_i \rfloor); \quad \binom{\vec{n}}{\vec{k}} = \binom{n_1}{k_1}\binom{n_2}{k_2}\binom{n_3}{k_3}, \quad \sum_{\vec{k}=\vec{m}}^{\vec{n}} = \sum_{k_1=m_1}^{n_1} \sum_{k_2=m_2}^{n_2} \sum_{k_3=m_3}^{n_3},$$

where $\lfloor \cdot \rfloor$ are Gauss Brackets and $\binom{n}{k}$ denotes the binomial coefficient, we can write the Shakeshaft integral (3.3) in our notation as following:

$$\begin{aligned} I(n_1+1,\vec{l}_1,n_2+1,\vec{l}_2) = 2\pi(-\imath)^{\vec{l}_1+\vec{l}_2} \int_0^1 dy \sum_{\vec{k}_1=\vec{0}}^{\vec{l}_1} \sum_{\vec{k}_2=\vec{0}}^{\vec{l}_2} \binom{\vec{l}_1}{\vec{k}_1}\binom{\vec{l}_2}{\vec{k}_2} (\imath y\vec{R})^{\vec{k}_1} (-\imath(1-y)\vec{R})^{\vec{k}_2} \\ \times \sum_{m_1=0}^{\lfloor n_1/2 \rfloor} \sum_{m_2=0}^{\lfloor n_2/2 \rfloor} D_{m_1}^{n_1} D_{m_2}^{n_2} c^{n_1-2m_1} d^{n_2-2m_2} y^{n_1-m_1} (1-y)^{n_2-m_2} \\ \times \sum_{\vec{m}=\vec{0}}^{\lfloor \vec{n}/2 \rfloor} D_{\vec{m}}^{\vec{n}} (\vec{a}+\vec{b})^{\vec{n}-2\vec{m}} (y(1-y))^{\vec{n}-\vec{m}} \times \sum_{j=0}^{\mathcal{N}} F_j^{\mathcal{N}} (AR)^j \frac{\exp(\imath\vec{B}\vec{R}-AR)}{A^{2\mathcal{N}+1}} \end{aligned} \tag{3.7}$$

where F_j^k and D_m^n are coefficients (see later), A and $\vec{B}$ are as defined in the previous section and we introduced the constants $\vec{n} := \vec{l}_1+\vec{l}_2-\vec{k}_1-\vec{k}_2$ and $\mathcal{N} := n_1+n_2-m_1-m_2+$

[7] Originally, it was called a *closed-form expresssion*, however, I will avoid this term because it may erroneously suggest an analytic expression for the *integral* (in y), rather than just for the *integrand*. Technically, the differentiation method by Shakeshaft translates to finite recurrence formulae (3.4) and is as such already of closed form.

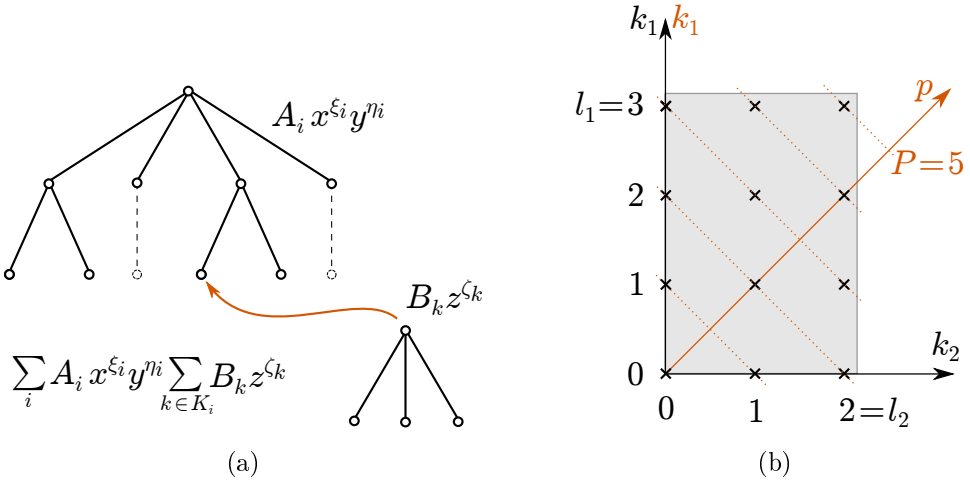


Figure 3.4: Building the symbolic structures (3.7): (a) Construction of nested the sums by "stacking" cached trees onto the parent's leaf nodes, (b) Re-indexing of the angular sums in order to allow stacking.

$n - m$. Note that this definition of $\mathcal{N}$ corrects the one in the original paper [41], which erroneously omitted m_1 and m_2. Moreover, please note that Kocbach and Liska used the symbols $\omega = 1 - y$ and $\vec{x} = -\vec{B}$ (which relates to $k_1 \leftrightarrow k_2$). Finally, they looked at n from a "computational" perspective and let it be equal to the number of differentiations ≥ 0, whereas Shakeshaft took the "physical" association with the principal quantum number $n \geq 1$ (see also Eq. 3.1) [53].

We immediately see that symmetries of the trajectory, i. e., zero components of $\vec{R}$ and $\vec{v} = \vec{a} + \vec{b}$, relate to collapsing sums in (3.7): if the corresponding n is even, then only one term $m = n/2$ survives, if n is odd, then the matrix element vanishes completely. This cancels a few of the 12 sums, making the computation easier.

Next, when examining the four factors separated by $\times$, we can clearly identify the angular part (first factor), the velocity-dependent part (third factor) and the radial part (second and fourth factor). Moreover, we see that the fourth factor decouples from the rest of the sum, allowing us to cache its terms completely and adding it later. Also, the other parts only couple via their y-dependency.

Remembering that the polynomials are represented as n-ary tree structures, a (direct) multiplication $p \otimes q$ of two polynomials essentially means substituting each leaf nodes of p by a multiple of q (see Figure 3.4a). More generally, for a "stacked sum", where the boundaries of the inner sum depend on the outer element, we can build a cache of trees that we can stack into each other.

In order to do that, we first take care of the coupling of the angular and the velocity part: The inner sum over the velocity part depends on $k_1 + k_2$, which we reduce to a

single dependency be re-indexing the outer sum over the angular part to

$$\sum_{\vec{k}_1=\vec{0}}^{\vec{l}_1}\sum_{\vec{k}_2=\vec{0}}^{\vec{l}_2} = \sum_{\vec{p}=\vec{0}}^{\vec{P}}\sum_{\vec{k}_1=\vec{K}^-}^{\vec{K}^+}$$

where we have introduced $\vec{p} = \vec{k}_1 + \vec{k}_1$ and $\vec{P} = \vec{l}_1 + \vec{l}_2$ (see Figure 3.4b). The boundaries for the $\vec{k}_1$ summations can easily be shown to be $\vec{K}^- = \max(0, \vec{p} - \vec{l}_2)$ and $\vec{K}^+ = \min(\vec{p}, \vec{l}_1)$. This definition aligns perfectly with our stacking intention, since the upper boundary for the velocity-dependent part now simplifies to $\vec{n} = \vec{P} - \vec{p}$.

3.4 Solution of the differential equation

To now compute the transition amplitudes, we must find a numerical solution to the coupled channels equations (2.22). In a first step, we multiply with S^{-1} from the left and arrive at:

$$\frac{d}{dt}\vec{a}(t) = -\imath S^{-1} M \cdot \vec{a} =: M_{\text{eff}}(t)\vec{a}(t); \quad \vec{a}(t \to -\infty) = \vec{\delta}_k \tag{3.8}$$

To simplify our equations and to be consistent with the notation commonly used in the underlying mathematical theory, we identify $\vec{a} =: y$ (dropping the vector arrow) and $M_{\text{eff}} =: A$ and rewrite this equation to:

$$y'(t) = A(t) \cdot y(t) \quad \text{with} \quad y(t_0) = y_0, \tag{3.9}$$

which is the canonical form of the so-called initial-value problem for a set of *matrix differential equations* (MDEs).

Ordinary differential equations Our matrix differential equation (3.9) is a specialization of an *ordinary differential equation* (ODE) of rank 1:

$$y'(t) = f(t, y(t)) \quad \text{with} \quad y(t_0) = y_0 \tag{3.10}$$

where $y \in \mathbb{R}^n$ is a real vector-value function, t again is a real parameter and the function $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ provides the derivatives ("right-hand side") of the equations [54].

This is usually the most general form of equations we have to solve: we can reduce equations over the complex field to this form simply by treating the real and the imaginary part separately, thus doubling the number of equations.[8] For this reason, there is a large amount of different approaches and lots of libraries available dealing with

[8] In a similar fashion, we can reduce an equation of higher order $k > 1$ by adding additional "dummy" variables for higher derivatives of $y^{(n)}$ to the result vector.

the problem (3.10). These approaches can be classified by (i) computational effort, (ii) accuracy and (iii) implementation complexity.

In the close coupling case, all computational effort is negligible against the evalutation of the matrix A at some point t. We therefore search for algorithms that

- use as few matrix evaluations per step as possible,
- while providing the greatest accuracy and numerical stability.

Implicit Adams method These requirements make a strong case for the *Adams method*: solvers based on this technique are usually very complicated and comparatively slow, but provide extremely high accuracy [54].

As with most ODE solvers, the Adams method "walks" through time domain $[t_0, t_1]$ in a series of small steps (y_n, t_n). In each step n it computes y_{n+1} based on y_n and its differentials. However, the Adams method is a *multi-step* method, which in this case means that it not only considers the current step (y_n, t_n) in the computation of the next step, but also some of the "history" of the trajectory (y_{n-k}, t_{n-k}).

In its explicit form, an Adams method of order k lays an interpolation polynomial through the previous k values of the result vector and uses this information together with the differential at the current point to extrapolate the vector. Implicit forms also account for the fact that interpolation does not translate well to extrapolation and, as a result, iteratively use the proposed result vector in the interpolation procedure.

The `SHAGO2` solver subroutine included in the programme also supports *order adoption* and *step size control*: Here, the order and the step size are changed dynamically with the local properties of the ODE. This minimises the number of costly matrix evaluations.

3.5 Spline integration of cross sections

When numerically evaluating the integral

$$\sigma_{lm} = \int_0^\infty 2\pi b \, db \, P_{lm}(b) \quad \text{with} \quad P_{lm}(b) = \sum_k |a_{lm}^k|^2$$

we need to take care, because the integrand is known on an non-equally spaced grid $(b_i, P_i = P_{lm}(b_i))$ only. As P_{lm} has high dynamics in b, we would like to use a higher-order Newton-Cotês method, which unfortunately requires an equidistant mesh.

Our first, straight-forward approach, therefore, was to choose an equidistant grid $\tilde{b}_i := i\tilde{h}$ and use a third-order polynomial P_3 to interpolate our function at the new grid points

$$\tilde{P}_i := P_3(P, i_1, i_2, i_3; \tilde{b}_i)$$

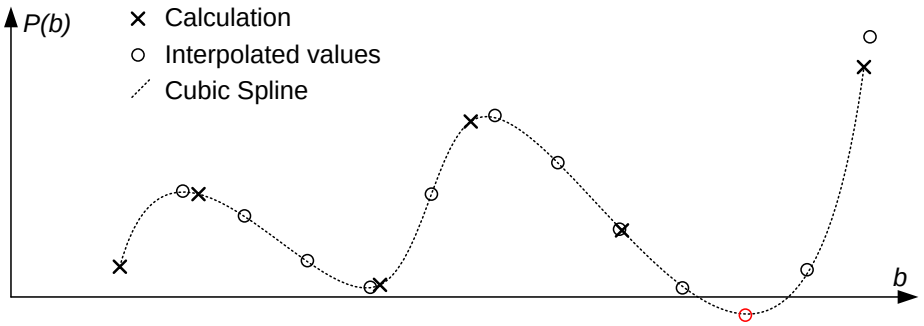


Figure 3.5: Interpolation of $P(b)$, given in arbitrary units at discrete points $\times$, to an equally spaced grid $\circ$. For comparison, a cubic free spline (dotted line) was laid through the grid points.

where the three nearest grid points i_k are used. Now, we can use the Simpson 3/8 rule on the interpolated function $(\tilde{b}_i, \tilde{P}_i)$ to calculate the integral (see Figure 3.5).

This algorithm, however, poses a series of problems, both numerically and physically:

1. The re-discretisation $b_i \mapsto \tilde{b}_i$ of the function results in a mutation of the original. More specifically, if the grid is chosen badly, the dynamics of the function may significantly increase.

2. By choosing a midpoint for the integration rather than integrating the polynomial directly, the approximation may suffer from Runge's phenomenon [55]: as we add grid points and polynomial orders, we actually *lose* local convergence.

3. As the dynamics rise, values of $\tilde{P}_i$ may become negative, which does not make any (physical) sense.

Trapezoid rule As a result, we fall back to a first-order method: we substitute our function by a piecewise linear curve through the grid points and use this function as the integrand:

$$I = \sum_{i=1}^{n-1} \frac{h_i}{2} \left(P_{i+1} + P_i\right) + O(h^3) \quad \text{with} \quad h_i = b_{i+1} - b_i$$

which is also known as the trapezoid rule, because we are summing up trapezoids between the grid points. Unfortunately, this method converges only with $O(h^3)$, which may be insufficient when we evaluating on a widely-spaced grid.

Spline interpolation To gain precision, we refine our method by integrating over splines. A spline is a piecewise cubic curve through the grid points (b_i, P_i). To guarantee a smooth transition between the curves, we require all derivatives $n \leq 2$ to be continuous

Integration method	Order	Result	Relative error
Simpson 3/8 on interpolated grid	varying	1.0372736	3.72%
Trapezoid rule	h^3	0.98710120	1.29%
Cubic spline integration	h^5	0.99956334	0.04%

Table 3.6: Approximations for $\int_0^{\pi/2} d\alpha \, \sin\alpha$, discretised in points $\{0, 0.2, 0.4, 0.9, 1.3, \frac{\pi}{2}\}$. The exact value of this integral is 1.

at the grid points $s_i^{(n)} = s_{i+1}^{(n)}$. With this constraints, it can be shown that the spline s is unique up to boundary conditions [45]:

$$\begin{aligned} s_P(b) &= AP_i + BP_{i+1} + CP_i'' + DP_{i+1}'' \quad \text{for} \quad b \in [b_i, b_{i+1}] \\ A &= \frac{b - b_i}{h_i}, \qquad B = 1 - A = \frac{b_{i+1} - b}{h_i} \\ C &= \frac{h_i^2}{6}(A^3 - A), \qquad D = \frac{h_i^2}{6}(B^3 - B) \end{aligned}$$

We use a free spline by requiring vanishing second derivatives at the scope boundaries $s''|_{\partial\Gamma} = 0$. If we substitute the integral over the function with the sum of the integrals of all spine segments, we get:

$$I = \sum_{i=1}^{n-1} \left[\frac{h_i}{2} \left(P_{i+1} + P_i\right) - \frac{h_i^3}{24} \left(P_{i+1}'' + P_i''\right) \right] + O(h^5)$$

which generalizes the trapezoid rule and surpasses its convergence by h^2. Moreover, by using a smooth spline and integrating over the whole spline, we effectively address the high dynamics and Runge's phenomenon.

To illustrate the convergence of the algorithms, we integrate

$$\int_0^{\frac{\pi}{2}} d\alpha \, \sin\alpha$$

, where $\sin\alpha$ is discretised in 6 points 0, 0.2, 0.4, 0.9, 1.3, $\frac{\pi}{2}$. This is a smooth discretisation compared to our (b, P) representations, so we expect all algorithms to yield values close to 1. For this example, the spline interpolation shows the best performance whereas our first approach shows the poorest one for $b = 0.2$ (see Table 3.6).

Negative values may still occur in a spline, but they are less likely because the lower order of the method and the requirement of continuity at the grid points "tightens" the curve around the grid. Moreover, if such values occur they are usually very small compared to a full grid point placed in the negative domain, which has a much greater "weight" in the interpolation.

4 Results

In this chapter, we present selected results that we calculated with the close-coupling method in its semi-classical impact parameter formulation (see Chapter 2) using our implementation (see Chapter 3).

As mentioned already in the introduction, we focus on the *charge exchange* process in the collisions of ions A^{q+} fully stripped of their electrons with neutral hydrogen

$$\mathrm{A}^{q+} + \mathrm{H}(1s) \rightarrow \mathrm{A}^{*(q-1)+}(n,l) + \mathrm{H}^{+}$$

for intermediate impact energies E_{kin} between 1 keV and 200 keV. Charge exchange processes at such energies are very important in fusion plasmas: when heavy atoms are induced into the deuterium plasma, they become highly ionised and migrate into the plasma as *high-q impurities* [56]. One distinguishes:

1. *Intrinsic impurities*, which are sputtered from the plasma facing components ("reactor wall") by the passing fuel stream. In modern *tokamaks*, mostly carbon and tungsten are used because of their excellent heat absorption properties [57].

2. *Artificial impurities*, which may be used to cool down the outer region of the plasma before it hits the plasma facing components [58] and mitigate disruption damages cause by plasma instabilities [59].

These impurities can now be used to diagnose the plasma, because they may interact through charge exchange with hydrogen heating beams or specifically designed neutral probe beams. The subsequent de-excitation of the ion

$$\mathrm{A}^{**(q-1)+}(n,l) \rightarrow \mathrm{A}^{*(q-1)+}(n',l') + h\nu$$

emits characteristic radiation, which is used in *charge exchange recombination spectroscopy* (CXRS) to measure plasma temperature and density [56].

We focus on the treatment of neon, because it has been proposed for plasma edge cooling in the ITER experimental tokamak reactor. However, not only fully stripped neon, but also N^{8+} and N^{9+} are important in a fusion plasma (see Figure 4.1). As shown by Igenbergs *et al.* [61], these ions can be sufficiently well approximated by the fully stripped ions O^{8+} and F^{9+}, respectively. For this reason, we also provide cross sections for these types of collisions.

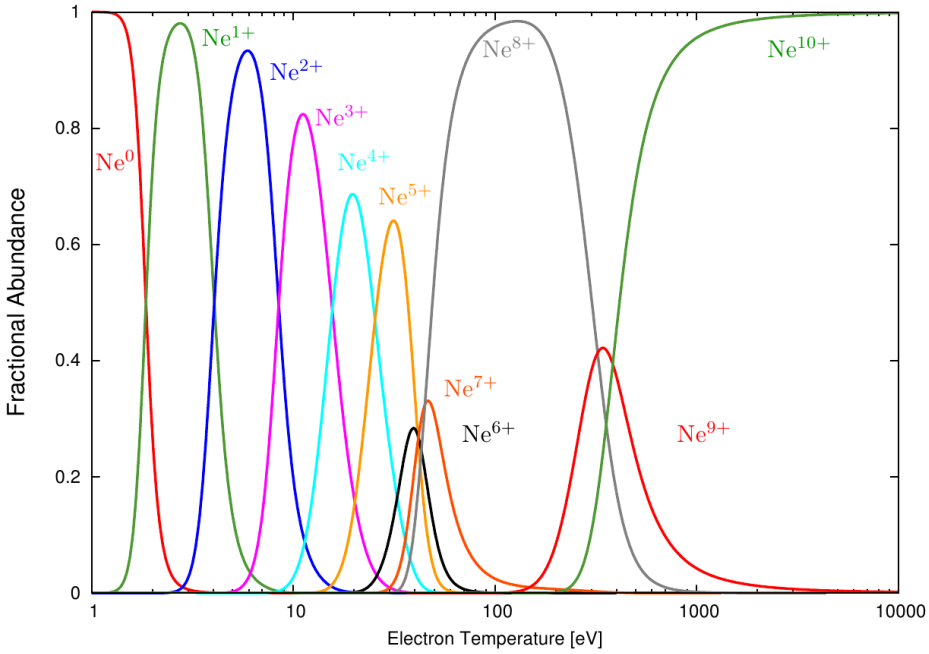


Figure 4.1: Fractional abundancies of Neon ions for different plasma temperatures [60], which are defined as the fractions of specific ions with a certain charge state.

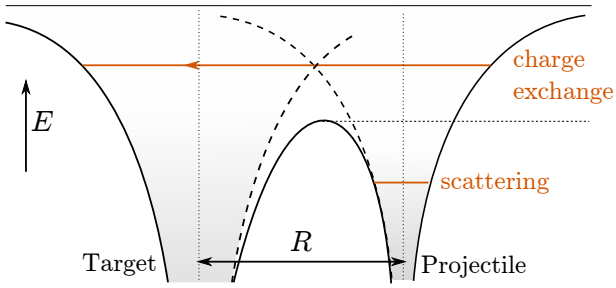
4.1 Classical cross sections and convergence

In order to understand the shape of charge exchange and ionisation cross sections, we use a well-known heuristic: the *classical over-barrier model* (COBM) originally developed by Bohr and Lindhardt in the 1940s uses the classical energy balance of the electron to decide whether it will escape the coulomb pool and switch to the other nucleus [18].

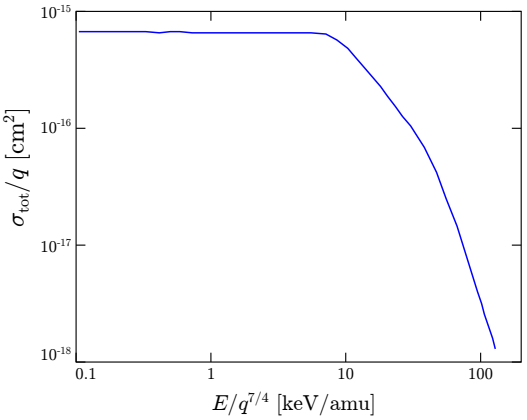
The method has been refined numerous times, but the basic idea can be seen in Figure 4.2a: between the two nuclei, one can imagine a Coulomb barrier, which is lowered as the internuclear distance R becomes smaller. If now the barrier height drops below the binding energy plus the additional electro-static energy from the target (not displayed), the electron can be captured.

This simple model yields surprisingly good charge exchange cross sections if mildly extended (see Figure 4.2b): we see that for low impact energies, the charge exchange probability is nearly constant, while for high energies, it decreases exponentially. This is in qualitative agreement with the experimental data and also with more accurate close-coupling calculations. In general, the classical over-boundary model tends to underestimate charge exchange cross sections in the mid-energy region. Semi-classically, this can be understood by the fact that the electron might tunnel through the barrier.

By examining diabatic potential curves in addition to mere consideration of potential and kinetic energies, one can also predict the main capture channel for the electron . One finds that in atom–ion collisions, the mainly populated n-shell is given by [52]:



(a) Schematic energy diagram with classical charge exchange process.



(b) Shape of the COBM charge exchange cross section (after [62])

Figure 4.2: On the classical over-barrier model (COBM)

atom	Z	n^*	number of AOs up to n^*
beryllium	4	3.16	10
nitrogen	7	5.01	35
neon	10	6.70	56
sulfur	16	10.29	220
argon	18	10.77	220

Table 4.1: Collection of main capture channels (4.1) for fully stripped ion collisions with 1s hydrogen and number of atomic orbitals $|nlm\rangle$ with $m \geq 0$ and $n \leq n^*$.

$$n^* = q/\sqrt{2I_\mathrm{p}\left(\frac{q-1}{2\sqrt{q}+1}+1\right)} \tag{4.1}$$

where I_P is the ionisation energy of the projectile (0.5 a.u. for $1s$ hydrogen). For large q this simplifies to:

$$n^*(q) \to \frac{1}{\sqrt{I_\mathrm{P}}} q^{3/4}$$

Table (4.1) collects some main capture channels for common ions colliding with 1s hydrogen. It is clear that for an accurate model of the collision process, the main capture channels must be included in the close-coupling basis set. Table (4.1) therefore also gives the number of atomic orbitals N with energies up to the main capture channel $n \leq n^*$ and with positive m, which can be calculated with the formula:

$$N = \frac{n(n+1)(n+2)}{6}$$

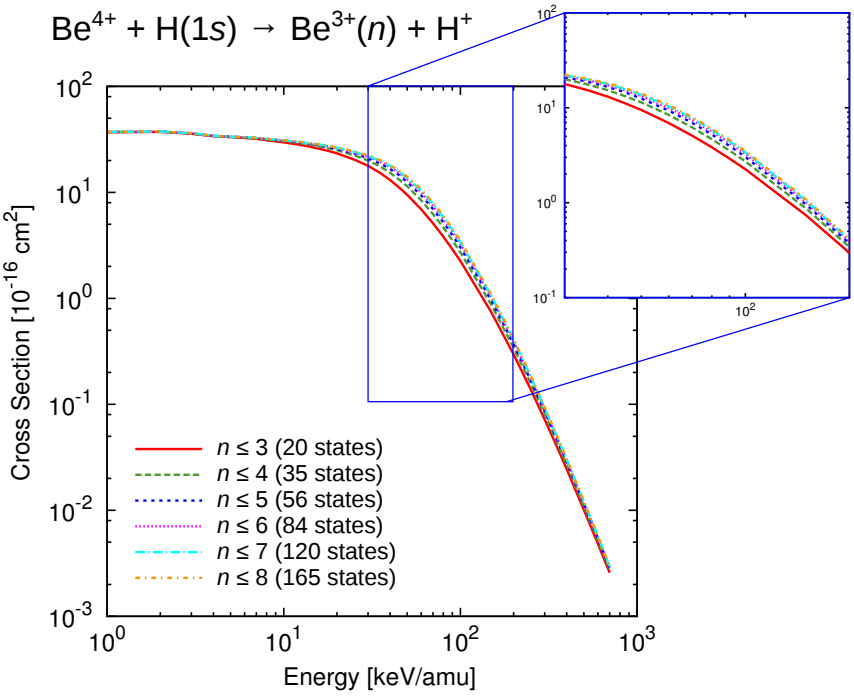


Figure 4.3: Total charge transfer cross sections of Be^{4+}–H collisions for AO basis sets of different sizes, which illustrates the convergence properties of the basis.

For large q, we easily see that the number of channels required rises approximately with $q^{9/4}$, justifying the demand for more efficient computational methods.

The main capture channel is a good guideline for the selection of the set of atomic orbitals, and as soon as it is included, the calculation starts to *converge*. Convergence in this case means that the addition of further basis states does not significantly change the resulting probability amplitudes and cross sections. Although this definition of convergence cannot be put into mathematically rigorous terms, it is nevertheless an important hint that the calculation is giving meaningful (and physical) results.

Figure 4.3 shows such convergence for beryllium–hydrogen calculations: the addition of higher n-shells above the main capture channel $n = 3$ does not significantly change the total charge transfer cross sections. However, CXRS and related methods require emission lines in the visible spectrum, which makes the analysis of higher n shells necessary.

4.2 Oxygen–hydrogen collisions

The first system we examined concerned collisions of fully stripped oxygen O^{8+} with neutral hydrogen H. For the calculation, a basis of 373 states in total was used:

- 354 atomic orbitals on the target: bound states $4 \leq n \leq 12$

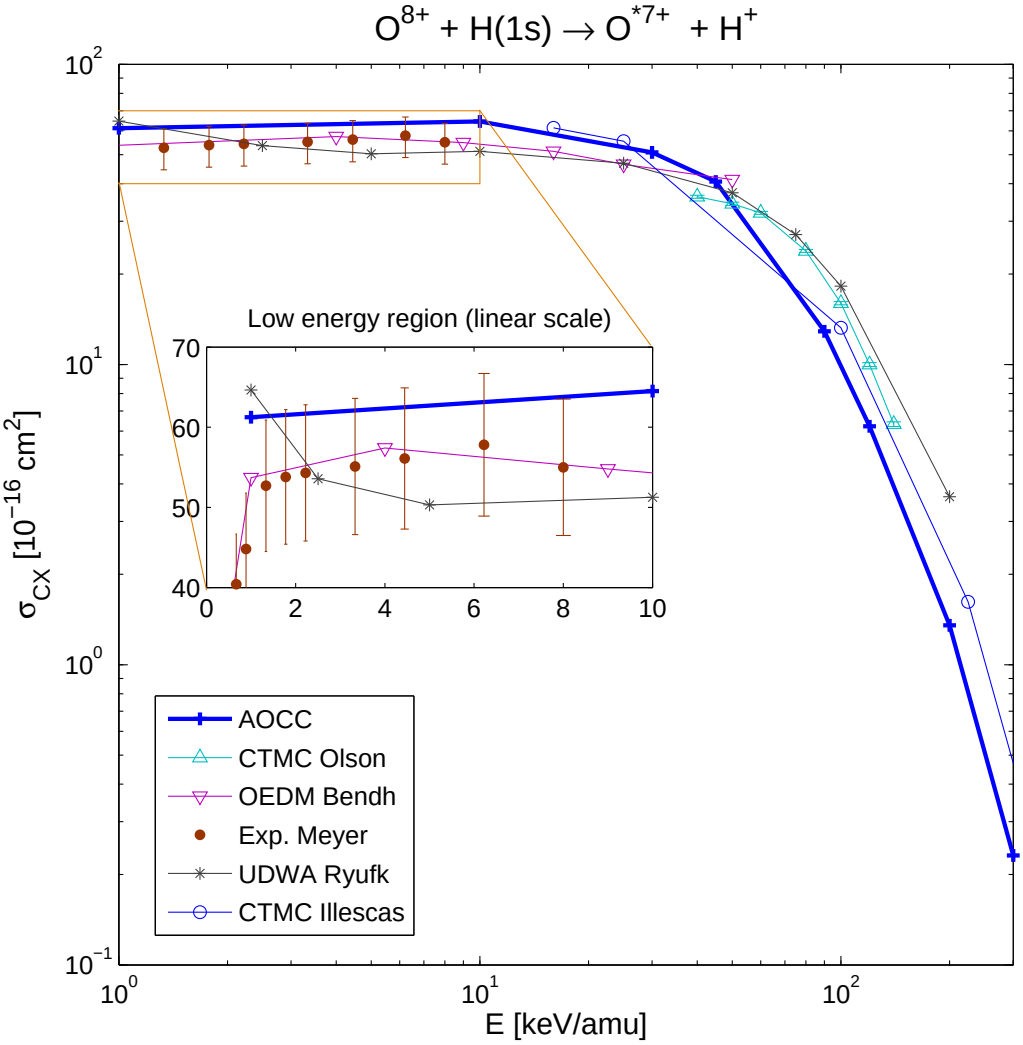


Figure 4.4: Calculated total charge transfer cross section σ_{CX} for collisions of fully stripped oxygen O^{8+} with neutral hydrogen atoms for intermediate impact energies $E_{\mathrm{kin}} \in [1, 300]\,\mathrm{keV/amu}$. The results are related to CTMC simulations (Δ) by Olson and Schultz [63], CTMC data by Illescas *et al.* [64], OEDM data (∇) by Bendahman *et al.* [65], and distorted wave calculations ($\times$) by Ryufuku and Watanabe [37]. The inset highlights the low energy region in linear scale to allow for a comparison with experimental data ($\bullet$) by Meyer *et al.* [66].

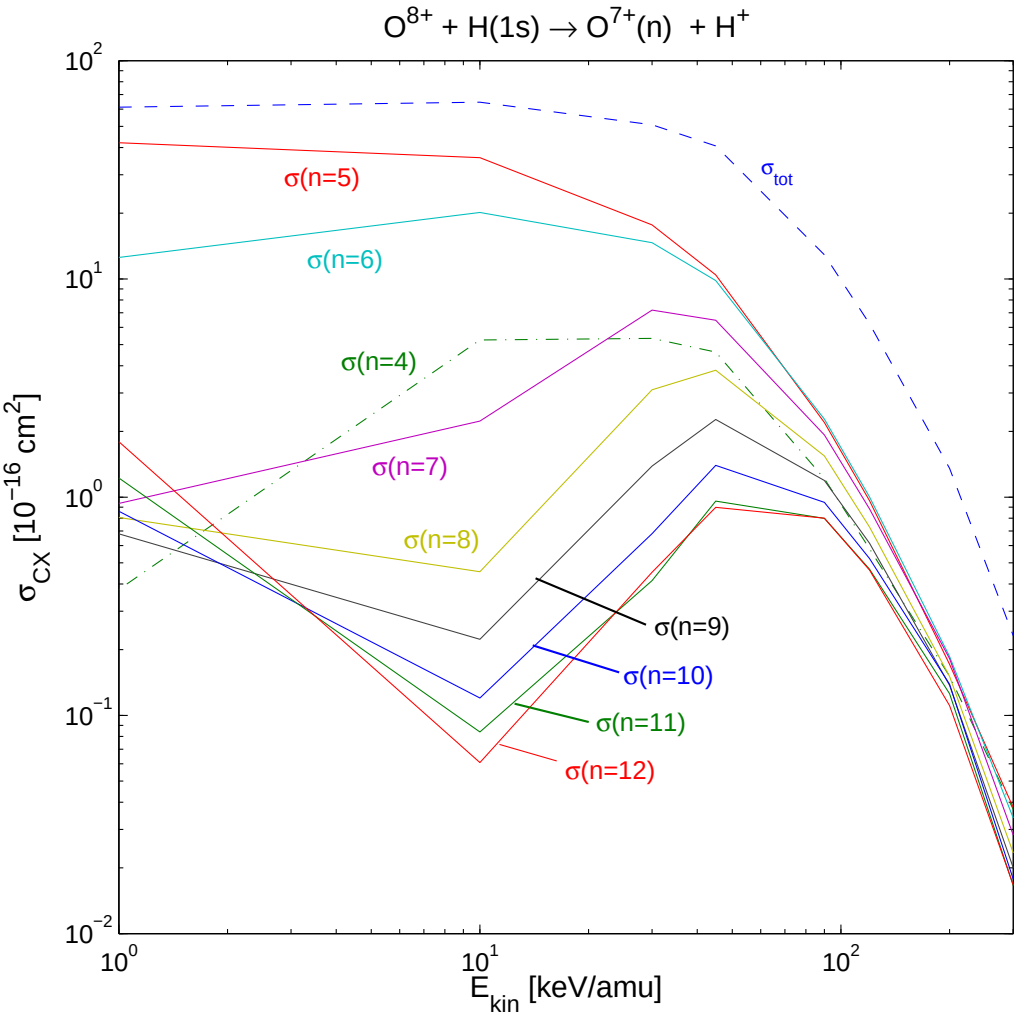


Figure 4.5: Calculated charge transfer cross sections to different n shells (capture channels) for collisions of fully stripped oxygen O^{8+} with neutral hydrogen atoms. The dashed line marks σ_{tot} (the sum of all curves), dash-dotted lines signify capture channels below the main capture level ($n < 5$) while solid lines are used for n levels equal or above $n = 5$.

- 10 atomic orbitals on the projectile: bound states $n \leq 3$
- 9 united atom pseudo-states on the projectile $n = 2, 3$

The UA states were placed on the projectile to ensure that they were assigned positive eigenenergies in the diagonalisation process (see Section 2.5.2), thus allowing us to model continuum states. However, since the main focus was charge exchange processes, only a small number of such states was included.

Figure 4.4 shows our calculation of the total cross section for the charge exchange process

$$\mathrm{O}^{8+} + \mathrm{H}(1s) \rightarrow \mathrm{O}^{*7+} + \mathrm{H}^{+}.$$

We see that in the low-energy region, our results are in good agreement with the measurements (•) by Meyer *et al.* [66] and the one-electron diatomic molecule (OEDM) calculations (+) by [65]. Especially considering the unitarised distorted-wave approximation (UDWA) calculations (×) by Ryufuku and Watanabe [37], we see that our results overestimate the charge exchange cross section here. This might be due to the fact that in this region, the simplification with atomic orbitals already starts to break down. In the medium to high energy region, the results show the well-known decline. Though there is a significant difference to Classical Trajectory Monte Carlo (CTMC) calculations by Olson *et al.* [63], similar calculations by Illescas *et al.* [64] match our AOCC data very well (for a short description of CTMC methods, see Section 2.6).

Figure 4.5 shows partial charge exchange cross sections into different n-levels. We can clearly identify the main capture channel $n = 5$, which agrees with the predictions of the classical over-barrier model. We also see that for $n \geq 7$, there is a local minimum in the cross sections at about $E = 10\,\mathrm{keV}$ and a global maximum around $E = 45\,\mathrm{keV}$.[1]

Figure 4.5 resolves the partial cross section (solid blue line) even further: the x axis shows the n shells, where each interval $[n, n+1)$ is split into n subintervals which correspond to the angular momentum quantum number $0 \leq l < n$. Each coloured line in turn represents one n level and the dots represent the l levels.

We see that around the main capture channel $n \approx n^*$, the mainly populated sub-shell is $l = l_{\max} = n - 1$. This agrees with observations in fusion plasmas. For higher n shells, the situation is quite different: for $E = 10\,\mathrm{keV/amu}$, the $l = 3$ sub-shell is preferred, while for $E = 45\,\mathrm{keV/amu}$, the $l = 5$ and $l = 6$ sub-shell is populated strongest. Qualitatively, this can be understood by angular momentum conservation [52]: initially, the electron carries no "atomic" angular momentum ($l = 0$) but it does carry "trajectory" angular momentum $L = vb$ with respect to the target. While the nucleus preserves this angular momentum, the electron loses it when switching to the target. One now assumes that in the process of switching, the "trajectory" angular

[1] The location of these local extrema is somewhat inaccurate, since the discretisation on the energy axis is fairly coarse.

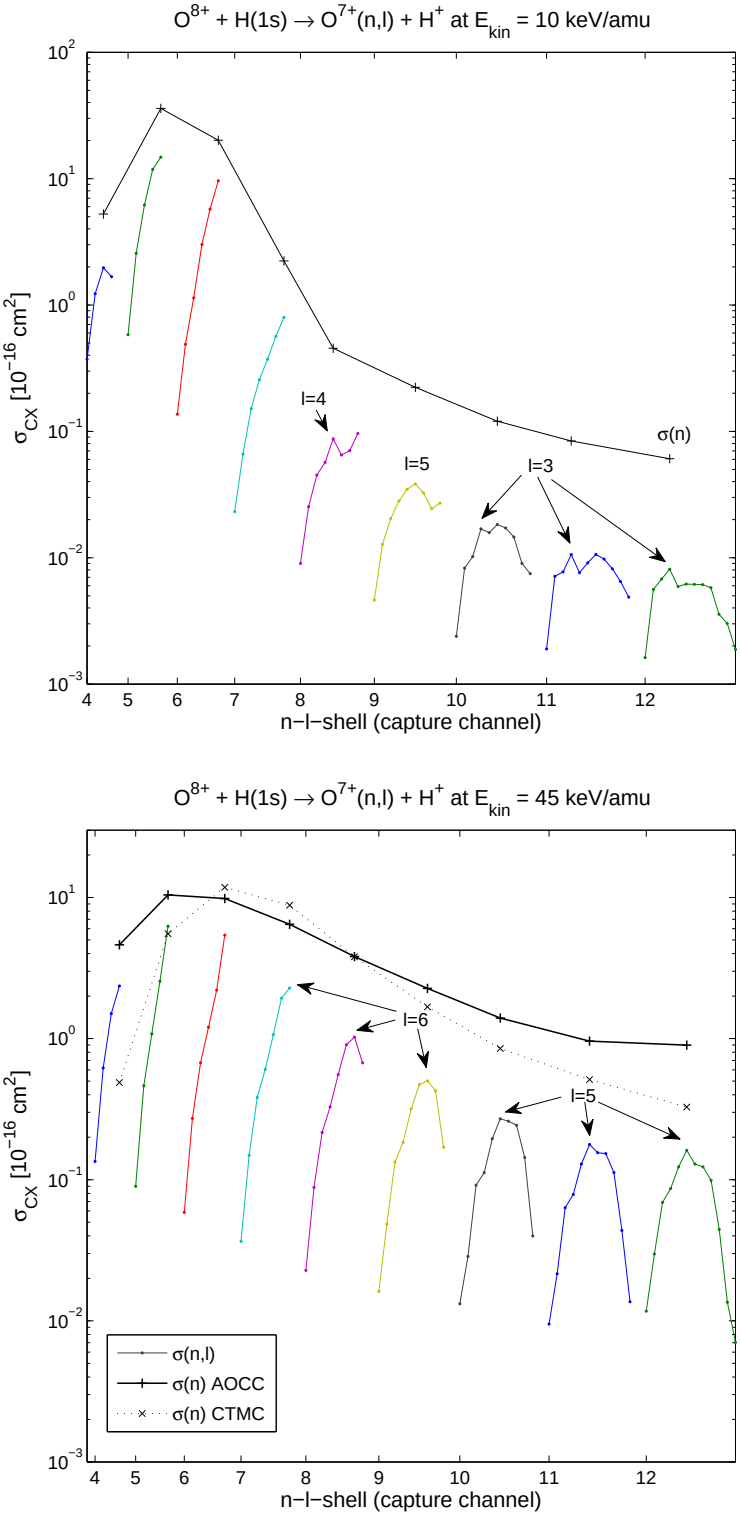


Figure 4.6: Calculated charge transfer cross sections to different n, l sub-shells for H–O^{8+} collisions for impact energies of 10 keV/amu (top) and 45 keV/amu (bottom). Solid coloured lines relate to different n shells ($4 \leq n \leq 12$), with the dots marking the l levels ($0 \leq l < n$). The calculated n resolved cross section (+) is compared with interpolated CTMC data (×) by Olson and Schultz [63].

momentum converts itself to "atomic" angular momentum $\langle L \rangle = \sqrt{l(l+1)}$. This means that for greater impact energy and hence greater angular momentum, the populated l sub-shell for charge exchange processes is greater.[2]

4.3 Fluorine–hydrogen collisions

The next system we studied was the collisions of fully stripped oxygen F^{9+} with neutral hydrogen H. For the calculation, a basis of 464 states in total was used:

- 445 atomic orbitals on the target: bound states $4 \leq n \leq 13$
- 10 atomic orbitals on the projectile: bound states $n \leq 3$
- 9 united atom pseudo-states on the projectile $n = 2, 3$

The UA states were again placed on the projectile to ensure that they were assigned positive eigenenergies in the diagonalisation.

The calculated total cross section for the charge exchange process

$$F^{9+} + H(1s) \rightarrow F^{*8+} + H^{+}$$

is presented in Figure 4.7. We see that in the low-energy region, we achieve very good agreement with both the experimental data (•) by Meyer *et al.* [66] and the one-electron diatomic molecule (OEDM) calculations (∇) by [65].

Since there is very little data on this type of collisions, the classical over-barrier (COBM) cross section (see Section 4.1) was plotted along with the data. We see the common behaviour that the COBM underestimates the charge exchange cross section in the medium energy region, while it is more accurate in the low and high energy regime.

Figure 4.8 shows partial charge exchange cross sections to different n levels. We can clearly identify the main capture channel $n = 6$, which again agrees with the predictions of the classical over-barrier model. We also see the extrema we encountered at oxygen collisions: for $n \geq 8$, there is a local minimum in the cross sections at about $E = 10\,\mathrm{keV/amu}$. The global maximum shifts to higher energies with greater n and is placed around $E = 45\,\mathrm{keV/amu}$ for $n \approx 9$.

Figure 4.9 resolves the partial cross section (solid blue line) to n, l sub-shells. Around the main capture channel, again the highest l sub-shell is strongly populated. For $E = 10\,\mathrm{keV/amu}$, the $l = 6$ sub-shell is preferred in higher shells ($n \approx 10$) and $l = 5$ is populated preferred very high n levels. For $E = 45\,\mathrm{keV/amu}$, the sub-shells populated most strongly above the main capture channel are $l = 7$ and $l = 6$, respectively. This is again in agreement with our angular momentum considerations.

[2] This method is however unsuited for the quantitative treatment of cross sections, since a weighing process over many impact parameters $\int 2bdb$ takes place

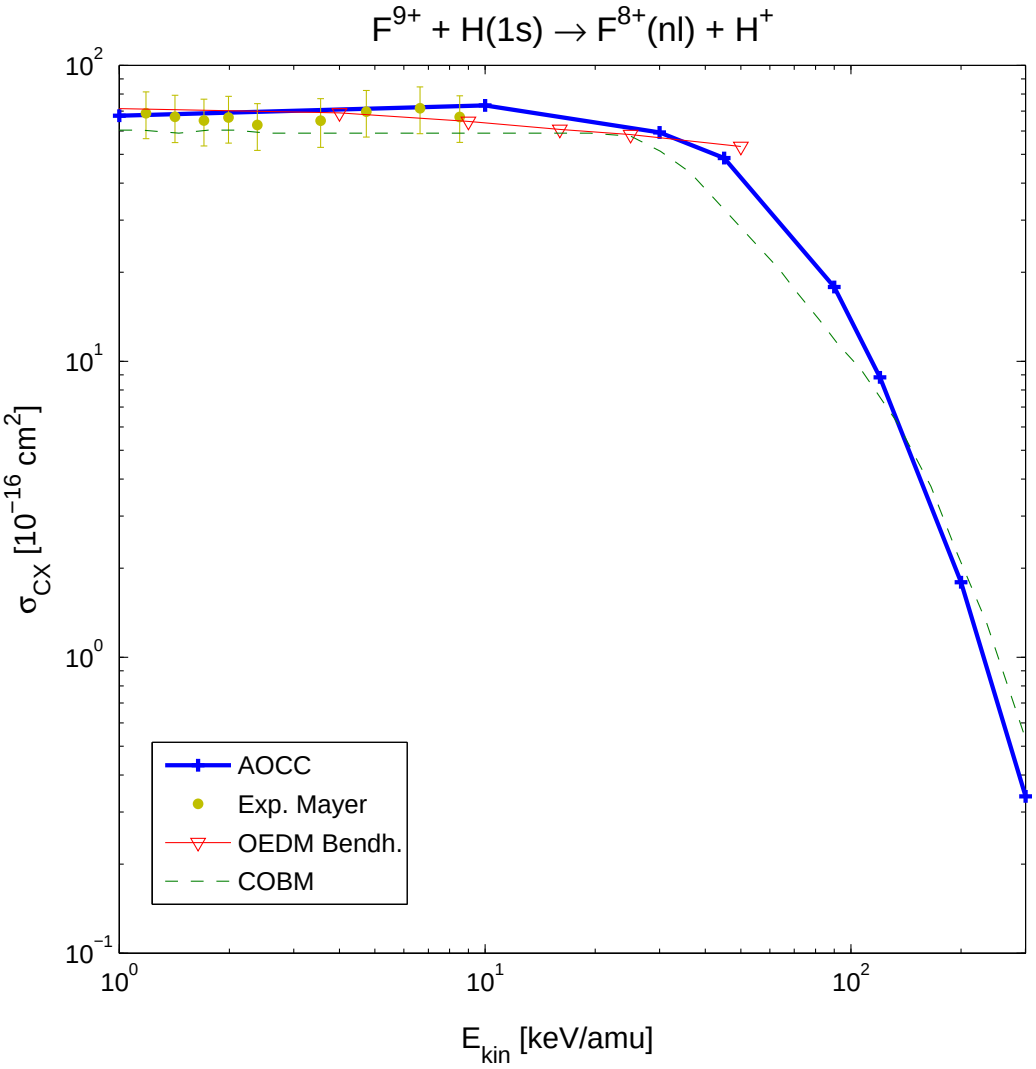


Figure 4.7: Calculated total charge transfer cross section σ_{CX} for collisions of fully stripped fluorine F^{9+} with neutral hydrogen atoms for intermediate impact energies $E_{\mathrm{kin}} \in [1, 300]\,\mathrm{keV/amu}$. The results are related to OEDM data (∇) by Bendahman *et al.* [65], experimental data ($\bullet$) by Meyer *et al.* [66] and classical over-barrier considerations [62].

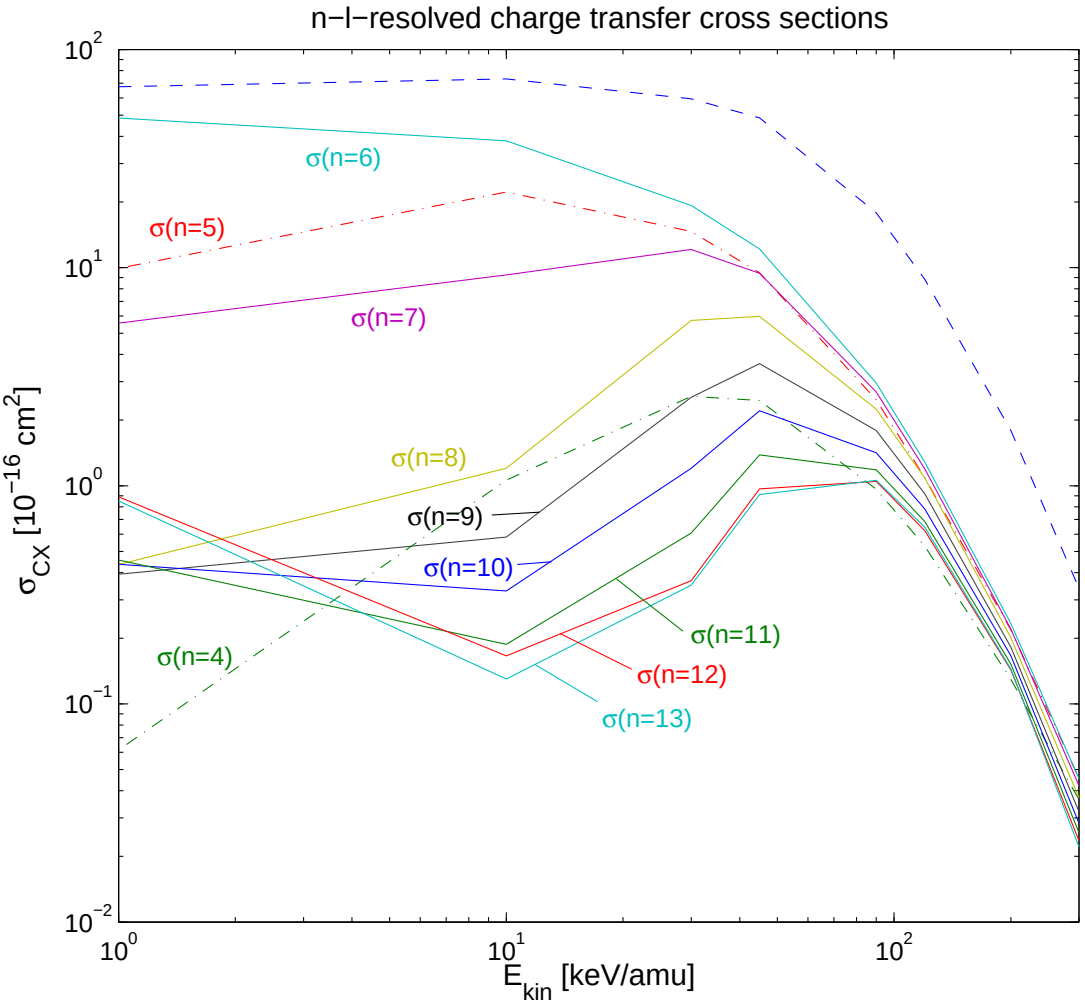


Figure 4.8: Calculated charge transfer cross sections to different n shells (capture channels) for collisions of fully stripped fluorine F^{9+} with neutral hydrogen atoms. The dashed line marks σ_{tot} (the sum of all other curves), dash-dotted lines signify capture channels below the main capture level ($n < 6$) while solid lines are used for n levels equal or above $n = 6$.

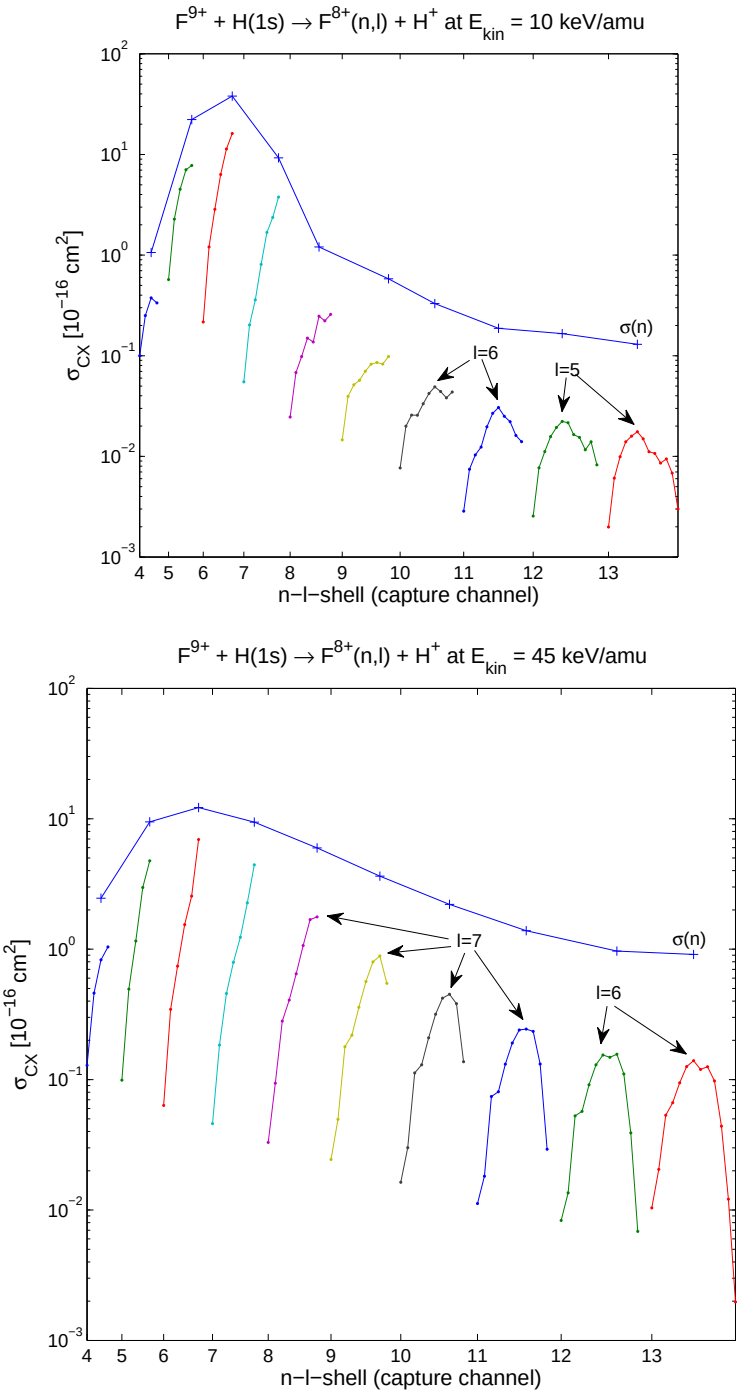


Figure 4.9: Calculated charge transfer cross sections to different n, l sub-shells for H–F^{9+} collisions for impact energies of 10 keV/amu (top) and 45 keV/amu (bottom). Solid coloured lines relate to different n-shells ($4 \leq n \leq 13$), with the dots marking the l levels ($0 \leq l < n$). + marks calculated n-resolved cross section.

4.4 Neon–hydrogen collisions

The final collision system we looked at was fully stripped neon Ne^{10+} on neutral hydrogen H. This system is particularly interesting for fusion research, because neon and argon have been proposed for plasma edge cooling in the ITER experimental tokamak reactor (see above). For the calculation, a basis of 569 states in total was used:

- 550 atomic orbitals on the target: bound states $4 \leq n \leq 14$
- 10 atomic orbitals on the projectile: bound states $n \leq 3$
- 9 united atom pseudo-states on the projectile $n = 2, 3$

The UA states were again placed on the projectile to ensure that they were assigned positive eigenenergies in the diagonalisation.

4.4.1 Total charge exchange

The calculated total cross section for the charge exchange process

$$Ne^{10+} + H(1s) \rightarrow Ne^{*9+} + H^{+}$$

is presented in Figure 4.7. We see that in the low-energy region, the agreement with the experiment (•) by Meyer *et al.* [66] is good, although we see the same offset as with oxygen (compare Figure 4.4).

We also compared our data to CTMC data by Errea *et al.* [67] (undecorated lines) and Schmidt *et al.* [68] (triangles). Both groups based their calculations on two different types of ensembles to draw their initial conditions from (see also Section 2.6):

1. the *micro-canonical ensemble* (solid lines in Figure 4.7), which is a momentum and angular momentum distribution to a definite energy E in a micro-canonical thermodynamic ensemble, and
2. the *hydrogenic ensemble* (dashed lines in Figure 4.7), which is a combined momentum and configuration distribution modelled after the $1s$ bound state of hydrogen.

We see that in the lower energy region, the hydrogenic ensemble yields results which are excellent agreement with our AOCC calculations. The micro-canonical ensemble on the other hand significantly under-estimates the cross sections in this region compared to our approach and experimental data.

In the medium energy region (see insert in Figure 4.7), both approaches start to converge and still fit well to our calculations.

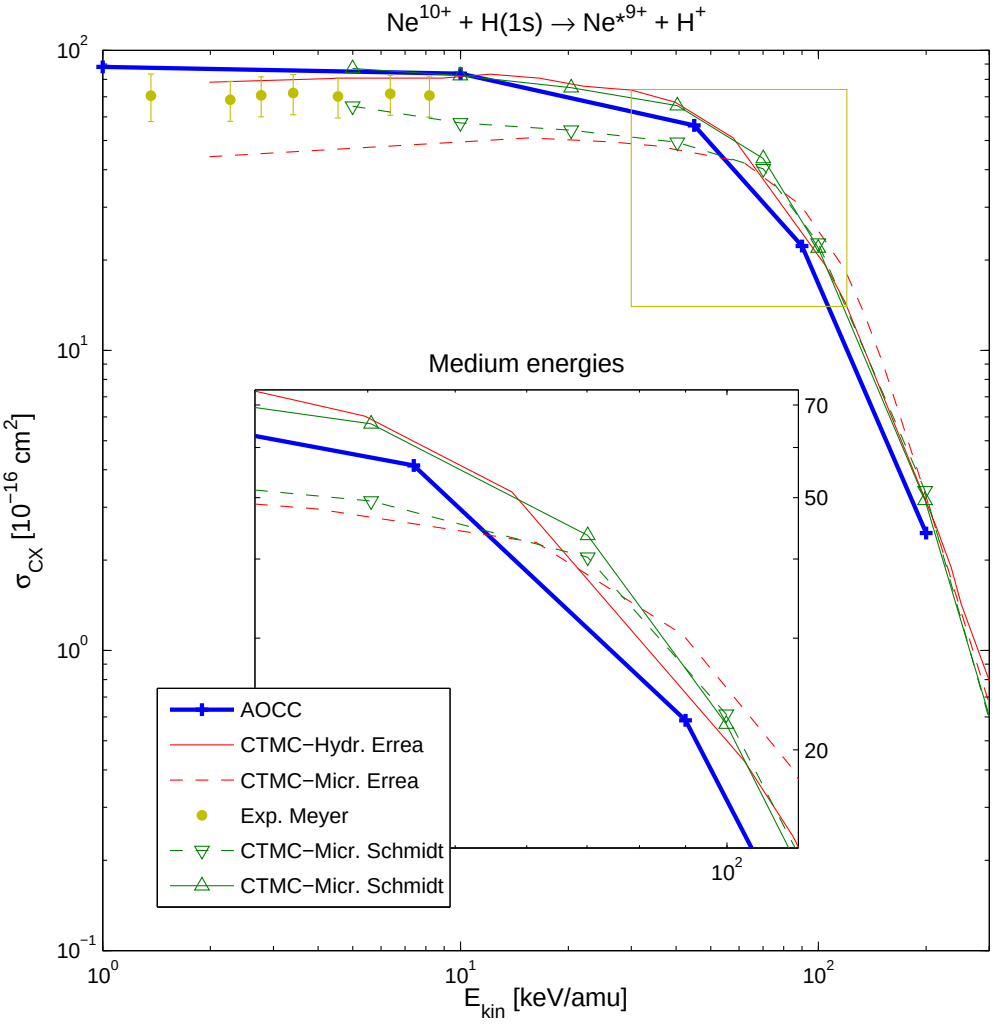


Figure 4.10: Calculated total charge transfer cross section σ_{CX} for collisions of fully stripped neon Ne^{10+} with neutral hydrogen atoms for intermediate impact energies $E_{\mathrm{kin}} \in [1, 300]\,\mathrm{keV/amu}$. The results are related to CTMC data by Errea *et al.* [67] (undecorated lines) and Schmidt *et al.* [68] (triangles). Solid lines signify CTMC calculations with a hydrogenic ensemble, whereas dashed lines mark CTMC calculations conducted with a micro-canonical ensemble. For comparison, experimental data (•) by Meyer *et al.* [66] was included.

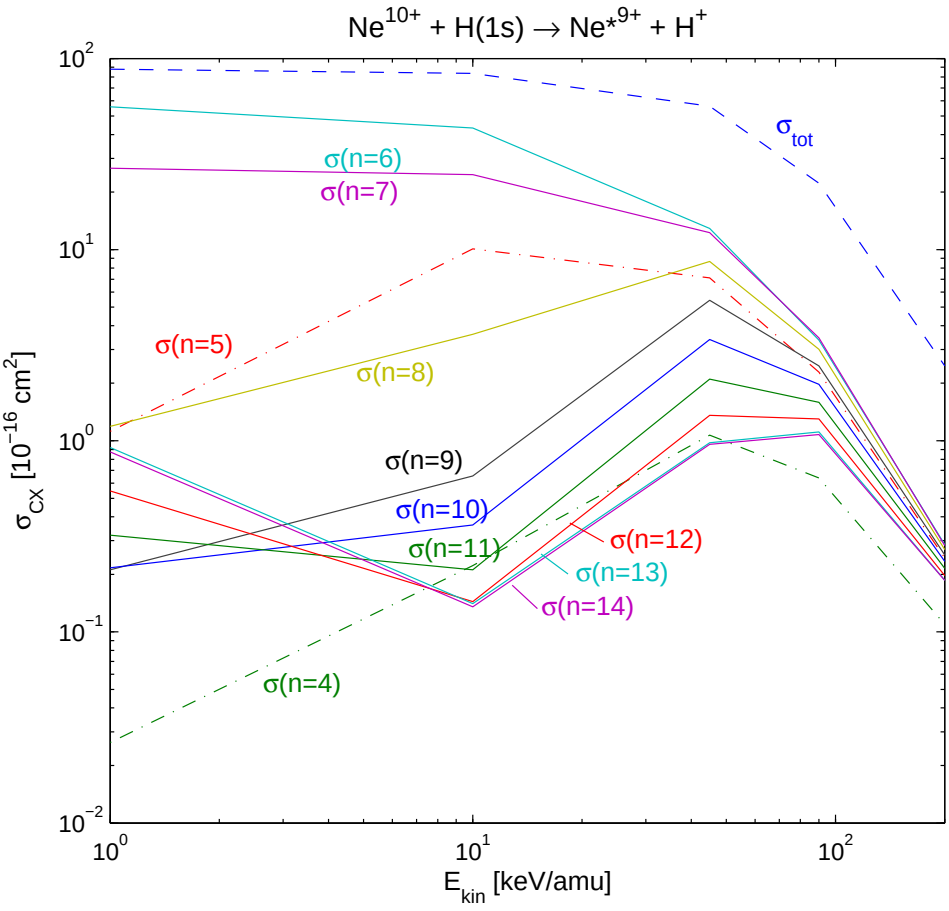


Figure 4.11: Calculated charge transfer cross sections to different n shells (capture channels) for collisions of fully stripped neon Ne^{10+} with neutral hydrogen atoms. The dashed line marks σ_{tot} (the sum of all other curves), dash-dotted lines signify capture channels below the main capture level ($n < 6$) while solid lines are used for n levels equal or above $n = 6$.

4.4.2 Partial charge exchange

Figure 4.11 shows partial charge exchange cross sections to different n levels. We can clearly identify the main capture channel $n = 6$, which again agrees with the predictions of the classical over-barrier model. We also see a significant contribution by the $n = 7$ shell, which is due to the fact that the main capture channel "6.70" is already close to 7.

We also see similar behaviour as for the fluorine collisions: for $n \geq 9$, there is a local minimum in the cross sections at about $E = 10\,\mathrm{keV/amu}$. The global maximum for $n \geq 8$ shifts to higher energies with greater n and is placed around $E = 45\,\mathrm{keV/amu}$. Figure 4.12 compares the partial charge transfer cross sections for $9 \leq n \leq 14$ with the Errea CTMC calculations [67]. We observe that for $n = 9$ and 10, the results are very similar. For higher n shells, the general shape agrees, but the CTMC results show a

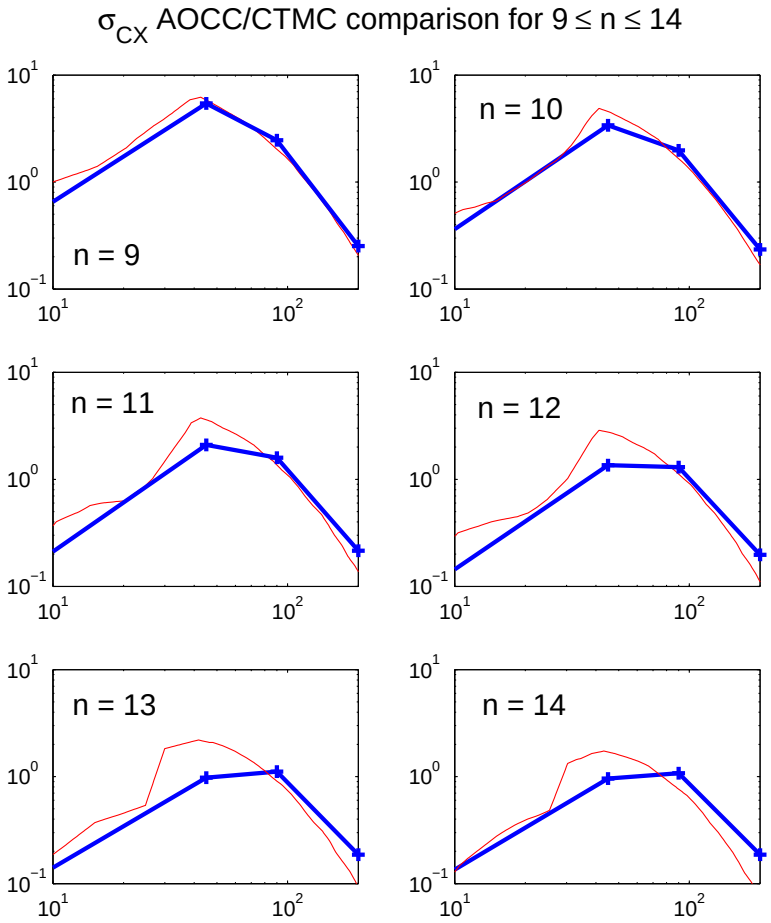


Figure 4.12: Comparison of AOCC partial charge transfer cross sections for Ne^{10+} – H collisions (+, blue lines) with CTMC calculations with a hydrogenic ensemble by Errea *et al.* [67] (red lines).

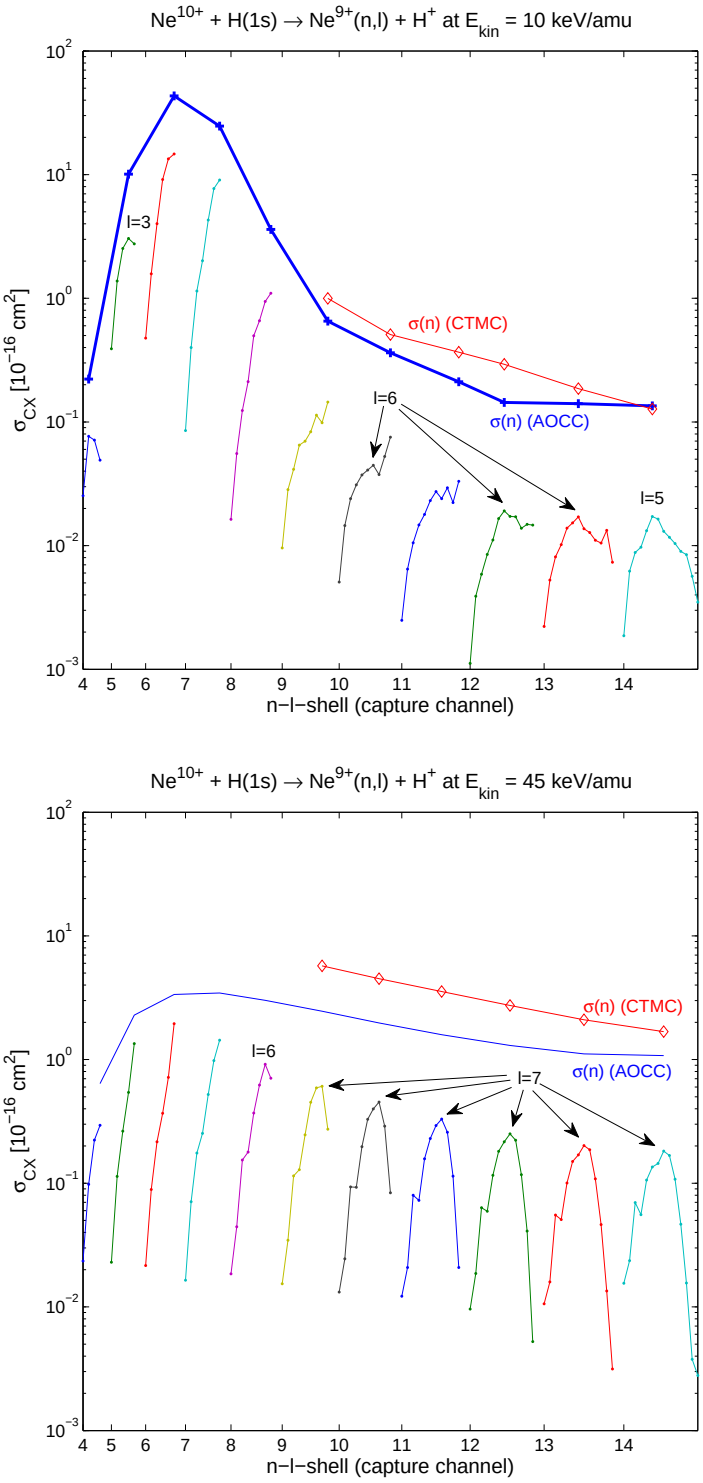


Figure 4.13: Calculated charge transfer cross sections to different n, l sub-shells for H–Ne^{10+} collisions for impact energies of 10 keV/amu (top) and 45 keV/amu (bottom). Solid coloured lines relate to different n-shells ($4 \leq n \leq 14$), with the dots marking the l levels ($0 \leq l < n$). The calculated n-resolved cross section (+) is compared with interpolated CTMC data ($\diamond$) by Errea *et al.* [67].

"hump" around $E = 50\,\mathrm{keV/amu}$, which grows larger as n increases. This behaviour was not reproduced by our calculations.

Figure 4.13 resolves the partial cross section (solid blue line) to n, l sub-shells. Aside from angular momentum maximisation around the main capture channel, we see that for $E = 10\,\mathrm{keV/amu}$, the $l = 6$ sub-shell is preferred in high n levels and $l = 5$ is populated preferred very high n levels. For $E = 45\,\mathrm{keV/amu}$, the sub-shells populated most strongly is $l = 7$. This is again in agreement with our angular momentum considerations.

4.4.3 Ionisation

Figure 4.14 shows the total cross section for the ionisation process

$$\mathrm{Ne}^{10+} + \mathrm{H}(1s) \rightarrow \mathrm{Ne}^{10+} + \mathrm{H}^{+} + e^{-}.$$

Again, the results are compared to CTMC data by Errea *et al.* [67] and Schmidt *et al.* [68]. Comparing the cross sections, we observe that although the shape agrees, our results are lower than the CTMC data by one order of magnitude. This is due to the fact that we included only a small number of states to model the continuum.

4.5 Scaling

In order to relate the previous results, we used the well-known classical scaling formula by Knudsen *et al.* [62]: here, the cross sections scale with q^{-1}, and the impact energy is divided by $q^{4/7}$. Figure 4.15 compiles the results for AOCC calculations with fully stripped ions A^{q+} for $4 \leq q \leq 10$. The results for beryllium were taken from [14], the results for carbon and nitrogen have been submitted to J. Phys. B [61].

We see that in the low-energy region the scaling formula works pretty well and all cross sections are very close to each other. In the high energy region, there is also good agreement although the asymptotic behaviour is not quite correct. This is due to the fact that in order to compute the COBM cross sections, one has one free parameter α which relates to the probability for the electron to be captured. Naturally, this parameter should change with the target ion's charge, however, in this plot it is set to $\alpha = 0.25$ (see [62]).

In the medium energy region, the agreement with the COBM cross section is worst. Also, the AOCC cross sections significantly differ from each other. Since this is the region particularly interesting to fusion, the scaling law has limited applicability to this field of research.

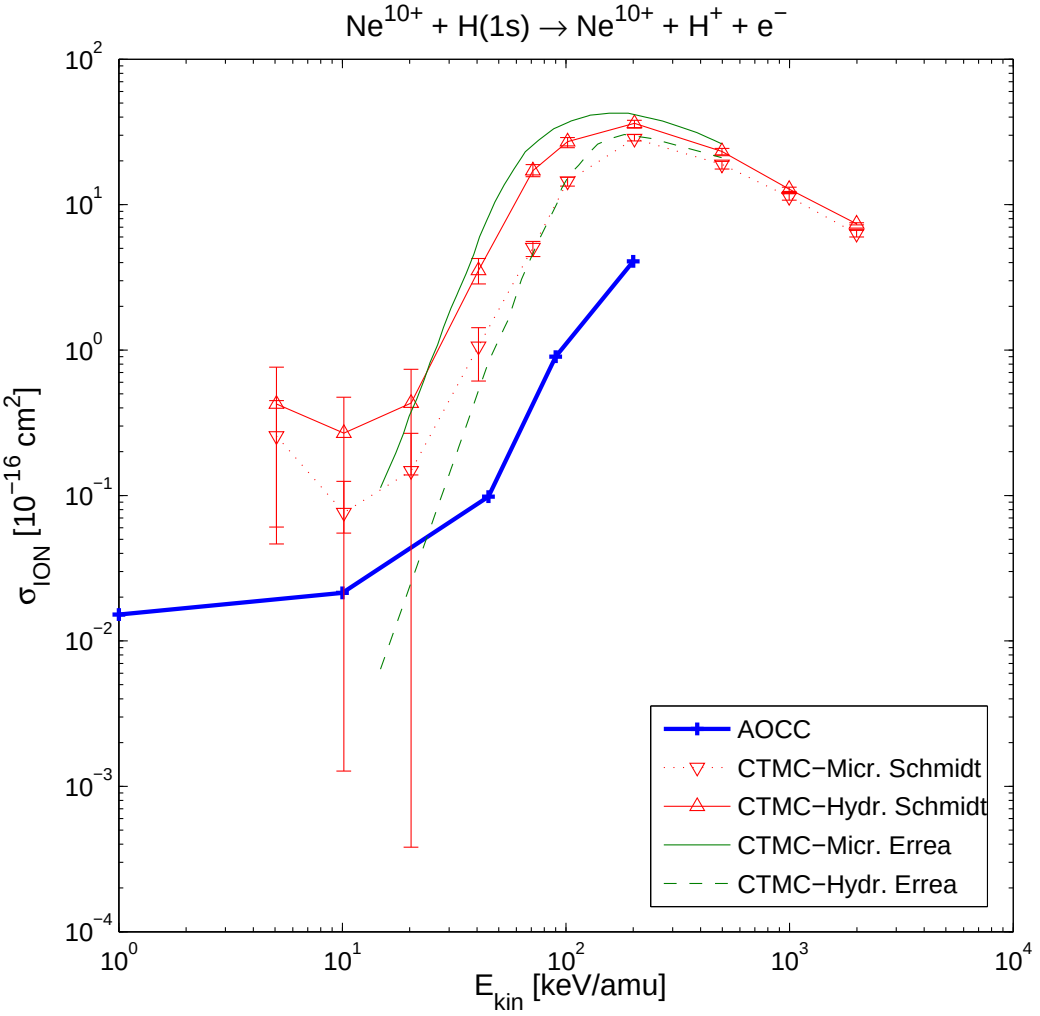


Figure 4.14: Calculated total ionisation cross section σ_{CX} for collisions of fully stripped neon Ne^{10+} with neutral hydrogen atoms for intermediate impact energies $E_{\mathrm{kin}} \in [1, 300]\,\mathrm{keV}$. The results are related to CTMC data by Errea *et al.* [67] (undecorated lines) and Schmidt *et al.* [68] (triangles). Solid lines signify CTMC calculations with a hydrogenic ensemble, whereas dashed lines mark CTMC calculations conducted with a micro-canonical ensemble.

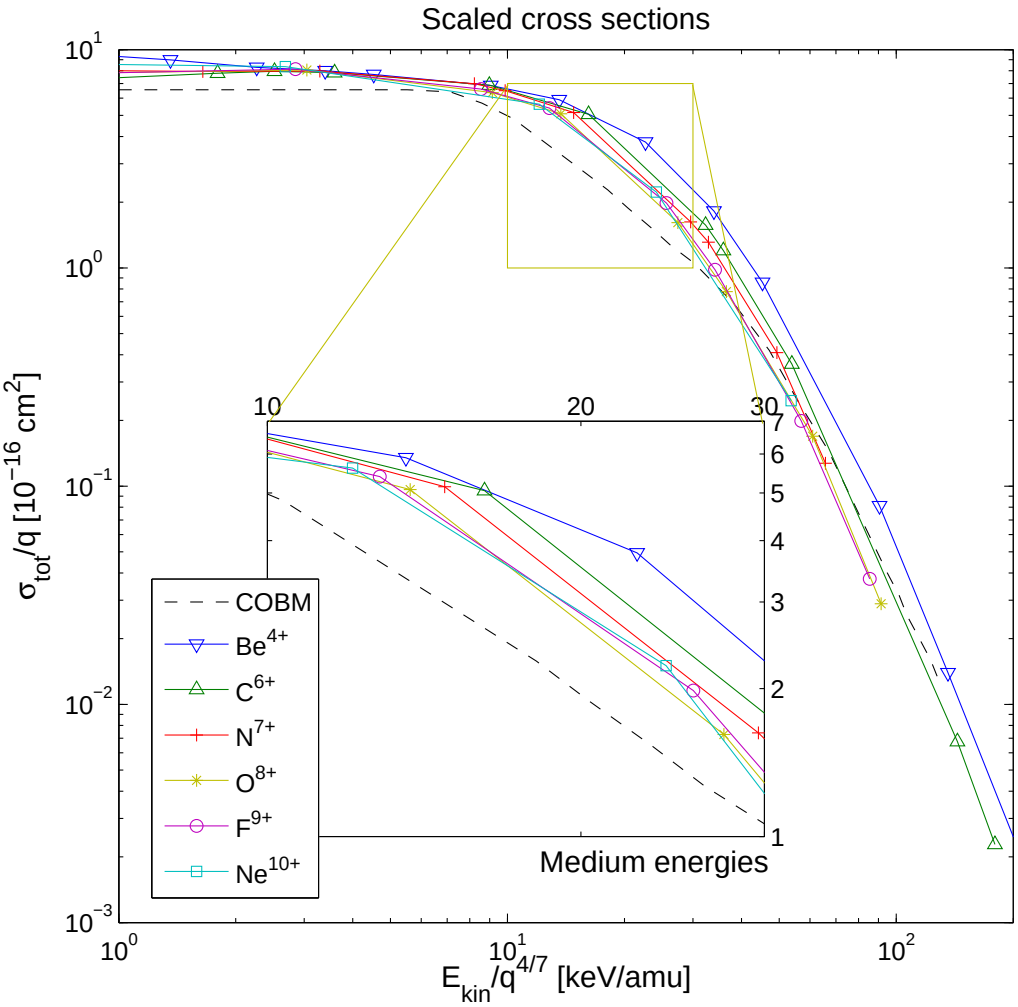


Figure 4.15: Calculated total charge transfer cross sections for collisions of neutral hydrogen with different fully stripped ions: beryllium (∇) [14], carbon (Δ) [14], nitrogen ($+$) [61], oxygen ($*$, see Figure 4.4), fluorine ($\circ$, see Figure 4.7) and neon ($\boxdot$, see Figure 4.10). The data was scaled with the well-known COBM scaling formula (see Figure 4.2b).

5 Conclusions and Outlook

Though being well advanced in years, the close-coupling method in its semi-classical impact parameter formulation proves a valuable tool in the treatment of single electron ion–atom collisions. We have shown that the results calculated with this method match both other classical calculations and experimental data very well and provide a reliable foundation for spectroscopy experiments in fusion research.

From a computational perspective, the use of symbolical structures like the program's underlying Shakeshaft representation is in a process of decline, because they require a considerable amount of book-keeping and careful circumnavigation of common numerical pitfalls. However, with our fast, reliable and easily accessible data structures, we minimised these difficulties and could demonstrate the speed and accuracy advantages of such techniques. Moreover, the improvements make the code shorter and more readable.

Such structures are interesting also in the field of quantum chemistry, where the use of Gaussian type orbitals drastically improved the scaling properties of multiple scattering centres and multiple active electrons. The introduction of symbolic structures would be highly beneficial both for the orbitals themselves and their angular coupling in highly dynamic systems.

Moreover, symbolic structures allow the extension to hybrid numeric-symbolical representations, which will bring us one step closer to the "holy grail" of scattering theory: an (up to machine accuracy) analytic expression for the scattering matrix, making it a highly rewarding subject for future research.

In the future, this may lead to the treatment of larger systems A^{q+}. For fusion, accurate cross sections for multiply charged tungsten W would be most interesting, but are far out of the scope of the present computation power.

A Supplementary calculations in Cartesian coordinates

As the Shakeshaft representation of the exchange integrals are strongly bound to Cartesian coordinates, we expand wave functions in the following way:

$$\langle \vec{r} | \psi \rangle = \exp(-\alpha r) \sum_i c_i r^{\rho_i} x^{\xi_i} y^{\eta_i} z^{\zeta_i} \tag{A.1}$$

This chapter first shows how spherical harmonics and Laguerre polynomials can be rewritten into terms of this form, effectively showing that Laguerre-type states and Slater type orbitals are compatible with the calculations.

Finally, we need to perform common calculus like inner products and derivatives in this representation. To clarify and shorten the calculations, we just show this for one term of the form:

$$T_{\rho\xi\eta\zeta} = \exp(-\alpha r)\, r^{\rho} x^{\xi} y^{\eta} z^{\zeta} \tag{A.2}$$

and define

$$l := \xi + \eta + \zeta \tag{A.3}$$

, which is consistent with the Cartesian expansion of the spherical harmonics Y_l^m (see Section A.1).

A.1 Spherical harmonics in Cartesian coordinates

We start with the general representation of the spherical harmonics [19]:

$$Y_l^m = \sqrt{\frac{(2l+1)}{4\pi} \frac{(l-m)!}{(l+m)!}} P_l^m(\cos\theta) \mathrm{e}^{\imath m\varphi}$$

where we chose the common normalization yielding an orthonormal system on the S^2:

$$\langle Y_l^m | Y_{l'}^{m'} \rangle = \int_\Omega (Y_l^m)^* Y_{l'}^{m'} d\Omega = \delta_{l,l'} \delta_{m,m'}$$

The associated Legendre polynomials P_l^m have the explicit form (including the Condon-Shortley phase $(-1)^m$):

$$P_l^m(x) = \begin{cases} (-1)^m \sqrt{1-x^2}^m \frac{\partial^m}{\partial x^m} P_l(x) & m \geq 0 \\ (-1)^m \frac{(l+m)!}{(l-m)!} P_l^{-m} & m < 0 \end{cases}$$

where P_l is the Legendre polynomial of order l. Shifting to Cartesian coordinates

$$\begin{aligned} x &= r \sin\theta \cos\varphi \\ y &= r \sin\theta \sin\varphi \\ z &= r \cos\theta \end{aligned}$$

as well as observing that $\exp(im\varphi) = (\cos\varphi \pm \imath \sin\varphi)^{|m|}$ and $\sqrt{1-\cos^2\theta} = \sin\theta$, we get:

$$Y_l^m = \sqrt{\frac{(2l+1)}{4\pi} \frac{(l-|m|)!}{(l+|m|)!}} \frac{\partial^{|m|}}{(\partial z/r)^{|m|}} P_l(z/r) \cdot r^{-|m|} \begin{cases} (-1)^m (x+iy)^{|m|} & m \geq 0 \\ (x-iy)^{|m|} & m < 0 \end{cases}$$

Finally, to store the function in terms of desired form, we need to expand the binomial:

$$(x \pm iy)^m = \sum_{i=0}^{m} \binom{m}{k} x^k (\pm \imath y)^{m-k}$$

Real spherical harmonics For wave functions, the program uses the real form of the spherical harmonics Υ_l^m, which can be written as a superposition

$$\Upsilon_l^m = \frac{1}{\sqrt{2}} \left[Y_l^m + (Y_l^m)^* \right] \quad \text{and} \quad \Upsilon_l^{-m} = \frac{1}{\imath\sqrt{2}} \left[Y_l^m - (Y_l^{-m})^* \right]$$

and form another orthonormal system on the 2-sphere. Because of the superposition, the phase factor $\exp \imath m\varphi$ turns into trigonometric functions:

$$\Upsilon_l^m = \sqrt{\frac{(2l+1)}{4\pi} \frac{(l-m)!}{(l+m)!}} P_l^m(\cos\theta) \cdot \begin{cases} 1 & m = 0 \\ \sqrt{2} \cos m\varphi & m > 0 \\ \sqrt{2} \sin m\varphi & m < 0 \end{cases}$$

Using the expansions

$$\sin, \cos m\varphi = \sum_{k=0}^{m} \binom{m}{k} \sin, \cos\left((m-k)\frac{\pi}{2}\right) \cos^k\varphi \sin^{m-k}\varphi$$

we can again switch to Cartesian coordinates:

$$\sin^m\theta \sin,\cos m\varphi = r^{-m}\sum_{k=0}^{m}\binom{m}{k}\sin,\cos\left((m-k)\frac{\pi}{2}\right)x^k y^{m-k}$$

A.2 Inner product of Cartesian wave functions

It can easily be shown that the product of two Cartesian wave functions is again of the form (A.1). As a result, the computation of inner products

$$\langle\psi|\phi\rangle = \int d^3r\,\langle\psi|\vec{r}\rangle\langle\vec{r}|\phi\rangle = \sum_k \tilde{c}_k \int d^3r \exp(-\tilde{\alpha}r) r^{\rho_k} x^{\xi_k} y^{\eta_k} z^{\zeta_k}$$

reduces to integration over terms T_{qijk} (A.2).

To perform this integration, we separate radial and angular part for each term:

$$\int d^3r \exp(-\alpha r) r^\rho x^\xi y^\eta z^\zeta = \int_0^\infty dr\, r^{\rho+l+2}\exp(-\alpha r)\cdot\int_\Omega d\Omega\, r^{-\rho}x^\xi y^\eta z^\zeta$$

where we defined l as in (A.3). We get a condition for the convergence (normalizability) of the integral:

$$\rho + l \geq -2$$

which is fulfilled for all Laguerre-type states as well as all exchange orbitals. Since $\alpha > 0$, the result of the radial integral is well-known in terms of the Gamma function:

$$\int_0^\infty dr\, r^{\rho+l+2}\exp(-\alpha r) = \frac{\Gamma(\rho+l+3)}{\alpha^{\rho+l+3}} \tag{A.4}$$

The angular part is slightly trickier and is best solved by switching back to spherical coordinates:

$$\int_\Omega d\Omega\, r^{-\rho}x^\xi y^\eta z^\zeta = \int_0^\pi d\vartheta\, \cos^\zeta\vartheta \sin^{\xi+\eta+1}\vartheta \cdot \int_0^{2\pi} d\varphi \cos^\xi\varphi \sin^\eta\varphi$$

First, we observe the following relations for $m, n \in \mathbb{N}$, which help us speeding up our computation:

$$\begin{aligned}\int_0^{2\pi} d\varphi \cos^m\varphi \sin^n\varphi &= 0 \quad \text{iff } m \text{ or } n \text{ is odd}\\ \int_0^{\pi} d\vartheta \cos^m\vartheta \sin^n\vartheta &= 0 \quad \text{iff } m \text{ is odd}\end{aligned}$$

These follow from symmetry properties of the integral. As an important consequence of

this, we can reduce the integrals to the first quadrant:

$$\begin{aligned}\int_0^{2\pi} d\varphi \cos^m\varphi \sin^n\varphi &= 4\int_0^{\pi/2} d\varphi \cos^m\varphi \sin^n\varphi \quad \text{iff } m,n \text{ are even}\\ \int_0^{\pi} d\vartheta \cos^m\vartheta \sin^n\vartheta &= 2\int_0^{\pi/2} d\vartheta \cos^m\vartheta \sin^n\vartheta \quad \text{iff } m \text{ is even}\end{aligned}$$

Finally, we use the Beta function $B(x,y)$

$$B(x,y) := \int_0^1 dt\, t^{x-1}(1-t)^{y-1} = 2\int_0^{\pi/2} d\theta\ \sin^{2x-1}\theta \cos^{2y-1}\theta \qquad \Re x, \Re y > 0$$

to give an expression for the angular part:

$$\int_\Omega d\Omega\, r^{-\rho} x^\xi y^\eta z^\zeta = \begin{cases} 2B\left(\frac{\eta+1}{2}, \frac{\xi+1}{2}\right) B\left(\frac{\xi+\eta+2}{2}, \frac{\zeta+1}{2}\right) & \xi,\eta,\zeta \text{ are even} \\ 0 & \text{otherwise}\end{cases}$$

Using the relationship between Beta and Gamma function

$$B(x,y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$$

and defining $\tilde{q} := \frac{1}{2}(q+1)$ we can write the non-trivial case of previous formula in the compact form, which eases its computation:

$$\int_\Omega d\Omega\, r^{-\rho} x^\xi y^\eta z^\zeta = 2\frac{\Gamma(\tilde{\xi})\Gamma(\tilde{\eta})\Gamma(\tilde{\zeta})}{\Gamma(\tilde{\xi}+\tilde{\eta}+\tilde{\zeta})} \quad \text{iff } \xi,\eta,\zeta \text{ are even} \tag{A.5}$$

Remark A.1. The calculation of the angular part using (A.5) is valid for all wave functions expandable in Cartesian coordinates. Because of the symmetric nature of most angular parts, the straightforward numerical evaluation of the sum leads to cancellation errors. Where available, it is therefore better to use orthogonality relationships.

A.3 Laplace on wave function expansion

For building pseudo-states $|\psi_P\rangle = (H-E)|\psi\rangle$, we need to calculate the Laplacian of the our wave function $\nabla^2|\psi\rangle$ and therefore on terms of the form (A.2). Using

$$\nabla^2(fg) = \nabla^2 f g + 2\nabla f \cdot \nabla g + f\nabla^2 g$$

and identifying $f = \exp(-\alpha r)r^\rho$ and $g = x^\xi y^\eta z^\zeta$, we can verify by elemental algebra that

$$\begin{aligned}
\nabla \exp(-\alpha r)r^\rho &= \hat{e}_r \frac{\partial}{\partial r}\left(\exp(-\alpha r)r^\rho\right) = \hat{e}_r \exp(-\alpha r)\left(-\alpha r^\rho + \rho r^{\rho-1}\right) \\
\hat{e}_r \nabla \left(x^\xi y^\eta z^\zeta\right) &= (\xi+\eta+\zeta) r^{-1} x^\xi y^\eta z^\zeta \\
\nabla^2 \left(\exp(-\alpha r)r^\rho\right) &= \exp(-\alpha r)\left(\alpha^2 r^\rho - 2\alpha(\rho+1)r^{\rho-1} + \rho(\rho+1)r^{\rho-2}\right) \\
\nabla^2 \left(x^\xi y^\eta z^\zeta\right) &= \xi(\xi-1)x^{\xi-2}y^\eta z^\zeta + \eta(\eta-1)x^\xi y^{\eta-2} z^\zeta + \zeta(\zeta-1)x^\xi y^\eta z^{\zeta-2}
\end{aligned}$$

where we used that $\hat{e}_m \hat{e}_r = x_m/r$. Using the compact notation for terms $T_{\rho\xi\eta\zeta}$ (A.2), we can easily write down the complete result:

$$\begin{aligned}
\nabla^2 T_{\rho\xi\eta\zeta} &= \alpha^2 T_{\rho\xi\eta\zeta} - 2\alpha(\rho+1+l)T_{(\rho-1)\xi\eta\zeta} + \rho(\rho+1+2l)T_{(\rho-2)\xi\eta\zeta} + \\
&\quad + \xi(\xi-1)T_{\rho(\xi-2)\eta\zeta} + \eta(\eta-1)T_{\rho\xi(\eta-2)\zeta} + \zeta(\zeta-1)T_{\rho\xi\eta(\zeta-2)}
\end{aligned}$$

B API specification of sic3ma

B.1 Polynomial

B.1.1 API overview

Polynomial represents a complex polynomial in n variables.

Although internally represented differently, it behaves like an array of terms, identified by the powers of the variables. The following example polynomial P in three variables x, y, z

$$P(x, y, z) = 0xy^3z^5 + 0.3x^8y^2(z^2 + z^3) - 0.6x^5y^3z^1$$

is represented by the following table:

powers of x	powers of y	powers of z	Coefficient
8	2	3	0.3
8	2	2	0.3
5	3	1	-0.6
1	3	5	0.0

Note how the terms are automatically ordered by the powers of their variables in descending order (*inverse lexicographic order*). To model our example polynomial, we can write:

```
program polytest
 use type_polynomial
 type(polynomial) :: poly

 poly = polycreate(3)       ! creates polynomial in three variables
 call polyaddterm(poly, (/ 1, 3, 5 /), cmplx(0.0, kind=polycfkind))
 call polyaddterm(poly, (/ 8, 2, 2 /), cmplx(0.3, kind=polycfkind))
 call polyaddterm(poly, (/ 8, 2, 3 /), cmplx(0.3, kind=polycfkind))
 call polyaddterm(poly, (/ 5, 3, 1 /), cmplx(0.6, kind=polycfkind))
 call polydump(poly, 6, (/ ' x', ' y', ' z' /))
 call polydestroy(poly)     ! destroys polynomial and frees memory
end program
```

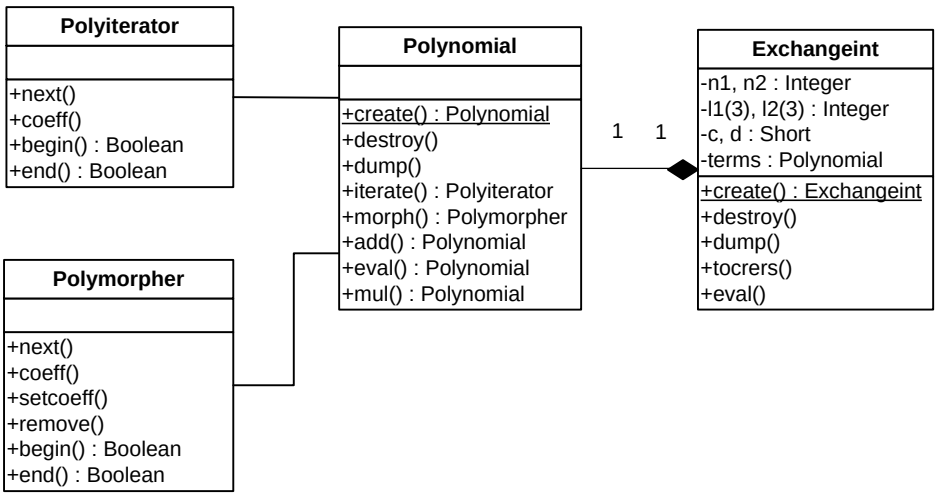


Figure B.1: Structure of the program in an UML class diagram [69]. Each block represents a static unit ("class"), where fields are given in the second and methods are given in the third section.

Note that due to its internal structure, every instance of `type(polynomial)` must be created before use and destroyed after use to avoid memory leaks. Also, to copy a polynomial, use `polyclone()` instead of an assignment (=), which just creates a shallow copy.

The structure of the polynomial class is displayed in Figure B.1.

Retrieving terms Terms are stored in a more complex manner than the array storage and cannot be indexed directly. Instead, the polynomial API follows the well-known *iterator pattern* for retrieving terms of the polynomial.

An iterator, represented by type `polyiterator`, is basically a pointer on the current term in a polynomial. It is initialized at the first term of the polynomial by calling `polyiterate`. Subsequent calls to `polynext` return the current term and move the pointer to the next term.

Thus, an iterator allows you to read the polynomial on a term-by-term basis, starting with the first term. For example, if we want to print all non-zero terms in a polynomial, we can extend our program to:

```
type(polyiterator) :: iter
integer :: powers(3)       ! array for powers of variables
complex(kind=polycfkind) :: coeff
   ...
iter = polyiterate(poly)  ! sets iterator before the first term

! fill powers with variable powers and coeff with the coefficient of the
```

```
! current term, and set the pointer before the next term. If no such
! term exists, iternext returns false and the loop terminates
do while(iternext(iter, powers, coeff))
    if (coeff .gt. 0) &
          & write (*,*) 'term:', powers, ' coeff=', coeff
end do
```

Please note that if you are looking for a specific term, you must iterate over the polynomial until you find it. In common use cases, this does not impeach polynomial performance, as we do not need specific terms.

Modifying and removing terms Polynomial provides another type of iterator: `polymorpher`. This iterator performs more poorly than `polyiterator`, but it allows us to remove a term from the polynomial by issuing `morphremove`.

For example, the following subroutine removes all zero terms from our polynomial:

```
subroutine removezeros(poly)
   type(polynomial), intent(inout) :: poly
   type(polymorpher) :: it

   it = polymorph(poly)         ! sets the morpher before first term
   do while(morphnext(it))      ! get next term
      if (morphcoeff(it) .eq. cmplx(0.0, kind=polycfkind)) then
         call morphremove(it)  ! remove current term if coefficient is zero
      end if
   end do
end subroutine
```

As this is a fairly common task, polynomial actually provides the similar subroutine `polycompress` for it. Please note that `morphremove` does not automatically move the iterator to the next element. Therefore, we must call `morphnext` before we can edit or remove the next element.

Serializing If we want to store our polynomial or send it via MPI, we must transfer our complex internal tree structure into a well-defined sequence of elements. We call this process *serialization* and its inverse — restoring the polynomial from the list — *deserialization*.

One way to serialize our polynomial would be to convert it to the table of term powers and coefficients. Polynomial, however, provides a faster and more compact way to serialize and deserialize itself: `polyserialize` and `polydeserialize`. These func-

tions operate on an array of type `polyserialterm`, which is the serialized form of the polynomial[1]:

```
type polyserialterm  ! element of the serialized form of polynomial
   logical :: separator = .false.
   integer*2 :: power = 0
   complex*8 :: coeff = pczero
end type polyserialterm
```

When broadcasting polynomials via MPI, it is convenient to create a MPI derived type to model `polyserialterm`:

```
subroutine polysertype(mpitypeid, ierror)
   use mpi
   integer, intent(out)   :: mpitypeid
   integer, intent(inout) :: ierror
   integer :: size_logical, size_integer2, size_complex8
   integer :: str_offsets(0:2), str_types(0:2), str_counts(0:2)

   ! determine size of the types, and fill elements of the custom type
   call mpi_type_extent(mpi_logical,  size_logical,  ierror)
   call mpi_type_extent(mpi_integer2, size_integer2, ierror)
   call mpi_type_extent(mpi_complex8, size_complex8, ierror)

   ! fill elements, types and positions into the type
   str_counts  = (/ 1, 1, 1 /)
   str_offsets = (/ 0, size_logical, size_logical+size_integer2 /)
   str_types   = (/ mpi_logical, mpi_integer2, mpi_complex8 /)

   ! create and commit the type, returning its ID
   call mpi_type_struct(3, str_counts, str_offsets, str_types, &
                        & mpitypeid, ierror)
   call mpi_type_commit(mpitypeid, ierror)
end subroutine
```

This allows us to communicate the polynomial to another process by directly sending the array we get from the serialization process:

```
integer, parameter :: nmax = 100
```

[1] Actually, this is only half of the serialization process, since number representations vary on most architectures (most prominently, little- and big-endian numbers). This is taken care of by MPI, when storing these terms into a binary file, however, you need to take care of a defined number representation.

```
type(polynomial) :: poly, dpoly
type(polyserialterm) :: elem(nmax)
integer :: n, sertypeid, ierror
   ...          ! mpi initialization here
! create type, serialize polynomial, broadcast its terms
call polysertype(sertypeid, ierror)
call polyserialize(poly, nmax, n, elem)
call mpi_bcast(elem, n, sertypeid, 0, mpi_comm_world, ierror)
   ...
dpoly = polydeserialize(n, elem)    ! restore polynomial
```

Serializing the polynomial is also a fast and space-saving way of storing it in a file.

Evaluation and calculus Polynomials, or parts thereof, can be evaluated using `polyeval`. `polyeval` returns a polynomial, where the variables that you provide values for have been evaluated. So if you evaluate m variables of a n-variables polynomial, you get a polynomial in $n - m$ variables. Consequently, inserting values for all unknowns yields a *constant polynomial* (a polynomial in zero variables). You can retrieve the value by combining the functions with `polyvalue`:

```
type(polynomial) :: poly
complex(kind=polykind), parameter :: vals(3) = &
       & (/ 3, -5, 2 /)*pcone      ! pcone is 1 of required type
  ...
value = polyvalue(polyeval(poly, values=vals))
```

The polynomial is collapsed effectively and in a numerically stably fashion using the Horner scheme provided that its is evaluated from the last variable forwards. Therefore, if you evaluate a polynomial in steps, you should choose the variable order inversely to the order of evaluation.

To add or multiply two polynomials, use `polyadd` and `polymul`, to add or multiply with constant values, use `polyaddconst` and `polyscale`. These functions do not modify their operands, but rather return the result of the computation. Remember that you need to destroy these results (even if they are only temporary results!). If we just want to operate on one polynomial, it is much faster and safer to use `polyaddto`, `polyaddconstto` and `polyscaleto` (compare Table B.1).

B.1.2 Implementation notes

Polynomial was not implemented with arrays, but using a binary tree structure (see Section 3.2 for a detailled description). An important consequence of this is that you need to explicitly `polydetroy` every single instance of polynomial (even temporary instances within a calculation) to avoid memory leaks.

	$p \diamondsuit q$	$p = p \diamondsuit q$	$p \diamondsuit c$	$p = p \diamondsuit c$
Addition +	`polyadd`	`polyaddto`	`polyaddconst`	`polyaddconstto`
Multiplication ×	`polymul`	–	`polyscale`	`polyscaleto`

Table B.1: Operations ($\diamondsuit$) on polynomials, where p and q are polynomials and c is a constant.

B.1.3 Methods summary

Method	Description
Polynomial methods (poly...)	
add	Adds two polynomials and returns the result without modifying the summands (use `polyaddto` if you want to add a polynomial to another). Remember to destroy the result after use.
`addconst`	Adds a polynomial and a complex scalar, returning the result without modifying the summand (use `polyaddconstto` if you want to add a constant to a polynomial).
`addconstto`	Adds a complex scalar to a polynomial.
`addtcb`	Adds a term to a polynomial, adding coefficients using a callback function. This function is identical to `polyaddterm`, except that it allows you to specify a function handling the actual summation of the coefficients.
`addterm`	Adds a term to polynomial, identified by the power of variables and its coefficient. If such a term already exists, then the coefficients are added.
`addto`	Adds a polynomial to another.
`addvars`	Adds a number of variables to the polynomial at the location specified (remember that polynomial distiguishes variables by their position).
`alive`	Checks whether a given polynomial has been initialized, i. e. if the polynomial is in a state after its creation with `polycreate` and before its destruction with `polydestroy`. As best practise, you should include this into sanity checks of your methods.
`clear`	Removes all terms from the polynomial and sets it to zero. Please note that this is no substitute for `polydestroy`.

Method	Description
`clone`	Returns a copy of the polynomial. This is essential because simple assignment just create a *shallow copy*, i. e., another pointer to the same data.
`compress`	Removes entries from the polynomial which are zero or near-zero
`conjg`	Returns the conjugate complex of the polynomial, that is a copy of the polynomial where all coefficients are replaced by their conjugate complex. Note that you need to conjungate the variables yourself when evaluating.
`count`	Returns the number terms with a specific coefficient
`create`	Creates and returns a new polynomial without any terms (value zero).
`debugdiff`	Checks two polynomials for significant differences, i. e., terms that are present in one but not in the other polynomial or terms that differ in their coefficient.
`debugtree`	Dumps the internal tree representation of the polynomial (use `polydump` for more human-readable output).
`delinearize`	Restores the polynomial from its linearised form (see `lpolynomial` and `linearize`)
`deserialize`	*Deprecated* (see `polyserialize`).
`destroy`	Removes all terms from the polynomial and frees the memory. Please not that you must call this function for every polynomial created.
`dump`	Writes the powers and coefficient of all or some of the terms of a polynomial to some file or output stream.
`eval`	Returns a (partly) evaluated copy the polynomial (see above).
`isconstant`	Checks whether the polynomial has no terms and represents just a scalar value
`iterate`	Returns a non-modifying iterator over the terms of the polynomial, which allows for term-by-term processing of the polynomial.
`linearize`	Generates a space-saving, but immutable form of the polynomial (see `lpolynomial`).
`morph`	Returns a modifying iterator over the terms of the polynomial, which behaves like `iterate`, but allow the removal of terms through `morphremove`.

Method	Description
`mul`	Multiplies two polynomials and returns the result without modifying the operands. If you just multiply with one term (monomial), use `polyshift` instead, if you are multiplying with a scalar, use `polyscale`.
`nbytes`	Estimates the memory demand of the polynomial
`nterms`	Returns the number of terms in the polynomial
`nvars`	Returns the number of variables (rank) of the polynomial
`scale`	Multiplies a polynomial and a complex scalar and returns the scaled polynomial as a result without modifying the operands (to scale a polynomial explicitly use `polyscaleto`).
`scaleto`	Scales a polynomial by a complex factor.
`serialize`	*Deprecated* (use `linearize` instead). Created a linearized structure, which is now superseded by `lpolynomial`.
`setvalue`	If the polynomial is a constant (`polyisconstant`), then set the constant scalar value.
`shift`	Multiplies the polynomial and a single term (monomial), returning the result without modifying the operand.
`value`	If the polynomial is a constant (`polyisconstant`), then returns the constant scalar value.
Non-modifying iteration methods (iter...)	
`begin`	Returns whether the iterator is at the beginning, i. e., before the first term.
`coeff`	Gets the coefficient of the current term
`end`	Returns whether the iterator is at the end, i. e., after the last term.
`next`	Advances the iterator to the next term.
Modifying iteration methods (morph...)	
`begin`	Returns whether the iterator is at the beginning, i. e., before the first term.
`coeff`	Gets the coefficient of the current term
`end`	Returns whether the iterator is at the end, i. e., after the last term.

Method	Description
`next`	Advances the iterator to the next term.
`remove`	Removes the current term
`setcoeff`	Sets the coefficient of the current term

B.2 Linear-layout polynomial

B.2.1 API overview

`lpolynomial` represents an complex polynomial in n variables, which is essentially equivalent to a `polynomial`.

The linear-layout polynomial type provides a memory saving way to store a polynomial, which also evaluates faster is more robust against memory leaks. However, no mutations or arithmetic operations can be performed except the evaluation of the polynomial, making it a non-modifiable version of `polynomial`. One can switch between normal representation (`polynomial`) and linear representation (`lpolynomial`) using the `linearize, linearizeto` and `delinearize` methods of type `polynomial`.

As a result, you will typically create an instance of type polynomial first and perform all the symbolic manipulations on it. Finally you use `polylinearize` to get an instance of type lpolynomial, which you can store or evaluate:

```
type(polynomial) :: p
type(lpolynomial) :: lp

p = polycreate(3)       ! creates polynomial
...       ! do something (add terms, etc.)
lp = polylinearize(p)
call polydestroy(p)
...       ! evaluate or store
```

Note that `lpolynomials` should only be accessed by the respective functions and not by manipulating the arrays directly. However, the arrays are not strictly private to allow for easy serialization/deserialization and interoperability with the polynomial type. Unlike polynomials, you need not explicitly destroy a lpolynomial.

Retrieving terms The iteration mechanism differs slightly from the one one polynomials for internal and historical reasons. As with polynomial, you first use `lpolyiterate` to generate a constant iterator over a lpolynomial. You need not destroy an iterator once you're finished with it, just let it run out of scope.

After you generated an iterator, it is placed at the beginning (`literbegin` returns true). A subsequent call to `liternext` returns the *first* term, if present. Typically, you will use `liternext` inside a loop, using its return value to check for further terms. Note that you need to call liternext() *before* any other operation (such as *litercoeff*) on the term.

```
type(lpolyiterator) :: it
integer(lpolyint) :: powers(3) complex(lpolykind) :: coeff
...
it = lpolyiterate(poly)
do while(liternext(it, powers, coeff))  ! fills coeff and powers
   write (*,*) powers, coeff
end do
```

File or network transfer Linear-layout polynomials essentially consist of three arrays and a couple of indices and can therefore easily be stored on disk or transmitted via MPI or OpenMP.

Following code stores a linear-layout polynomial in a binary file:

```
type(lpolynomial) :: p
...
open(9, file='save.bin', access='stream', form='unformatted', &
        status='new', convert='little_endian')
write(9) int(p%nvars,1), int(p%nterms,4)
write(9) int(size(p%children),4), int(size(p%powers),4), &
          int(size(p%coeffs),4)
write(9) p%children(:), p%powers(:), p%coeffs(:)
close(9)
```

where the integer conversions and the endian conversions are used to ensure compatibility over different architectures. For all other operations, it is not recommended to use the arrays directly.

B.2.2 Implementation notes

The linear-layout polynomial class was implemented using three arrays (see Section 3.3 for details). Since allocatable structures are discarded when they go out of scope, you do not need to destroy lpolynomials explicitly.

Bibliography

[1] International Atomic Energy Agency (IAEA). *INES – The International Nuclear and Radiological Event Scale – User's Manual*. IAEA Vienna, 2008 edition, 2009.

[2] United States Geological Survey (USGS). Magnitude 9.0 – near the east coast of Honshu, Japan. accessed 22/3/2011.

[3] Japanese Atomic Industrial Forum (JAIF). Reactor Status and Major Events Update 25 – NPPs in Fukushima as of 21:00 March 22. accessed 22/3/2011.

[4] H. Aichinger. Sepp Linhart: "Sie wissen einfach nicht, was sie tun". *Der Standard*, 16 March 2011.

[5] E. A. Rosa, G. E. Machlis, and K. M. Keating. Energy and society. *Ann. Rev. Sociology*, 14:149–172, 1988.

[6] C. E. Housecroft and A. G. Sharpe. *Inorganic Chemistry*. Prentice Hall, 2nd edition, 2004. Nuclear binding energy (ch. 2.2).

[7] R. J. Fonck, D. S. Darrow, and K. P. Jaehnig. Determination of plasma-ion velocity distribution via charge-exchange recombination spectroscopy. *Phys. Rev. A*, 29(6):3288–3308, 1984.

[8] G. Federici, C.H. Skinner, J.N. Brooks, J.P. Coad, C. Grisolia, A.A. Haasz, A. Hassanein, V. Philipps, C.S. Pitcher, J. Roth, W.R. Wampler, and D.G. Whyte. Plasma-material interactions in current tokamaks and their implications for next step fusion reactors. *Nuclear Fusion*, 41(12R):1967–2137, 2001.

[9] D. R. Bates. A note on the exchange integrals in the impact-parameter treatment of heavy-particle collisions. *Proc. R. Soc.* A, 247(1250):294–301, 1958.

[10] B. H. Bransden and M. R. C. McDowell. *Charge exchange and the theory of ion–atom collisions*. Oxford University Press, 1st edition, 1992.

[11] W. Fritsch and C. D. Lin. The semiclassical close-coupling description of atomic collisions: Recent developments and results. *Physics Reports*, 202(1-2):1–97, 1991.

[12] J. Schweinzer. *Ein- und Zweielektronenprozesse beim Einelektroneneinfang von Alkaliatomen in doppelt geladene Edelgasionen [Single and double electron processes in charge exchanges between alkali atoms and doubly charged noble gas ions].* Dissertation, Technische Universität Wien – Vienna University of Technology, 1990.

[13] K. Igenbergs. *Cross section database for neutral sodium plasma edge diagnostics.* Diplomarbeit, Technische Universität Wien – Vienna University of Technology, 2007.

[14] K. Igenbergs, J. Schweinzer, and F. Aumayr. Charge exchange in $Be^{4+}-H(n=1,2)$ collisions studied systematically by atomic-orbital close-coupling calculations. *J. Phys.* B*: At. Mol. Opt. Phys.*, 42(23):235206, 2009.

[15] H. Friedrich. *Theoretical Atomic Physics.* Springer, 3rd edition, 2006.

[16] J. J. Sakurai. *Modern Quantum Mechanics.* Addison Wesley, revised edition, September 1993.

[17] J. Z. H. Zhang. *Theory and Application of Quantum Molecular Dynamics.* World Scientific, 1st edition, 1999. Jacobi Coordinates in triatomic systems (ch. 4.5.3).

[18] B. H. Brandsen and C. J. Joachain. *Physics of Atoms and Molecules.* Paerson education, 2nd edition, 2003.

[19] A. Messiah. *Quantum Mechanics (Physics).* Dover Publications, January 2000.

[20] L. Wilets and D. F. Gallaher. Coupled-state calculations of H^+ – H scattering. *Physics Review*, 147(1):13–20, 1966.

[21] T. A. Green. A proof of detailed balancing for the impact parameter method. *Proc. Phys. Soc.*, 86(5):1017–1029, 1965.

[22] M. Sozzi. *Discrete Symmetries and CP Violation: From Experiment to Theory.* Oxford University Press, 1st edition, 2008. Symmetries and invariances (ch. 1), Time reversal symmetry (ch. 4.3), Scattering (app. C.1).

[23] Keith Hutchison. Is classical mechanics really time-reversible and deterministic? *Brit. J. Phil. Sci.*, 44(2):307–323, 1993.

[24] Steven F. Savitt. Is classical mechanics time reversal invariant? *Brit. J. Phil. Sci.*, 45(3):907–913, 1994.

[25] Craig Callender. The metaphysics of time reversal: Hutchison on classical mechanics. *Brit. J. Phil. Sci.*, 46(3):331–340, 1995.

[26] F. Aumayr. *Vorlesung Atomare Stoßprozesse [Atomic collision processes]*. Lecture notes, 2004/05. Klassische Streufunktion (ch. 2.5).

[27] E. P. Wigner. Normal form of antiunitary operators. *J. Math. Phys.*, 1(5):409–413, 1960.

[28] G. E. Andrews, R. Askey, and R. Roy. *Special Functions*. Cambridge University Press, 1st edition, 2001. Completeness of Orthogonal Polynomials (ch. 6.5).

[29] I. M. Cheshire, D. F. Gallaher, and A. J. Taylor. Coupled-state calculations of proton-hydrogen scattering with a pseudo-state expansion. *J. Phys.* B*: At. Mol. Opt. Phys.*, 3(6):813, 1970.

[30] D. G. M. Anderson, M. J. Antal, and M. B. McElroy. The triple centre approximation for charge exchange in atomic scattering theory. *J. Phys.* B*: At. Mol. Opt. Phys.*, 7(4):L118 – L121, 1974.

[31] W. Fritsch and C. D. Lin. Close-coupling calculations for inelastic processes in intermediate energy ion-atom collisions. *J. Phys.* B*: At. Mol. Opt. Phys.*, 15(8):1255, 1982.

[32] G. H. Golub and C. F. van Loan. *Matrix computations*. Johns Hopkins University Press, 3rd edition, 1996.

[33] Alan J. Laub. *Matrix Analysis for Scientists and Engineers*. Society for Industrial Mathematics, 2004.

[34] N. Sisourat, I. Pilskog, and A. Dubois. Non-perturbative treatment of multi-electron processes in ion-molecule scattering: application to $He^{2+} - H$ collisions. *Submitted*, July 2011.

[35] Wilhelm Magnus. On the exponential solution of differential equations for a linear operator. *Commun. Pure Appl. Math.*, 7(4):649–673, 1954.

[36] S. Blanes, F. Casas, J.A. Oteo, and J. Ros. The magnus expansion and some of its applications. *Physics Reports*, 470(5-6):151–238, 2009.

[37] H. Ryufuku and T. Watanabe. Charge transfer in collisions of atomic hydrogen with O^{8+}, He^{2+}, and H^{+}. *Phys. Rev. A*, 18(5):2005–2015, 1978.

[38] R. Abrines and I. C. Percival. Classical theory of charge transfer and ionization of hydrogen atoms by protons. *Proc. Phys. Soc.*, 88:861–872, 1966.

[39] S. S. Sawilowsky. You think you've got trivials? *J. Mod. Appl. Stat. Meth.*, 2(1):218–225, 2003.

[40] Nobuyuki Toshima. Classical-trajectory Monte Carlo calculations for energetic electron-capture processes. *Phys. Rev. A*, 42(9):5739–5742, 1990.

[41] L. Kocbach and R. Liska. Closed form formula for the exchange integrals in the impact-parameter treatment of heavy-particle collisions. *J. Phys.* B*: At. Mol. Opt. Phys.*, 27(18):L619–L624, 1994.

[42] R. Shakeshaft. A note on the exchange integrals in the impact-parameter treatment of heavy-particle collisions. *J. Phys.* B*: At. Mol. Opt. Phys.*, 8(8):L134–L136, 1975.

[43] J.-P. Hansen. General subroutines for the calculation of atomic and molecular two-centre integrals (ALAIN). *Comput. Phys. Commun.*, 58(1-2):217–221, 1990.

[44] J.-P. Hansen and A. Dubois. Procedures for analytical and numerical calculation of Coulombic one- and two-centre integrals (JANAL). *Comput. Phys. Commun.*, 67(3):456 – 464, 1992.

[45] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in Fortran 77: The Art of Scientific Computing.* Cambridge University Press, 2nd edition, September 1992.

[46] H. Hulett. Monomials and the lexicographic order. In S. K. Jain and Rizvi S. Tariq, editors, *Advances in Ring Theory*, pages 145–150. Birkhäuser Boston, 1st edition, 1997.

[47] Donald E. Knuth. *The Art of Computer Programming, Volume 1: Fundamental Algorithms.* Addison-Wesley, 3rd edition, 1997. Binary Tree Representations (ch. 2.3.2), Preorder Sequential Representation (ch. 2.3.3).

[48] ISO/IEC 1539-1:2004(E). *Information technology – Programming languages – Fortran – Part 1: Base language.* ANSI, 2004. Informally known as "Fortran 2003".

[49] ISO/IEC 1539-1:2010. *Information technology – Programming languages – Fortran – Part 1: Base language.* Final Draft International Standard, 2010. Informally known as "Fortran 2008".

[50] M. Wallerberger, K. Igenbergs, J. Schweinzer, and F. Aumayr. Fast computation of close-coupling exchange integrals using polynomials in a tree representation. *Comput. Phys. Commun.*, 182(3):775–778, 2011.

[51] K. Pearson. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *Philosoph. Mag. Ser.* V, 50:157–175.

[52] J. Burgdörfer, R. Morgenstern, and A. Niehaus. Angular momentum distribution in the classical over-barrier model for electron capture into highly charged slow projectiles. *J. Phys.* B*: At. Mol. Opt. Phys.*, 19(14):L507–L513, 1986.

[53] A. Maier. *Calculation of the exchange integral with a closed-form fomula for the integrand.* Projektarbeit, Technische Universität Wien – Vienna University of Technology, 2011.

[54] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems.* December 2009.

[55] C. Runge. Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten [On empirically known functions and interpolation between equidistant ordinates]. *Zeitschrift für Mathematik und Physik*, 46:224–243, 1901.

[56] R. C. Isler. An overview of charge-exchange spectroscopy as a plasma diagnostic. *Plasma Physics and Controlled Fusion*, 36:171–208, 1994.

[57] M. Merola, D. Loesser, A. Martin, et al. ITER plasma-facing components. *Fusion Engineering and Design*, 85:2312–2322, 2010. Proceedings of the Ninth International Symposium on Fusion Nuclear Technology.

[58] A. W. Kleyn, N. J. Lopes Cardozo, and U. Samm. Plasma-Surface interaction in the context of ITER. *Phys. Chem. Chem. Phys.*, 8:1761–1774, 2006.

[59] D. G. Whyte, T. C. Jernigan, D. A. Humphreys, et al. Mitigation of tokamak disruptions using high-pressure gas injection. *Phys. Rev. Lett.*, 89:055001–055005, 2002.

[60] K. Igenbergs. *Calculation of Cross Sections Relevant for Diagnostics of Hot Fusion Plasmas.* Dissertation, Technische Universität Wien – Vienna University of Technology, 2011.

[61] K. Igenbergs, J. Schweinzer, A. Veiter, L. Perneczky, E. Frühwirth, M. Wallerberger, R. E. Olson, and F. Aumayr. Charge exchange and ionisation in N^{7+}, N^{6+}, C^{6+} - $H(n = 1, 2)$ collisions studied systematically by theoretical approaches. *J. Phys.* B*: At. Mol. Opt. Phys.*, 2011. submitted.

[62] H. Knudsen, H. K. Haugen, and P. Hvelplund. Single-electron-capture cross section for medium- and high-velocity, highly charged ions colliding with atoms. *Phys. Rev. A*, 23(2):597–610, 1981.

[63] R. E. Olson and D. R. Schultz. n, l distributions for electron capture from H($1s$) by C^{6+} and O^{8+}. *Physica Scripta*, 1989(T28):71, 1989.

[64] C. Illescas and A. Riera. Classical study of single-electron capture and ionization processes in A^{q+} – (H, H_2) collisions. *Phys. Rev. A*, 60:4546–4560, 1999.

[65] M. Bendahman, S. Bliman, S. Dousson, D. Hitz, R. ayet, J. Hanssen, C. Harel, and A. Salin. Electron capture from atomic hydrogen by multiply charged ions in low energy collisions. *J. Physique*, 46:561–572, 1985.

[66] F. W. Meyer, A. M. Howald, C. C. Havener, and R. A. Phaneuf. Low-energy total-electron-capture cross sections for fully stripped and H-like projectiles incident on H and H_2. *Phys. Rev. A*, 32(6):3310–3318, 1985.

[67] L. F. Errea, C. Illescas, L. Mendez, B. Pons, A. Riera, and J. Suare. Classical and semiclassical treatments of highly charged ions + H($1s$) collisions. *Nucl. Instr. Meth.* B, 235:315–320, 2005.

[68] A. Schmidt, M. Horbatsch, and R. M. Dreizsler. Semiclassical phase space description of ionisation and capture for ions colliding with hydrogen-like targets. *J. Phys.* B*: At. Mol. Opt. Phys.*, 23:2327S–2340S, 1990.

[69] Object Management Group. *Unified Modelling Language 2.0: Superstructure*. online: http://www.omg.org/spec/UML/2.0/Superstructure/PDF, 8 2005.

List of Figures

List of Tables

Index

Acknowledgements

First and foremost, I want to thank my adviser, Professor Friedrich Aumayr for his constant support and guidance throughout my work. Aside from a permanent working place and equipment, he also gave me the great opportunity to attend an international conference and present my work there. He also guided me through the publication process in a journal. His deep physical insights and his kind character made this work highly enjoyable.

I also had the pleasure of working with Dipl-Ing. Katharina Igenbergs, whose continued advice had a great impact on my work. In the course of my thesis, her bright mind and her bright nature were dear companions and I very much enjoyed the stay with her at the conference in Belfast.

I also want to thank Professor Josef Schweinzer for valuable discussions during my stay in Garching and also for collaborating in the publications. Moreover, I want to thank other members of the working group, which I increasingly perceived as a scientific family (and the excellent coffee machine, which brought us together many times).

Last but certainly not least, I would like to thank my girlfriend Kathi for enduring me and unconditionally supporting me the whole time. This work is dedicated to my father and my brother, whose support throughout made this work and my studies possible.