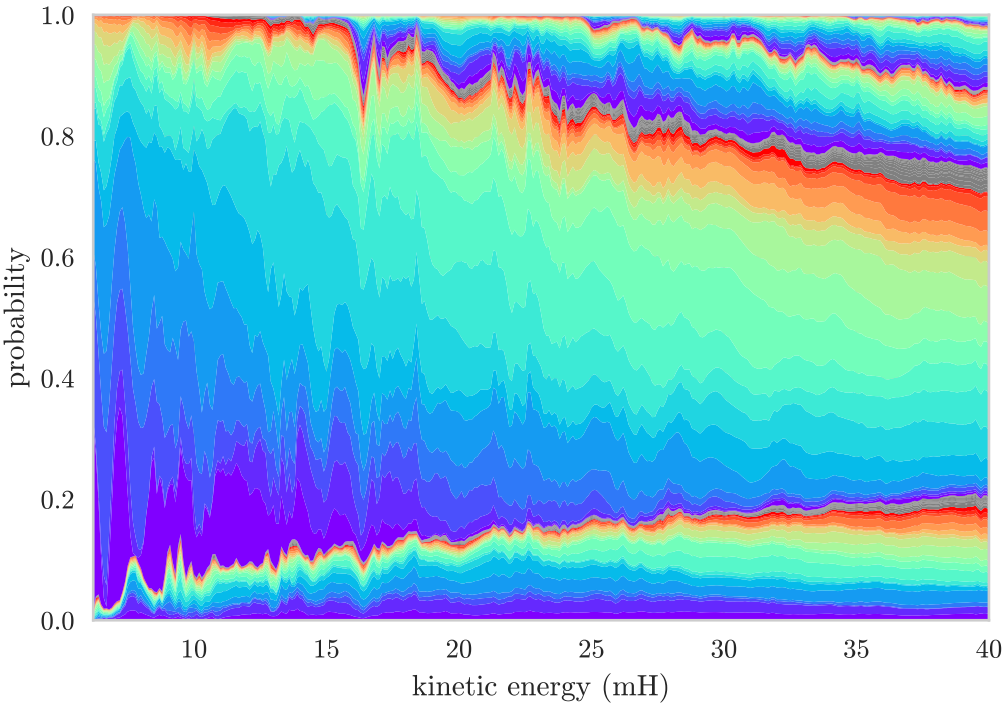
Ab initio time-dependent inelastic scattering of H-CO

Mark Verleg
Theoretical Chemistry, Radboud University Nijmegen



Abstract

Inelastic transitions during scattering between hydrogen (H) and carbon monoxide (CO) are studied. Quantifying these transitions is important in astronomy for understanding the interstellar medium from spectrum data. A time-dependent quantum mechanical wavepacket approach is used to obtain *ab initio* state-to-state results for this reaction at intermediate and higher energy, while including almost all angular momentum projection states. Cross sections agree well with those recently computed by Song *et al.*[1]. This work lays the foundation to extend these to higher excitations and kinetic energies.

Contents

1 Introduction

1.1 H-CO

This work studies the interaction between hydrogen (H) and carbon monoxide (CO). It looks at how the particles change when they hit each other - how energy is converted between kinetic, rotational and vibrational energy for many different collisions. It does not look at chemical reactions (rearranging of bonds) or at electronic transitions.

1.2 Application

The interaction of hydrogen atoms (H) with carbon monoxide molecules (CO) is important in astronomy, because hydrogen is the most common atom and carbon monoxide is the second most common molecule in the interstellar medium (see figure 1). Besides that, interactions between the two are also common in atmospheric processes and during combustion of organic matter.

Understanding the rate of reaction from and to different rovibrational states helps understand the composition of a gas from the light emitted and absorbed by its transitions. CO has various absorption and transmission lines in different ranges of the electromagnetic spectrum, even at low temperatures. Because the regions where this reaction happens in space are often not in local thermal equilibrium, detailed information on these transitions is needed.

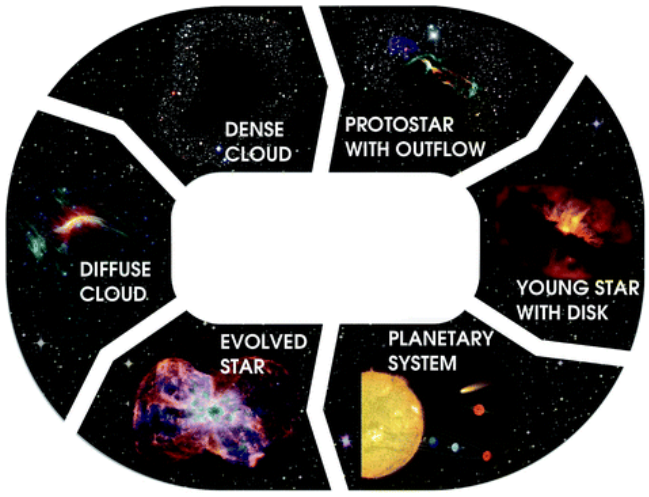


Figure 1: A visualization describing the processes in the interstellar medium: the formation of stars from clouds of gas, and the clouds of gas produced by the eventual demise of the stars. The understanding of these processes requires detailed information about molecular interactions such as the one discussed in this work. Image from van Dishoeck [2].

1.3 Earlier work

Theorists[3–12] and experimentalists[13–16] have studied scattering events involving H-CO including photo-dissociation and rotational and vibrational transitions.

There are two reasons that calculations on H-CO are relatively difficult. First, the interaction between H and CO is strong compared to other partners of CO like He and H_2, which is evident from the deep minimum in the interaction potential (30.7 mE_H). Second, there is an energy barrier to reach this minimum or to dissociate from it (5.18 mE_H). These lead to slow reactions and strong non-classical effects, and many basis functions are required to describe a wavefunction with high enough energy during interaction.

Until several years ago, experimental collision de-excitation rate coefficients were available for H-CO only for vibrational transitions without any rotational information. Recently, Song *et al.* [17] performed calculations that extended the theoretical results beyond $\nu = 1 \rightarrow 0$, using a novel extrapolation method[18]. Even more recent work by [16] used MCTDH to study H-CO transitions, studying the effect of Renner-Teller coupling in particular.

A detailed history of the existing work on H-CO scattering is available in [19], and will not be repeated here.

1.4 Goal

The aim of this work is to calculate state-to-state cross sections for inelastic H-CO scattering, which may be used for astronomy purposes, see section 1.2 (Application). Specifically, the aim is first to obtain results matching those of Song *et al.* in the intermediate energy region. And second, to extend results to higher vibrational and kinetic energies.

High j states are also important for astronomy, as high as 90. This work will not reach that high, since high rotational states necessitate a large number of Ω quantum states for accurate description, which rapidly increases the computational cost. Therefore results were only calculated starting from the first 8 initial rotational levels (the basis consisted of more).

2 Methods

2.1 Quantum mechanics

Classically, the addition of a particle to a molecule might require a set of three positions and three momenta for computer representation of an unconstrained system in three-dimensional space. That means that N particles might need $6N$ numbers without[1] discretization: linear ($\mathcal{O}(N)$) space complexity.

In a quantum mechanical treatment, the addition of a particle extends the wavefunction by three dimensions. This happens because the wavefunction for particle n at position $\boldsymbol{x}_n$ depends on every other dimension and every other particle. This is essentially the Cartesian product of every degree of freedom: the existing wavefunction now needs a value at each combination of $x_{n,1}, x_{n,2}, x_{n,3}$. This changes the problem from linear into exponential $\mathcal{O}\left(D^N\right)$ space complexity, with D the grid size per dimension.

This is what makes full quantum mechanical calculations very computationally expensive. A number of approximations and computational optimizations are used in this work to make it tractable. Using these, treatment of a three-atom

[1] Beyond the discretization that is implicit when storing reals in a computer.

system has been possible since before the turn of the century[20], but remain time-consuming to this day.

There are other differences as well:

1. Classically, many trajectories are computed and averaged, whereas in quantum mechanics many paths can and must be sampled at once, because they interfere with each other.

2. Constrained motion, such as angular momentum, is quantized. For example in body-fixed coordinates, the angular momenta of CO, the total angular momentum and their projection onto the H-CO axis are fixed by the positive integer quantum numbers j, J, and Ω respectively.

3. When the kinetic energy is lower than the energy of a barrier, that is no longer a guarantee that there will not be any reactivity – tunneling is now possible. Conversely, reflections occur even at energies above the barrier.

Due to these differences, the true results could be rather different from what would be expected classically, especially for energies near to or below the barrier. They also make the mathematical approach different and more computationally expensive than the classical one.

2.2 Solution outline

To obtain state-to-state cross sections for H-CO, the Schrödinger equation needs to be solved. For that, a coordinate system is needed, which will be defined first. The Born-Oppenheimer approximation is used so that the nuclear motion of electrons can be treated as an energy surface.

A basis is then chosen, in which the wavefunction can be expanded. Besides the basis coordinates, the wavefunction is also a function of several quantum numbers, some of which will be constant during a collision, but others will not.

The equation will be solved by propagating the initial wavefunction using a time-dependent scattering method. This is repeated for many initial states. Each of these will produce probabilities for a range of energies for all outgoing states.

Finally, the state transition probabilities are extracted from the flux that is recorded during propagation. From these, $\underline{\underline{\boldsymbol{S}}}(E)$-matrix elements and cross sections are calculated.

2.3 Coordinate system

There is a choice of how to orient the coordinate system[21–23]. The choices that are convenient to work with are a space-fixed or a body-fixed basis. This is an important choice, since it determines which coordinates and quantum numbers will be used.

With **space-fixed coordinates**, as the name suggests, the orientation of the coordinate axes are fixed in space. This means that it is an inertial frame, which makes it seem like an attractive choice. However, there are more coordinates[24] that need to be used in this case to describe the overall rotation. In this case, the orbital angular momentum quantum number ℓ is used, which is related to the impact parameter.

With **boxy-fixed coordinates**, the coordinate axes are attached to the system, and rotate along with it. This means the frame is not inertial, and Coriolis forces will be present. The frame has an orientation relative to an inertial frame, and there are three angles that describe the orientation of the the body-fixed coordinates, but these angles will play a minor role.

Choosing boxy-fixed coordinates, the computation can be mostly concerned with the remaining 3 coordinates that describe the relative positions. The standard Jacobi coordinates will be used: r, R, and ϑ:

1. r is the separation of C and O (which will remain bound).
2. R is the distance between the H atom and the center of mass of CO (on r).
3. ϑ is the angle between the r and R axis.

The volume element in these coordinates is $dRdrd\cos\vartheta$.

Since the system described by these three coordinates is (potentially) rotating, it is not an inertial system and the laws of motion must be adapted to include centrifugal and Coriolis effects. This makes the algebra involved more complex[25, ch 3.3-4]. However, it is easier to make approximations such as the infinite order sudden approximation, which was not done in this work, but might be useful in the future.

2.3.1 Embedding

After choosing a coordinate frame in section 2.3 (Coordinate system), there is still a choice of axis to project the angular momentum on (typically labeled the z-axis). Any axis is equivalent in exact treatment, but only a few are convenient to work with or suitable for approximations, in this case R and r.

The reason that one may prefer one over the other is that Coriolis coupling may be stronger in one frame than in the other[26]. This means that, if one decides to use one of many possible approximations that limit the treatment of Coriolis coupling, one embedding may be better than the other. Because CO is very heavy compared to H, one might expect that the r coordinate leads to less coupling.

However, for similarity to earlier work[1] on H-CO, the R-embedding was chosen. In the current work, only a small part of the calculations neglected any Ω states, and then only the ones which are virtually unoccupied. If more such approximations are made in future work, it may be worthwhile to experiment with r embedding.

In the R-embedding, an Ω of zero means that the angular momentum is perpendicular to the R axis (which connects H and the center of CO). Because the plane of rotation is perpendicular to the angular momentum, this means that R is inside the plane of rotation of CO. Conversely, $\Omega = j \leq J$ would be a situation where CO rotates around the R axis.

It is possible to transform between both embeddings, using equations 6 and 7 in [26]. The code used, introduced in section 3.1 (The software), also has a parameter to choose embedding for the calculation.

2.4 Schrödinger equation

The goal for scattering is to solve the time-dependent Schrödinger equation, which describes how the wavefunction Ψ evolves in time t.

$$i\hbar\frac{\partial\Psi\left(R,r,\vartheta,t\right)}{\partial t}=\hat{H}\Psi\left(R,r,\vartheta,t\right). \tag{1}$$

This is a function of nuclear coordinates, because electron motion has been extracted into a potential energy surface using the Born-Oppenheimer approximation, as described in the next sections. After that, the Hamiltonian $\hat{H}$ will be specified.

2.4.1 Born-Oppenheimer approximation

The Born-Oppenheimer approximation is usually made in molecular scattering calculations. This assumes that, due to the much higher mass of the atom's nucleus compared to its electrons, the nuclear and electronic motions can be separated. The scattering event is seen as the motion of nuclei in the instantaneously adapting potential of the electrons, a potential energy surface (PES).

The PES gives the interaction energy as a function of nuclear positions. The nuclei can be viewed as moving on this surface during a scattering event. Interaction energies for individual positions are obtained in time-consuming calculations. These positions are then combined into a continuous surface by one of various fitting or interpolation procedures. This allows the surface to be reused for any discretization, greatly speeding up scattering (or other) calculations by no longer needing to calculate interaction energies each time.

A distinct advantage of separating nuclear and electronic motion, other than creating a reusable PES, is an immensely reduced dimension of the scattering calculation, with a spatial basis needed only for the nuclei during scattering, and not for the many electrons. It also allows for bigger timesteps, since the faster motion of electrons is already included in the PES. Without this separation, scattering calculations for systems of interesting size would be completely infeasible, since computational cost grows very rapidly in quantum mechanics.

2.4.2 H-CO surface by Song *et al.*

Song et al. [27] published a potential energy surface for the ground state ($\tilde{X}^2A'$) of H-CO in the full 3 dimensions in 2013. Accurate long-range behavior was obtained, which had previously [3–5] been found to be important to obtain matching rotational inelastic cross sections. Special attention was also paid to the well region, which was also found to be important for scattering.

The PES by Song *et al.* was created using around 4400 points calculated using Spin-unrestricted open-shell single and double excitation coupled cluster method with perturbative triples (RHF-UCCSD[T]). The system is open shell (not all spatial orbitals are filled with two paired electrons of opposite spin), meaning spin orbitals should be used and some correlation between only parallel-spin electrons is included due to antisymmetry of the Slater determinants.

The points were fit into a smooth surface using a combination of inverse powers of R for long range and the Reproducing Kernel Hilbert Space (RKHS) method for short range, and it was found to be accurate in both ranges. The

potential has been compared against earlier PES in 2D scattering as well as experimental vibrational and rotational constants, to good agreement[27].

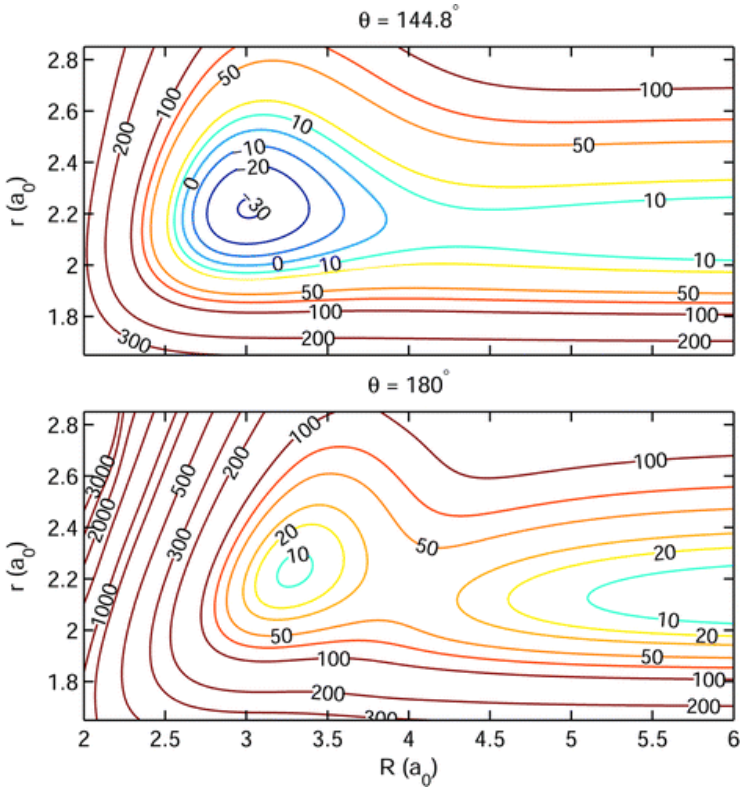


Figure 2: H-CO PES as a function of R and ϑ at fixed r. Image and PES from Song et al. [27].

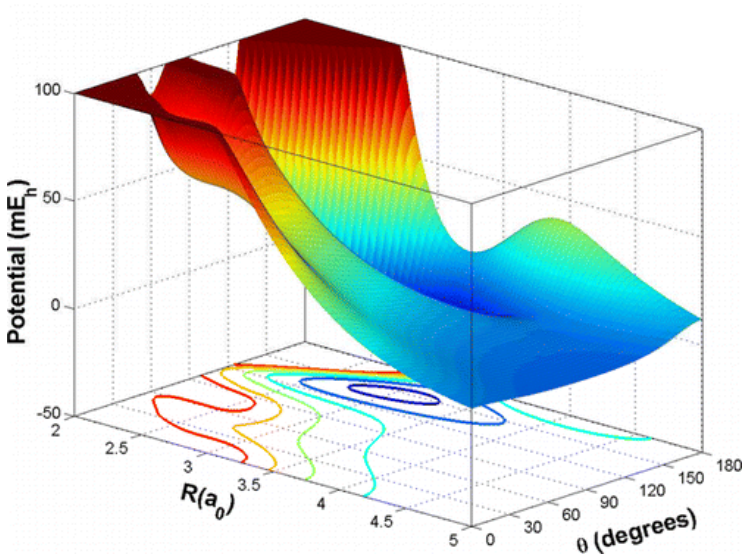


Figure 3: Countour of the H-CO PES for two different ϑ: equilibrium and linear configuration respectively. Image and PES from Song et al. [27].

2.4.3 Hamiltonian

The Hamiltonian for the H-CO system in the coordinates discussed above is[20, Eq. 5]

$$\hat{H} = -\frac{\hbar^2}{2\mu_R}\frac{\partial^2}{\partial R^2} - \frac{\hbar^2}{2\mu_r}\frac{\partial^2}{\partial r^2} + \frac{\left(\hat{J}-\hat{j}\right)^2}{2\mu_R R^2} + \frac{\hat{j}^2}{2\mu_r r^2} + V\left(R, r, \vartheta\right). \tag{2}$$

The potential $V\left(R, r, \vartheta\right)$ is the potential energy surface described in the previous sections.

This Hamiltonian is entered into Equation (1), at which point $\Psi\left(R, r, \vartheta, t\right)$ can be expanded and the time-dependence solved numerically.

2.5 Basis

For the wavefunction to be expanded, a basis is first chosen. Different coordinates use a different basis, which will be discussed below. After that, the quantum numbers of the wavefunction will be discussed.

2.5.1 Discrete variable representation

This section gives some general background about discrete variable representations. Generally speaking, one can use (a combination of) localized or global basis functions to describe the wavefunction. An example of the former would be a discrete grid, while an example of the latter would be vibrational eigenstates.

In the case of localized basis functions, the potential is easy to apply, as it can just be evaluated at the local points while being zero at all others. However, the kinetic term of the Hamiltonian may be easier in a global basis, where one can choose functions with analytical and possibly localized derivatives.

A discrete variable representation (DVR), also called the pseudospectral method, is a basis that combines some of the advantages from both.

As a general case, consider the wavefunction Ψ expanded in a carefully chosen set of N orthonormal basis function ψ_i:

$$\Psi(x) = \sum_{i=1}^{N} a_i \psi_i(x). \tag{3}$$

When using a discrete variable representation, one uses a quadrature to make the following approximation:

$$a_i = \int \psi_i^*(x)\Psi(x)dx. \approx \sum_{j=1}^{N} w_j \psi_i(x_j)\Psi(x_j). \tag{4}$$

In general, the points x_j are the eigenvalues of the position operator $\hat{x}$ in the basis ψ_i. Now the choice of basis functions $\psi_i(x)$ becomes important: this determines how good (and easy) this approximation is. For example, when using classical polynomials, this would be the Gauss-Legendre quadrature.

The x_j are the DVR points, with each point paired with one of set of orthonormal DVR functions:

$$\chi_i(x) = \sqrt{w_i} \sum_{j=1}^{N} \psi_j^*(x_i)\psi_j(x). \tag{5}$$

This is the DVR basis. What is special about it, and why does each basis function correspond to a point? The reason is that[28, p 3.3.5.1.ii]

$$\sqrt{w_n}\chi_i(x_n) = \sqrt{w_n}\sqrt{w_i} \sum_{j=1}^{N} \psi_j^*(x_i)\psi_j(x_n) = \delta_{in}. \tag{6}$$

This means that each DVR function is zero at all DVR points except for its own one.

As discussed, this locality means that the potential energy is easy to apply: it can just be evaluated at each grid point. This means computation time scales linearly with the number of grid points, rather than quadratically. The amount of memory needed for the potential energy is also decreased by a whole dimension.

The kinetic energy matrix can be computed from the derivatives of the ψ_i functions. This is typically a dense matrix. Since the ψs are usually known functions like polynomials, the first and second order derivatives may be analytical functions, such as in the case of a sinc DVR, making $\hat{H}$ fast to construct.

Another advantage of DVRs of the direct product variety is that they are sparse for multidimensional cases.

The advantage of the DVR comes from the fact that the Hamiltonian is easy to prepare: calculations are only needed for ϑ to construct the $\underline{\underline{\boldsymbol{V}}}$ matrix[29]. Furthermore, for some DVR choices, the kinetic part is analytical.

There are many types of DVRs, each with their own extra benefits.

2.5.2 R and r basis

A sinc DVR is used for the r and R coordinates. Like some other DVRs, the sinc DVR offers analytical expressions for the kinetic energy operators. The analytic kinetic energy is given exactly in an infinite sinc basic, but that is not feasible[29]. Specific to the sinc DVR, it has equally spaced grid points[29]. Together with its orthonormality, this makes potential energy relatively easy to apply. See also section 3.2.3 (Calculating $\hat{H}\Psi$).

A DVR grid was used for the r coordinate, rather than a basis of vibrational states. The reason for this is that the code that this project developed from also supports reactive scattering. In that case, a vibrational basis would not have been a good choice.

However, for a coordinate where not much change is expected, such the CO distance in inelastic scattering, a good set of basis functions such as vibrational eigenfunctions may offer a description using fewer points. For example, Lei *et al.* did use a vibrational grid[1], and used 9 basis functions, whereas these calculations need at least 12 DVR points at the same vibrational excitation. Another good choice may be a potential-optimized DVR.

2.5.3 ϑ and Ω basis

For the rotation-related coordinates ϑ and Ω, the situation is more complex. When applying the Hamiltonian, there is a transform between a finite basis representation of parity adapted angular functions $G_{j,\Omega}^{J,M,p}$ is used[20], and a DVR basis. The former makes the Coriolis coupling take less computation time, while the latter does the same for the application of the potential.

For the finite basis representation, consider $\boldsymbol{\alpha}$ the Euler angles of the complex with respect to the space-fixed frame, ϑ the angle between the H atom and the center of mass of CO, Θ_j^Ω the associated Legendre functions and $F_{\Omega,M}^J$ the reduced Wigner D functions, which are orthogonal.

$$G_{j,\Omega}^{J,M,p}(\boldsymbol{\alpha},\vartheta) = \sqrt{2+2\delta_{\Omega,0}}\Theta_j^\Omega(\vartheta)\cdot\left[F_{\Omega,M}^J(\boldsymbol{\alpha}) + (-1)^{J+\Omega+p}F_{-\Omega,M}^J(\boldsymbol{\alpha})\right]. \quad (7)$$

The $G_{j,\Omega}^{J,M,p}$ are eigenfunctions of $\hat{J}^2$, $\hat{J}_z^{SF}$, $\hat{J}_z^{BF}$, parity, and $\hat{j}^2$, with eigenvalues proportional to $J(J+1)$, M, Ω, $(-1)^p$ and $j(j+1)$ respectively, where $p \in \{0,1\}$[20].

Parity (p), J, and M are constant during the propagation as they are good quantum numbers.

The quantum numbers j and Ω are not conserved, so there is a basis function for each combination of j and Ω where $|\Omega|$ is smaller than the minimum of J and j.

Note that the $G_{j,\Omega}^{J,M,p}$ are eigenfunctions of $\hat{J}_z^{BF}$, so the basis functions are pure Ω states. But $\hat{j}$ is a vector quantity whose components do not commute, so the $G_{j,\Omega}^{J,M,p}$ are eigenfunction of $\hat{J}^2$ and $\hat{j}^2$ instead. Pure rotational states will thus be described by a combination of basis functions.

It can be verified using Equation (7) that if $J+p$ is odd, then $\Omega = 0$ yields $G_{j,0}^{J,M,p} = 0$, so this state does not exist.

2.6 Quantum numbers

2.6.1 States

To describe the quantum state of the system, a complete set of commuting observables is needed. This means that all the observables must be knowable at the same time (their operators must commute). And there must be enough observables to describe everything that can be simultaneously known about the system (which may be less than is the case classically). Much of the state information is captured in the potential energy surface, see section 2.4.2 (H-CO surface by Song *et al.*), which has been assumed independent from the scattering coordinates (Section 2.4.1). Section 2.6.2 and 2.6.3 describe the observables used.

2.6.2 Good quantum numbers

There are several good quantum numbers. "Good" is used here in the quantum-mechanical sense that their corresponding states stay pure during the interaction, so these quantum numbers maintain a single value over time. They are:

1. J is a good quantum number, since it represents total angular momentum which is conserved.

2. M, the projection of J on the space-fixed z-axis (see the section on body-fixed coordinates), is a good quantum number. But in the absense of external magnetic fields, it does not have an effect on the interaction. All it effectively does is make J levels degenerate, with degeneracy factor $2J + 1$.

3. Parity p, which determines the sign of the wavefunction under inversion of the space-fixed coordinates, is also a good quantum number. Parity $p = 0$ and $p = 1$ are referred to as positive and negative parity respectively, since they appear as $(-1)^p$ in equations[2].

At a given initial rovibrational state and energy, increasing J states (partial waves) eventually stop contributing. This follows because J is related to the impact parameter b through reduced mass μ and relative velocity v between the particles:

$$\boldsymbol{J} = \boldsymbol{j} + \boldsymbol{\ell}. \tag{8}$$

$$\ell = \mu v b. \tag{9}$$

A higher impact parameter means a bigger distance between H and CO, and less reactivity. The equivalent in the non-inertial body-fixed coordinates is that the centrifugal barrier gets too high.

In the situations studied up to this point, the initial value of j was often substantially lower than that of J, so J is approximately proportional to ℓ. This may change if higher states are calculated in the future.

Since these quantum numbers are conserved, calculations at different values can be performed in isolation.

[2]Jumping ahead a bit, note that -1 and $+1$ should be used as input to the Fortran code.

2.6.3 Coupled states

There are also several states which do not (necessarily) stay pure during scattering, which constrains the possible values:

1. Vibrational level v of CO, which is described by the r coordinate.

2. Rotational level j of CO.
   In the case of vibrational (v) and rotational (j) state, the coupling is exactly what this work aims to quantify. Both v and j must be non-negative, with their upper limits set by available energy, including potential energy when H and CO are close together.

3. Ω is the projection of both J and j. Different Ω states get mixed due to Coriolis coupling. Similar to the classical Coriolis effect, this happens because the body-fixed coordinates are non-inertial. It is part of the angular terms in Equation (2), which also includes the centrifugal barrier, another effect of the non-inertial coordinates.
   Each Ω state is only coupled to its neighbour $\Omega \pm 1$[20, Eq. 7]. It will be seen in section 3.2.6 (Parallelism) that this can be used to parallelize calculations.
   Unlike v and j, the differentiated effect per Ω state is not of interest for the final result. The different final Ω state contributions are summed at the end of each job, and initial Ω state contributions are averaged.

Since transitions between these states do happen, all accessible states must be considered in each computation. One starts with a wavefunction with pure v, j, and Ω values and the interaction is simulated, causing the wavefunction to become a mix of various v, j and Ω states. The rovibrational states are analyzed, and Ω states are summed.

2.6.4 Restrictions

There are some relations between the different quantum states:

1. Ω is the projection of $\hat{J}$ and $\hat{\jmath}$ on the z-axis. As such, is cannot be larger than either J or j.

2. As discussed in the section about the ϑ basis (Equation (7)), if $J + p$ is odd, then the state with $\Omega = 0$ is not allowed. In that case, Ω has values from 1 to J, instead of from 0 to J.

3. As a consequence of the above, when $J = 0$, parity must be positive ($p = 0$). Negative parity would mean $\Omega = 0$ does not exist, which at $J = 0$ means there is are no Ω states at all.

4. Because angular momenta are vector quantities and $\boldsymbol{J} = \boldsymbol{j} + \boldsymbol{\ell}$, it *is* possible for eigenvalues j to be larger than J throughout the calculation.

The product $(-1)^p \cdot \Omega$ take all $2J+1$ values from $-J$ to $+J$ once, guaranteeing an equal number of states in body-fixed coordinates as there are ℓ states in space-fixed coordinates. Note though that parity is conserved, whereas Ω is not.

2.6.5 Continuous quantities

Finally, the relative kinetic energy does not have a quantum number: it is a continuous quantity because the system is not bound in the R coordinate. Kinetic energy is not conserved during the collision.

Note that the kinetic energy does not have a single value, but rather a distribution. This is because only plane waves have an exact energy, while wavepackets are used in this work. The width of the wavepacket in energy is inversely proportional to the width of the wavepacket in R, a properly of a Gaussian shape and its Fourier transform.

Having a wavepacket that contains a range of energies, has the advantage that results for a range of energies can be obtained in a single calculation. This range is not unlimited though, as the wavepacket cannot be made arbitrarily small.

2.7 Wavefunction

2.7.1 Transformed wavefunction

To make it easier to solve the Schrödinger equation, a transformed wavefunction $\Psi^{J,M,p}(R,r,\vartheta,t)$ is used. This is the proper wavefunction multiplied by Rr, because that is easier to integrate with the volume element mentioned in section 2.3 (Coordinate system).

This wavefunction is expanded in products of the vibrational (r), scattering (R), and angular (ϑ) basis functions. The propagation of the wavepacket in time is captured in the expansion coefficients $C^{J,M,p}_{\lambda\nu j\Omega}(t)$. The basis functions themselves are static.

$$\Psi^{J,M,p}(R,r,\vartheta,t) = \sum_{\Omega=\Omega_{min}}^{J} \sum_{j\geq\Omega}^{j_{max}} \sum_{\nu=1}^{\nu_{max}} \sum_{\lambda}^{N_R} C^{J,M,p}_{\lambda\nu j\Omega}(t)\,\psi_\lambda(R)\psi_\nu(r)G^{J,M,p}_{j,\Omega}(\boldsymbol{\alpha},\vartheta)\,.$$

Entering the expansion above and the Hamiltonian from Equation (2) into the time-dependent Schrödinger equation (1) yields the expression given by equation 7 of Meijer and Goldfield [20].

2.7.2 Initial wavefunction

The initial wavefunction is a product of coordinate functions, which will be described in the next sections.

2.7.3 Initial R wavefunction

The initial wavepacket in the free R coordinate is given by a typical product of an infinite plane wave and a localized envelope[20]:

$$\Psi_{scat}(R) = \frac{1}{\sqrt[4]{2\pi\beta}} \exp\left(\frac{-(R-R_0)^2}{4\beta}\right). \tag{10}$$

The width of this wavepacket can be chosen through the parameter β and its center position through R_0. It is important to note that making the wavepacket wider in R makes it more narrow in kinetic energy and vice versa. This is the case because these distributions are related by a Fourier transform, and the width of a Gaussian distribution and its Fourier transform are inversely proportional.

Only results for those energies with enough wavefunction amplitude are reliable, so a wavepacket with big energy spread seems preferable. But note that there must be enough amplitude at each of the energies in the range. Also note that initially, any substantial amplitude of the initial wavepacket must correspond to positive energies, and must be outside the potential, the centrifugal barrier, the flux surface and the absorption region. This limits the maximum energy spread.

2.7.4 Initial vibrational (r) wavefunction

The initial wavepacket in the bound r coordinate is given by a vibrational eigenstate of CO which can be chosen for each calculation. It is calculated at $R = R_0$, which should have such a value that H is far enough away to have negligible effect on CO.

2.7.5 Initial ϑ wavefunction

The initial wavepacket in the ϑ coordinate is a rotational eigenstate. The shape of the ϑ basis functions, and hence the occupation, will depend on J, p, and Ω, as seen in section 2.5.3 (ϑ and Ω basis). The basis functions are not pure function of j and Ω, so various basis function will be occupied, visible in Figure 4.

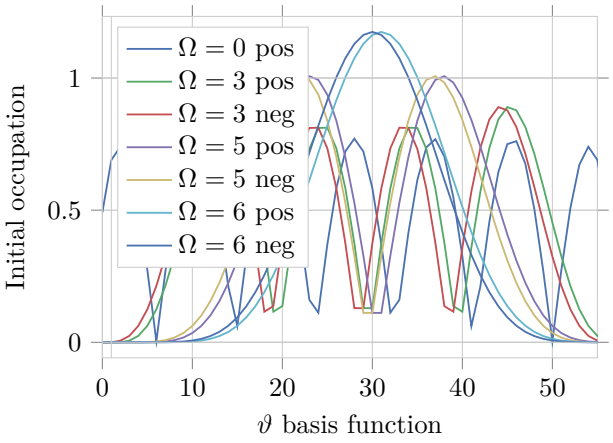


Figure 4: The initial occupations of the ϑ basis functions at different Ω and parity ($J = j = 6$). The R and r dimensions are integrated over. The figure is symmetric.

2.7.6 Initial Ω wavefunction

The wavefunction starts in a pure Ω state, and each omega is handled by a different CPU process. Therefore, the whole wavefunction starts out in one

process. It is then spread to others during the scattering interaction, through Coriolis coupling to its direct neighbors.

2.8 Propagation

The propagation of the wavepacket through time is alternatively called integration.

2.8.1 Time-dependent vs time-independent

Available methods for solving the Schrödinger equation can be generally classified as time-dependent and time-independent, although techniques from each increasingly get used by the other field (e.g. time-independent wavepackets[30]), making the distinction less useful.

Time-independent computational methods such as the one used by Song solve the time-independent Schrödinger equation, obtaining a kind of snapshot. They generally compute all state-to-state reaction probabilities at a fixed energy in one calculation. This yields all elements of a scattering matrix ($\underline{\underline{\boldsymbol{S}}}(E)$-matrix) at that energy, and must then be repeated for each energy.

Time-dependent wave packet methods use largely different methods, starting with a wave packet in a single initial state and calculating its evolution in small timesteps. The result is an $\underline{\underline{\boldsymbol{S}}}$-matrix for a single initial state to all final states, for a range of energies. This corresponds to one column in a number of fixed-energy $\underline{\underline{\boldsymbol{S}}}(E)$-matrices.

Time-independent methods are more suitable for low energy calculations, for several reasons[31]:

1. At low energy, time dependent methods need long propagation times because movement is slow.

2. At low energy, the wavefunction is more likely to get trapped in a potential minimum for a time, exacerbating the above effect.

3. Higher quantum states are not accessible at low temperature, making the matrices that time-independent methods use smaller. This speeds calculations up substantially, since eigenvalue calculations may have complexity $\mathcal{O}\left(N^3\right)$ in the absence of useful symmetry properties[32, p385].

In general, time-dependent wave packet methods scale better with increasing system size than do time independent methods[28, p68]. However, all quantum methods are quite limited in the size of systems that can be treated using current hardware.

Another important difference is that time-independent methods produce all input and output states in a calculation, but at a single fixed energy. Time-dependent methods produce results for a range of energies, but only a single input state. Hence, depending on how many input states and how fine an energy grid is needed, one or the other may be favored.

This work uses a time-dependent method in the traditional sense, whereas earlier work[1, 18, 33] on H-CO used a time-independent method. This project can be compared to said earlier work as a form of verification, but can extend results to a wider range of physical states.

2.8.2 Integration

The equation to describe the evolution of a wavefunction $\Psi(\boldsymbol{x})$ in time by a step of Δt is

$$\Psi(\boldsymbol{x}, t + \Delta t) = \exp\left(-\frac{i\hat{H}(\boldsymbol{x})\Delta t}{\hbar}\right)\Psi(\boldsymbol{x}, t). \tag{11}$$

Though this seems straightforward, to be able to apply the exponential of an operator like $\hat{H}(\boldsymbol{x})$, it must be expanded into in a power series which may not terminate. Some tricks will be described.

2.8.3 Real wavepacket integration

Wavefunctions and operator matrices in quantum mechanics consist of complex numbers. But computationally, it is possible to use only a real-valued transformation for most of the propagation, while still getting the information of interest. In this work, this is achieved using a method by Gray and Balint-Kurti [34], although other methods exist.

The real-valued propagation relation is obtained by adding Equation (11) to the same equation, but with a negative timestep:

$$\Psi(\boldsymbol{x}, t + \Delta t) = -\Psi(\boldsymbol{x}, t - \Delta t) + 2\cos\left(\hat{H}(\boldsymbol{x})\Delta t/\hbar\right)\Psi(\boldsymbol{x}, t). \tag{12}$$

A reference to two timesteps back has been added, so two initial steps will be needed.

This no longer contains the i, and does not mix the real and imaginary parts. Therefore, the real and imaginary parts can be propagated independently.

Yet if the flux can be obtained using only one of these, then the imaginary part need not be integrated at all (beyond the first step). As discussed in section 2.9.1 (Analysis approach), this real analysis is indeed possible.

This is very beneficial for performance: multiplications may be more than four times faster for real values compared to complex ones[35]. Memory use stays similar, since two real wavefunctions must be kept instead of one complex one. The timestep is often smaller for the real wavepacket method, which requires more propagation steps, but this effect is smaller than the savings.

Because the observables do not explicitly depend on time, a transformed Hamiltonian and wavefunction can be used[34, II B].

A choice which simplifies the integration is

$$f(\hat{H}) = -\frac{\hbar}{\Delta t}\cos^{-1}\left(\hat{H}_s\right), \tag{13}$$

for a Hamiltonian $\hat{H}_s$ that has been linearly scaled to have eigenvalues between -1 and $+1$ (see section 2.10.3 (Range of eigenvalues)). The eigenvalues of $f(\hat{H})$ are obtained from the eigenvalues of $\hat{H}_s$ by the same transformation f.

The resulting equation

$$\tilde{\Psi}(\boldsymbol{x}, t + \Delta t) = -\tilde{\Psi}(\boldsymbol{x}, t - \Delta t) + 2\hat{H}_s(\boldsymbol{x})\tilde{\Psi}(\boldsymbol{x}, t) \tag{14}$$

is not only more simple, but is also that of Chebyshev recursion. A damped version also exists; this work does damping as a separate step. The $\tilde{\Psi}$ are real, and are different from the Ψ before due to the transformation of $\hat{H}$.

The Chebyshev series has many interesting properties. It is, for example, related to the Fourier cosine series, which allows shortcuts when transforming to energy domain[36]. It is also complete and orthonormal, and offers the best approximation to bounded continuous functions, by some measures of 'best'.

This is accurate and stable for tens of thousands of steps[36] if the spectral range of $\hat{H}_s$ is in the correct range and 64-bit floats are used.

Equation (14) refers to two earlier steps. The first step is just the choice of initial wavefunction. But the second step requires one iteration to be calculate using separately using a process that does not need the last two steps.

This is done as a separate Chebyshev iteration using a tunable number of steps to compute this equation through its expansion[34, II B]:

$$\tilde{\Psi}_{\mathrm{Re}}(\boldsymbol{x}, t+\Delta t) = \hat{H}_s \tilde{\Psi}_{\mathrm{Re}}(\boldsymbol{x}, t=0) - \sqrt{\hat{H}_s^2 - 1}\tilde{\Psi}_{\mathrm{Im}}(\boldsymbol{x}, t=0), \tag{15}$$

with $\tilde{\Psi} \equiv \tilde{\Psi}_{\mathrm{Re}}$.

This uses the complex wavefunction, the imaginary part is discarded after this step.

Note that only one previous wavefunction must be stored, and $\hat{H}_s$ need not be stored as long as its effect on the wavefunction can be computed, so the method is memory efficient.

2.8.4 Absorption

To prevent the wavefunction from reaching the end of the grid, reflecting and getting absorbed again, wavefunction amplitude near the end of the grid is absorbed at the end of each timestep. This is done according to the equation:

$$C_{\lambda\nu j\Omega}(t+\Delta t) = e^{-B(R_\lambda - R_{\mathrm{abs}})^2} C_{\lambda\nu j\Omega}(t). \quad \Big| \quad R_\lambda > R_{\mathrm{abs}} \tag{16}$$

This has been plotted in Figure 5.

If the absorption is too strong, the absorption region itself will cause reflections, defeating its purpose. The absorption strength B should be as high as possible without significant reflection, with the absorption region large enough to fully absorb the wavepacket.

The wavefunction is only absorbed in the R coordinate. In the r coordinate, the wavefunction should not reach the end of the grid, since the CO bond should not break in non-reactive scattering.

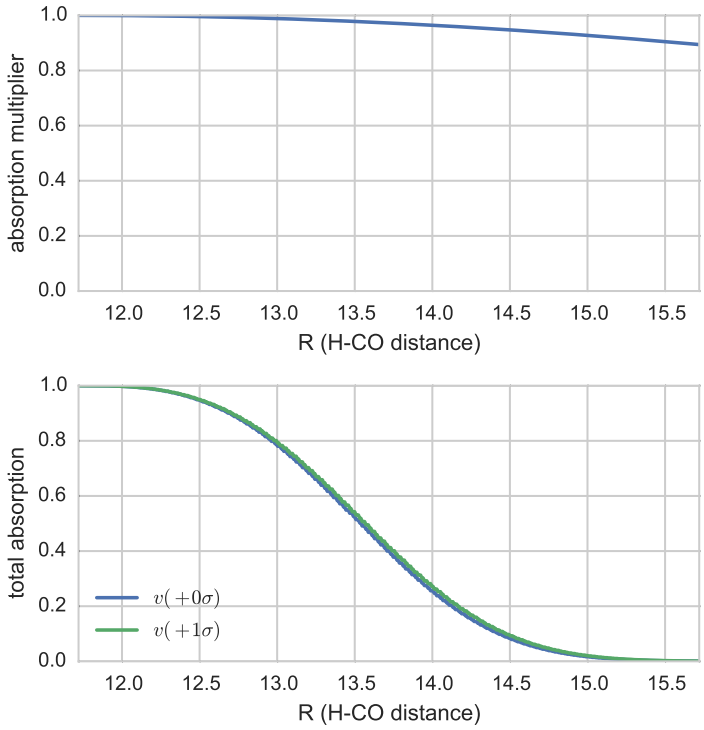


Figure 5: Approximation of the absorption as a function of R for one scattering calculation. Green lines represent the absorption if the (classical) velocity is one standard deviation higher, since the initial wavepacket represents a range of kinetic energies.

2.9 Analysis

After the propagation, the results must be analyzed.

In the present work, only non-reactive scattering is studied. Therefore flux analysis and absorption happen only in the R channel. For reactive scattering, the same procedure can be repeated in the r channel, which is implemented in the code.

The quantities of interest are the kinetic and rovibrational energy after interaction. The scattering direction is not analyzed in this work, since the primary purpose is to obtain gas-phase reaction rates, which do not depend on direction.

2.9.1 Analysis approach

After simulating the interaction, the results must be extracted. This happens by analyzing the recorded flux moving away in the R-channel.

For this, a surface is chosen that is far enough away to be considered asymptotic. It must be further away than the initial wavefunction (which has a spread), but before the start of absorption.

Every N steps, at this surface, the wavefunction and its derivative are determined. These are stored as a time series, which is analyzed after the integration is complete[20, p II C].

The first results of interest are the outgoing rotational and vibrational excitation levels. To obtain these, the wavefunction (derivative) slices in the time series are expanded into rovibrational states.

The other result of interest is the kinetic energy. To obtain this, the time series is Fourier transformed into an energy spectrum. With enough snapshots, no precision is lost by doing this discretely.

Some final operations transform these results into $\underline{\underline{\boldsymbol{S}}}$-matrix elements, $\underline{\underline{\boldsymbol{T}}}$-matrix elements or in this work, probabilities. These are then available for each outgoing state as a function of energy[21, appendix C].

2.9.2 Real wavepacket analysis

As described in section 2.8.3 (Real wavepacket integration), only the real part of the wavepacket is propagated, and it is different due to the Hamiltonian being transformed. The analysis will have to be adapted somewhat.

The approach is in fact rather similar to that for complex wavefunctions. Tracking the transformations takes some careful work, which is done in appendices A and B of Gray and Balint-Kurti [34]. A very brief summary:

1. E and $\hat{H}$ are replaced by $f(E)$ and $f(\hat{H})$ [with f from Equation (13)].
2. The eigenfunctions are different, but the equations stay the same.
3. The only difference for the $\underline{\underline{\boldsymbol{S}}}$-matrix equation is that the real-valued energy-dependent flux coefficients are multiplied by two.

The results are independent of the timestep Δt, as expected physically.

This yields $\underline{\underline{\boldsymbol{S}}}$-matrix elements, which can be used to calculate probabilities, cross sections and finally reaction rates.

2.9.3 Cross sections

The analysis in the previous section yields $\underline{\underline{\boldsymbol{S}}}$-matrices, that give the probability of each rovibrational transition as a function of energy at specific J, Ω_i and p.

Often, parameters of the overall configuration are not of interest, since they are difficult to control experimentally. A way to express the reactivity with this information summed out is with the state-to-state cross section. This has units of surface and is easy to compare with the classical hard sphere model.

The equation is similar to [38, Eq. 4.230] and [19, Eq. 3.2]:

$$\sigma_{vj\to v'j'}(E_{vj}) = \frac{\pi}{(2j+1)\,k_{vj}^2} \sum_{J=0}^{J_{max}} (2J+1) \sum_{p}^{+,-} \sum_{\Omega=0\vee 1}^{\min(j,J)} \sum_{\Omega'=0\vee 1}^{\min(j',J)} |\underline{\underline{\boldsymbol{T}}}^{J,p,\Omega,\Omega'}_{vj\to v'j'}|^2, \tag{17}$$

where the transmission matrix is $\underline{\underline{\boldsymbol{T}}}^{J,\Omega,\Omega'}_{vj\to v'j'} = \delta_{vv'}\delta_{jj'} - \underline{\underline{\boldsymbol{S}}}^{J,\Omega,\Omega'}_{vj\to v'j'}$.

The factor $2J+1$ stems from degeneracy caused by the M quantum number.

Note that the cross section is not an actual size of the particle - the classical concept of size does not apply to quantum mechanical particles. Note that the cross section is not a fixed particle property either, as the above shows it changes with interaction energy.

2.9.4 Rate coefficients

Cross sections describe the reactivity at a fixed particle energy. But in many real-life systems, particles have different energies, often distributed according to the Boltzmann distribution at the temperature of the material.

This is described by a reaction rate, which integrates out energy, weighted by the prevalence of that energy in the Boltzmann distribution, yielding a rate as a function of temperature T.

The precise equation, from Song [19, Eq. 4.8]:

$$r_{v,j\to v',j'}(T) = \sqrt{\frac{8}{\pi\mu\ k_b^3 T^3}} \int_0^\infty \sigma_{v,j\to v',j'}(E_{v,j}) \exp\left(\frac{E_{v,j}}{k_b T}\right) E_{v,j}\ dE_{v,j}. \quad (18)$$

2.10 Optimizations

Now that the approach is clear, there are a few methods that are used to make the computations faster. They are section 2.10.1 (Grid filtering), section 2.10.2 (Computing potential energy) and section 2.10.3 (Range of eigenvalues), and are described here at the end to not interrupt the method description above with 'implementation details'.

2.10.1 Grid filtering

To save computation time and memory usage, those grid points that are unreachable are removed from the DVR grids for R and r. The first criterion for this is that points for which all three atoms are far apart are removed. This is useful in reactive scattering, but has a smaller effect for a properly chosen rectangular non-reactive grid, because large separations of carbon and oxygen are already outside the grid.

As a second criterion, all points are removed for which the potential energy is too high, tunable by a parameter. The number of points removed is modest, but since many time-consuming operations scale quadratically or worse with the number of grid points and filtering is fairly fast, it is worthwhile. It is also useful to limit the spectral range of the Hamiltonian, see section 2.10.3 (Range of eigenvalues).

Special handling is included to deal with the case that a row of grid points is interrupted by dropped points in the middle. Coupling between these regions is still taken into account in that rare case, mostly because the code already existed.

For more details about this part of the calculation, we refer to the appendix of [20].

This would likely save quite some grid points, since if m v-states are needed at $j = 0$ and n v states at $j = 0$, then the $v = m, j = n$ state might require as much as twice the available energy.

2.10.2 Computing potential energy

As explained in section 2.4.2 (H-CO surface by Song *et al.*), a potential energy surface for the interaction between H and CO is available. This will now be used to compute the potential energy term for the Hamiltonian, $V(R, r, \vartheta)$. For this, the energy for each of the basis functions must be computed.

As an advantage of using a DVR basis for the R and r coordinates, getting the interaction energy in those coordinates $V^{\Omega}(R_\lambda, r_\nu,)$ just involves an integration over the remaining ϑ coordinate of $V_{PES}(R_\lambda, r_\nu,)$ at those grid points[20].

$$\tilde{V}^{\Omega}_{jj'}(R_\lambda, r_\nu) = \int_{-1}^{+1} \Theta^{\Omega}_{j}(\vartheta) V_{\mathrm{PES}}(R_\lambda, r_\nu, \vartheta)\, \Theta^{\Omega}_{j'}(\vartheta) dcos(\vartheta). \tag{19}$$

This integral is solved numerically using a Gauss-Legendre quadrature. The potential can then be applied to a coefficient $C_{\lambda\nu j\Omega}(t)$ as

$$V^{\Omega}(R, r, \vartheta)\, C_{\lambda\nu j\Omega}(t) = \sum_{j' \geq \Omega}^{j_{max}} \tilde{V}^{\Omega}_{jj'}(R_\lambda, r_\nu)\, C_{\lambda\nu j'\Omega}. \tag{20}$$

2.10.3 Range of eigenvalues

Because the cosine used in Equation (13) is defined for the range -1 to $+1$, all eigenvalues of $\hat{H}_s(\boldsymbol{x})$ must be in this range. This is simple to arrange if the range of eigenvalues is known:

$$\hat{H}_s(\boldsymbol{x}) = \frac{2\hat{H}(\boldsymbol{x}) - \hat{I}(E_{min} + E_{max})}{E_{max} - E_{min}}. \tag{21}$$

If eigenvalues have magnitude larger than 1, the integration is not stable. However, if only a part of the range is used, then the updated per iteration will be smaller.

The eigenvalue range was determined using the Lanczos algorithm, specifically the Hermitian variant. This is a numerical, iterative approximation method. For a Hermitian matrix, it constructs a symmetric, tridiagonal matrix that is similar to the original Hermitian matrix, where "similar" is understood in the mathematical sense of representing the same operation in different bases. Similar matrices have the same characteristic polynomial and therefore the same eigenvalues.

The Hermitian matrix does not need to be available, it must merely be possible to compute $H_{n,n}\psi_n$. The range is determined by the extreme eigenvalues, and these converge fastest. The number of evaluations needed is much smaller than the dimension of the matrix, several dozen is typical.

Note in Equation (14) that that the value of Δt no longer affects the propagation. It is indeed not an input to the program. The update per iteration instead depends on the scale of $\hat{H}_s$.

3 Computational

3.1 The software

The Fortran code used for this work was originally written by prof. Meijer as described in [20, 26, 39]. It was adapted and extended for this project, summarized in attachment section 6.4.1 (Obstacles & improvements). It counts 13.149 lines, of which 7.948 unique.

The Python code was entirely written from scratch for this project, and counts 12.000 with little duplication. The queue management system which was written as a reusable component, fenpei[40], counts 2.110 lines was also written for this project.

An extensive usage manual describing the code, the parameters and how to tune them is attached in section 6.2.2 (Physical parameters).

Also included is an example of a single job directory, before and after calculation (see Code structure). This would normally be generated automatically, and is included for reference and testing. Execute readygo.sh to start the job.

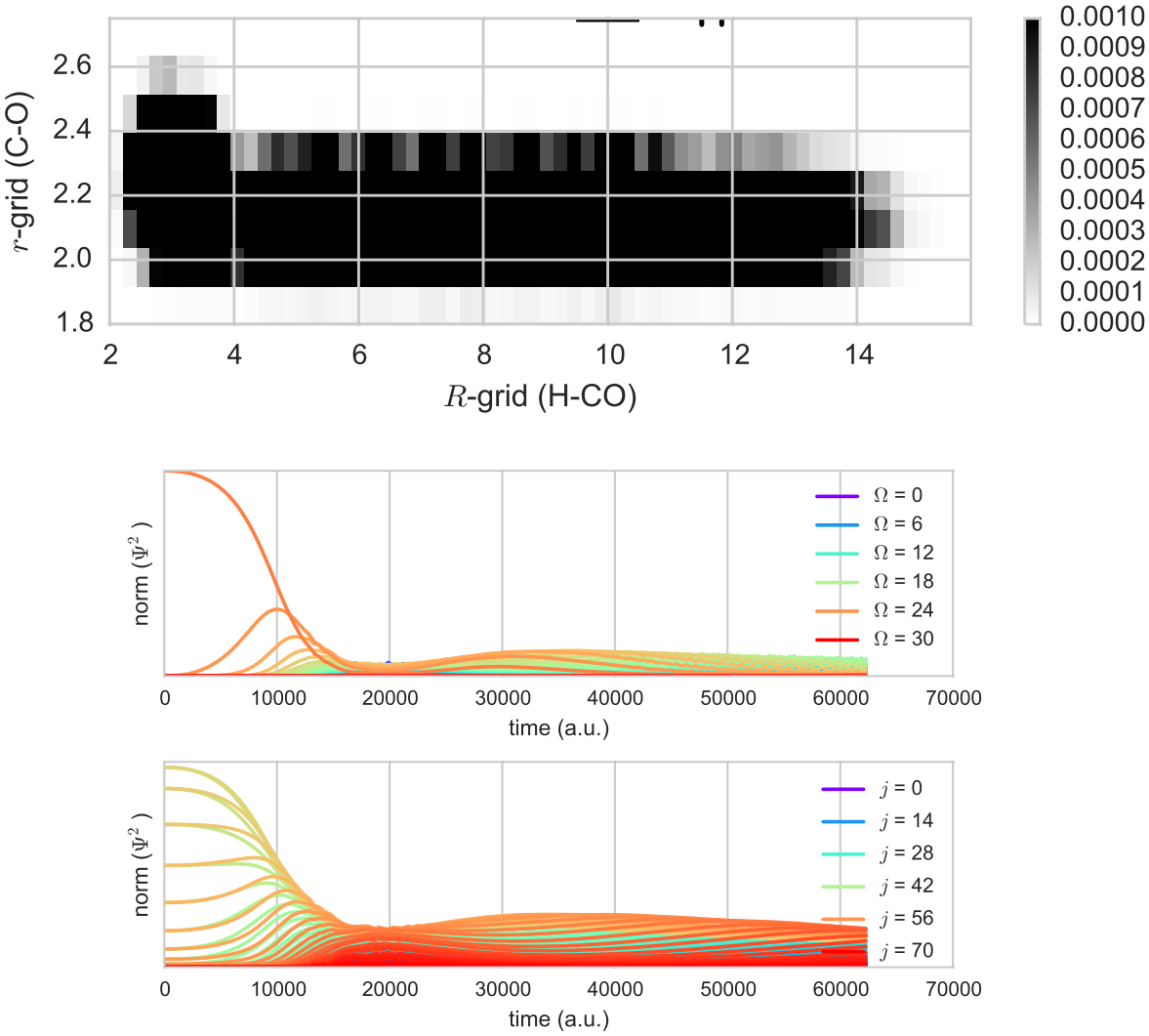


Figure 6: Basis function occupation recorded at several snapshots every 100 integration steps. **First**: the occupation of the R-r grid, with angular states summed out. High occupations are all black to make low occupations visible. The range, dark and light marker at the top are the initial wavepacket, the flux surface and the start of the absorption region respectively. $v_i = j_i = \Omega_i = J = 0$ **Second**: The occupation per Ω is plotted against timestep for part of the integration - at larger times, most of the wavefunction has been absorbed, and this is removed from the plot. Note that each Ω state is computed by a different process, and any exchange happens through inter-process communication. $v_i = 0$, $J = 50$, $j_i = \Omega_i = 25$ for this and the next figure. **Third**: The occupation per ϑ basis function is plotted against timestep for part of the integration (truncated as above). Many states are initially occupied, since the basis functions are not eigenfunction of $\hat{j}$ - see also figure 4.

3.2 Performance

Good performance is important, as individual jobs can take hours of CPU time, and the whole calculation takes thousands of jobs.

The most important effect on computation time of individual jobs is the number of omega states, which is primarily determined by J. This is shown in Figure 7.

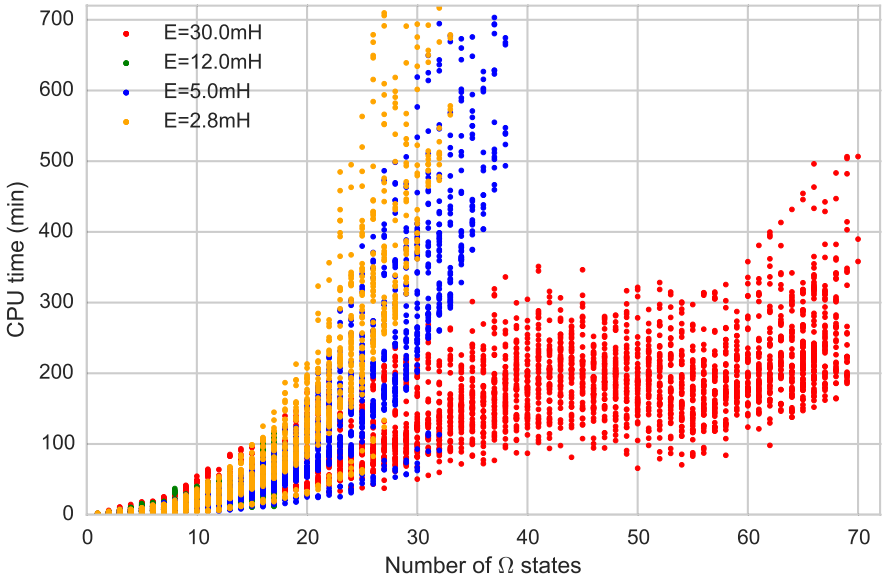


Figure 7: Computation time (in CPU minutes) is compared against the number of omega states for different initial kinetic energies. Other parameters vary; the variance due to that is seen to be smaller than that due to kinetic energy or number of Ω states. Hardware differences were compensated for. Note that more omega states were neglected for kinetic energy 12 mH.

As seen in the figure, the relation between time and number of Ω states is not linear. However as a very rough approximation, computations take 5 minutes of CPU time per Ω state. If the number of Ω states is less than than the number of CPU cores, the real-world time is just a bit more than this.

Five minutes per Ω may not sound like much. But over 17.000 jobs with up to dozens of Ω states puts the total at over 3 years of CPU time.

3.2.1 Split jobs

The smallest calculation that can be performed in isolation must consider all accessible R, ν, j, and Ω values, as these are not conserved.

It starts in a pure ν, j, and Ω state for a range of kinetic energies.

These states get mixed, and output is produced for all ν and j channels, at the energy range. Ω channels are summed as they are not of interest.

During such a computation, J and p are conserved, so different values can be computed in independent calculations.

3.2.2 Computational complexity

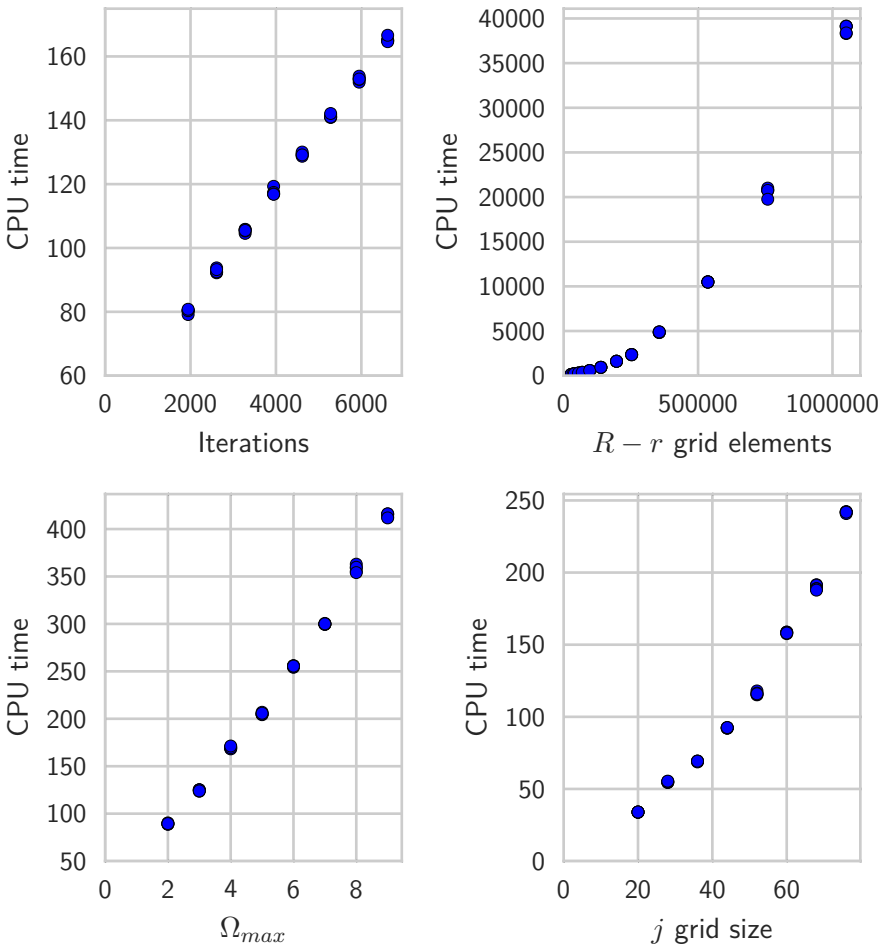


Figure 8: This figure shows the trend in CPU time as a function of the parameters discussed in section 3.2.2 (Computational complexity).

It is the states that get mixed, as well as the continuous quantities, that contribute most of the high CPU requirements of these calculations. The wavefunction has an amplitude for every combination of states, scaling approximately as the product of allowed values per quantum number[3].

For a specific initial $\nu = \nu_i$ and $j = j_i$, and a small range of kinetic energies, how many such calculations are needed to compute all partial waves up to J_{max}? All the output rovibrational output states are computed at once, but all initial Ω states must be computed and added. Combining this with parity, there are $2\min(J, j_{max}) + 1$ states to be added. This must happen for all J up to J_{max}.

[3]In reality, it depends on the basis size, which is only approximately proportional to the number of quantum states.

Using $q \equiv \min(J_{max}, j_{max})$, a total of $(2q+1)(J_{max}+1) - q(q+1)$ calculations must be performed.

Keep in mind that the difficulty of individual calculations also scales approximately linearly with $\min(J, j_{max})$, since even though there is a pure Ω state initially, all Ω levels are taken into account due to mixing. Higher Ω states are easier to compute though, because the j basis is smaller as $j < \Omega$ is not allowed. j_{max} likely needs to be higher if J is higher. For these reasons, including more J-states is very expensive. At the same time, because for every partial wave there are $2J+1$ degenerate m states that contribute, it is often important to include high J states (16).

These jobs for partial waves must then be repeated for each of the ν_i, j_i, and each of the desirable energy ranges ($N_\nu N_j N_{\Delta E}$)

Neglecting initialization and analysis, the runtime complexity is:

- Linear ($\mathcal{O}(N_{\Delta t})$) in the number of timesteps, since each step is the same and depends only on the previous step. The size of a timestep differs between calculations, but only by a factor two for comparable runs.
- Linear ($\mathcal{O}(N_\Omega)$) in the number of Ω states, since these only depend on neighboring Ω states. Note that Ω states are not equally demanding since $j \geq \Omega$. The communication overhead also increases with more Ω states. Neither effects are reflected in the runtime complexity.
- The application of kinetic energy has complexity:
  - $\mathcal{O}\left(N_R^2 N_r N_j\right)$ for the scattering (R, size N_R) coordinate.
  - $\mathcal{O}\left(N_R N_r^2 N_j\right)$ for the vibrational (r or ν, size N_r) coordinate.
  - $\mathcal{O}\left(N_R N_r N_j^2\right)$ for the angular (j, size N_j) coordinate.

  This is a very time-consuming step in calculating the effect of the Hamiltonian on the wavefunction. Among these three, the scattering coordinate usually dominates.

This is visible in Figure 8. It can be seen that CPU time indeed increases linearly with the number of iterations and the number of Ω states, with overhead noticeable in the latter case. Complexity if quadratic in the grid sizes, with the angular case only becoming slightly visible at high values, because the scattering coordinate dominates.

Overall, a number of runs proportional to $N_\nu N_j N_{\Delta E} J_{max} \min(J_{max}, j_{max})$, with complexity $\mathcal{O}(N_{\Delta t} N_\Omega N_R N_r N_j \max(N_R, N_r, N_j))$ per run.

3.2.3 Calculating $\hat{H}\Psi$

Propagating the wavefunction in time, as well as other aspects of the calculation, will depend on applying the Hamiltonian to the wavefunction. This is a time-consuming process[4], which should be highly optimized.

Therefore, applying $\hat{H}\Psi$ is not done as a simple matrix-vector multiplication. A matrix-vector calculation would not only scale as $\mathcal{O}\left(N^2\right)$ in computation time (excluding Coriolis coupling), but would also take $\mathcal{O}\left(N^2\right)$ reals of memory. Here N is the number of basis functions, which could be as high as 200.000 per Ω state.

[4] About 80% of the system time of a calculation is spent calculating $\hat{H}\Psi$.

3.2.4 Indexing order

The value of the wavefunction depends on several quantum numbers. This means that it can be organized into a 1D vector in several ways. The order matters, because it must be multiplied by a matrices for specific quantum numbers. This multiplication becomes both easier and faster if the wavefunction is sorted to have the right quantum number changing fastest. That way, it is just a series of matrix-vector multiplications for contiguous slices of the array[41].

For example, if the wavefunction depends on $\alpha = 1, 2$ and $\beta = 1, 2, 3$, then applying a matrix that depends on α is easiest if α changes fastest: $\alpha_1\beta_1; \alpha_2\beta_1; \alpha_1\beta_2; \alpha_2\beta_2; \alpha_2\beta_3; \alpha_2\beta_3$. The α-matrix can be multiplied for each β-value in the vector.

This is fast because the data is localized in memory and because standard matrix multiplication can be used, for which optimized code exists.

However, there are matrices to multiply for the other quantum numbers as well. For that purpose, the wavefunction coefficients are reordered before each such multiplication such that the appropriate index changes fastest. This reorganization happens many times, so it is important that it is as efficient as possible. This can be done in $\mathcal{O}(N)$ by using pre-computed indexing vectors.

Another concept which also relates to the order of indexing, is the order of dimensions to iterate in multidimensional matrices. Although the number of operations is identical for row- or column-wise iteration, for reasons including of SIMD (Single Instruction Multiple Data, a type of data-level parallelism by CPUs) and CPU caching, performance is much better when working on contiguous memory. Fortran matrix storage is column-major, so the leftmost index should vary in be the innermost loop.

3.2.5 Coriolis coupling

After calculating the kinetic energy terms, the potential terms, and the rotational terms, the Coriolis coupling terms should be calculated.

This would normally be rather simple, since each Ω state is coupled only to states with adjacent Ω values. But because of that simplicity and loose coupling, this is where the split into processes was made, and so coupling involves inter-process communication. Each Ω state is a separate process. Calculating the interaction involves communicating the wavefunction to (at most) two neighbors[20, appendix].

Practically, each omega value is assigned a left and right neighbour, except the extrema. Communication happens through message passing with MPI. Each time the Hamiltonian is evaluated, each process sends its wavefunction Ψ_Ω to its right neighbour, if any. Each process then reads results from its left neighbour, if any. After processing this data, they then send to left and receive from right.

Note that this also means that all process need to synchronize regularly. This causes some idle time for fast processes. The total time is determined by the slowest iterations. Since $j \geq \Omega$, low Ω states are a bit more expensive, see section 2.6.4 (Restrictions).

The communication for Coriolis coupling this way incurs a performance cost, which is the trade off to save clock time by parallelism. It makes Coriolis coupling one of the more time consuming parts of evaluating the effect of the Hamiltonian, see Figure 30.

3.2.6 Parallelism

There are two benefits to parallelism:

- Less clock time is needed to get results. This fast feedback is very helpful during testing and tuning.
- Memory usage is similar, but the memory is reserved a shorter time, due to the above point. If memory is the limiting factor, this decreases throughput[5].

Note that parallelism does not decrease the CPU time, but rather somewhat increases it.

There are several reasons why using N CPUs does not quite make things N times faster:

- There is added overhead. Within the same machine, communication is fairly fast.
- Total computation time is determined by the slowest process. Since Ω is the projection of j, that means $\Omega \leq j$. Therefore, the angular basis is bigger for lower Ω, and these processes will need more time.
- Some parts have to be serial. For example, file writing had to happen one process at a time.
- In the implementation used, some parts happened identically on each process. This limited the amount of inter-process communication needed, keeping things more simple.
- Some tasks are limited by input/output rather than CPU. Such tasks parallelize differently. This is very limited in this work.

There are two types on parallelism that can be used.

The first is that calculations that differ in good quantum numbers can be performed completely independently, as can different initial values. For example, the calculation for $\nu_i = 1$, $j_i = 5$ with $J = 5$ and positive parity can be calculated in parallel with $j_i = 4$, or $J = 4$. It can also be calculated on another machine on another day, as these calculations are independent.

The second is for quantum states that are coupled, and must be calculated together. Since the Ω states are only coupled to their direct neighbors, they are the best candidate for parallelization, since then processes only need to communicate with neighbors. To calculate j or ν states in parallel, each thread would need to communicate with all others. Therefore only Ω states were calculated in parallel.

Each process does very similar computations but at different Ω. Every time information must be exchanged, the processes communicate with their neighbors by sending and receiving messages. This communication is primarily necessary while evaluating $\hat{H}\Psi$, for the Coriolis coupling term. But it also happens at other places, e.g. to sum Ω states when calculating the output probabilities. This method is one version of the approach proposed by Goldfield and Gray[42].

[5]The machines used have sufficient memory, so this was not actually the case.

3.2.7 Ω states

Of course, many of the parameters strongly affect performance. This is discussed in other parts of this document. However, decreasing Ω_{max} is worth highlighting as a future improvement. Neglecting some Ω states was a large speed boost that was made too late, after most computations were done.

Ω states are only coupled to neighboring states ($\Omega \pm 1$, [20]), and initially, Ω does not exceed the rotational level. Furthermore, Ω-resolved results are not of interest in this work, since they are not discernible in signals from space.

This means high Ω states take long to be occupied to any noticeable extent for the rotational levels of interest (for $j_{init} < J$). This is visible for an actual computation by comparing Figure 10 and Figure 11.

Indeed, as seen in Figure 9, in some cases three quarter of the Ω states can be neglected, and the final result (cross sections) are hardly affected even for large rotational transitions.

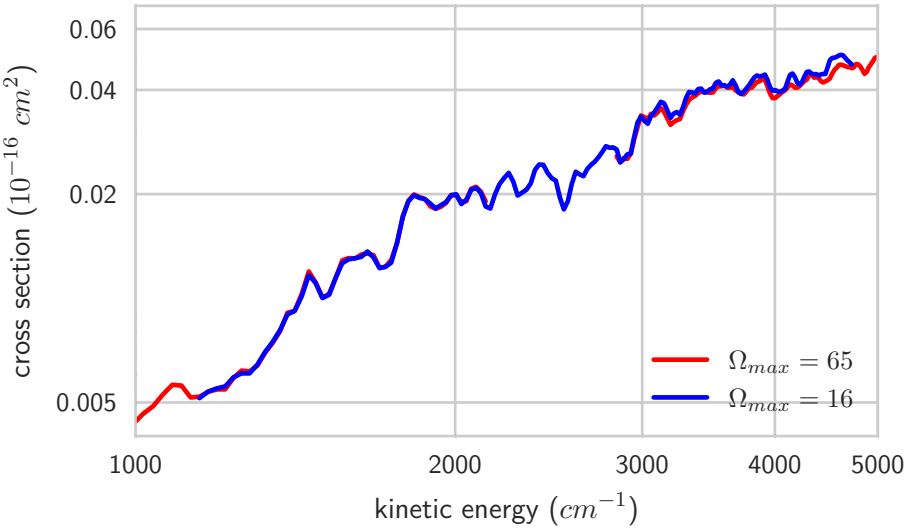


Figure 9: The effect of Ω_{max} on the cross section. Ω_{max} is the main difference, but not the only one. J differs per line segment, increasing with energy from 30 to 65. The transition is $\nu = 1, j = 0$ to $\nu = 0, j = 12$.

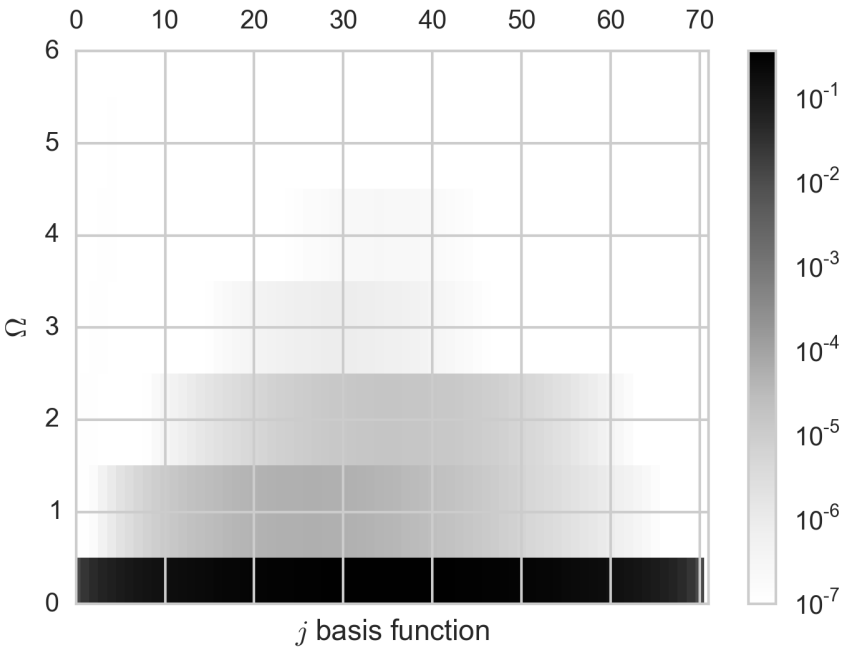


Figure 10: Wave function occupation summed over r and R coordinates for 100 snapshots. Calculation performed at $J = 50$ with $j_i = \Omega_i = 0$. The figure shows that many Ω states do not contribute.

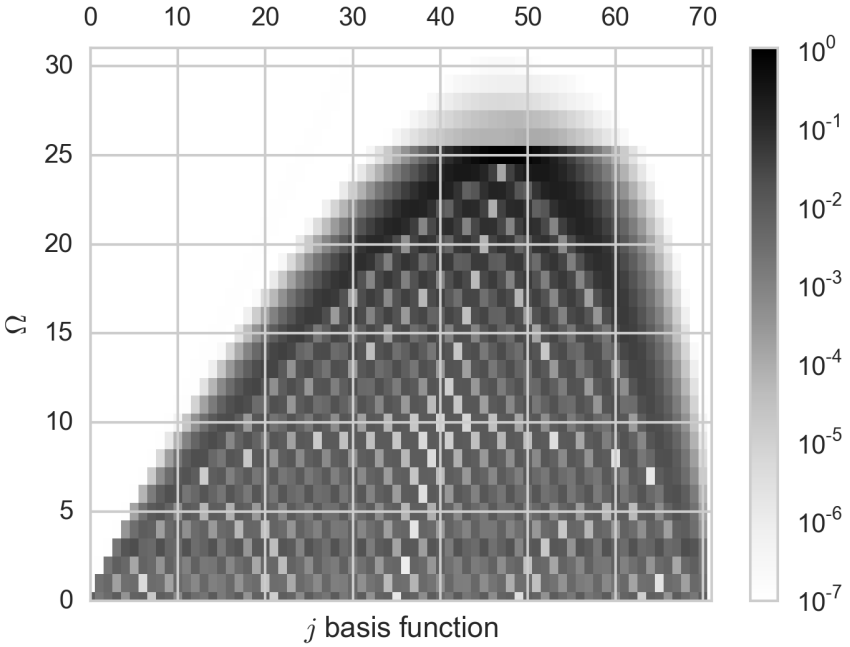


Figure 11: Wave function occupation summed over r and R coordinates for 100 snapshots. Calculation performed at $J = 50$ with $j_i = \Omega_i = 25$.

3.2.8 Miscellaneous

- Possibly surprisingly, turning off threading in BLAS improved performance. Processor cores were already at full load due to parallel Ω states and J jobs. The extra context switching from threading in matrix operations seems to have mostly caused overhead.
- Any parameters that make the results accurate over a longer range -most notably the wavepacket spread- affects performance by needing fewer calculations to get probabilities for all relevant energies.

For the actual time usage, see Section 4.4.

4 Results

4.1 Propagation

The propagation of the wavepacket as a function of timestep can be visualized if projected on two of the four dimensions.

This has been done for one example propagation in the r-R plane. This animation is included as an attachment, see section 6.1 (Animation).

The remaining norm, as well as the total absorption, can give an idea of the progress of the interaction. This is visible in Figure 12, with non-reaction probability as an indication of absorption.

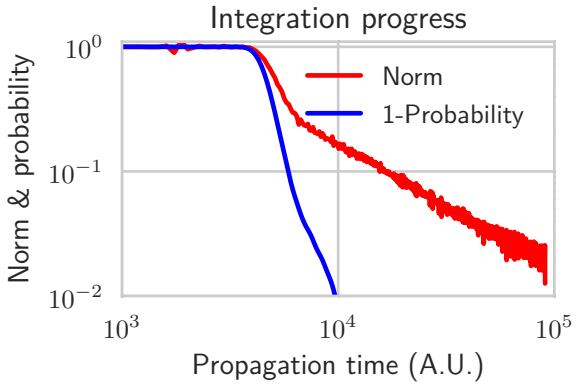


Figure 12: The norm remaining on the grid, and complement of the non-reaction probability, as a function of integration time (logarithmic). Three ranges are are visible: Initially, the wavepacket is on the way to the interaction region and the reflection on the way back, so no absorption happens. After that, the part of the wavefunction that was immediately reflected is absorbed, resulting in a brief, steep decline in the norm. Finally, the parts of the wavefunction that were trapped in the interaction region escape and are absorbed approximately mononomially.

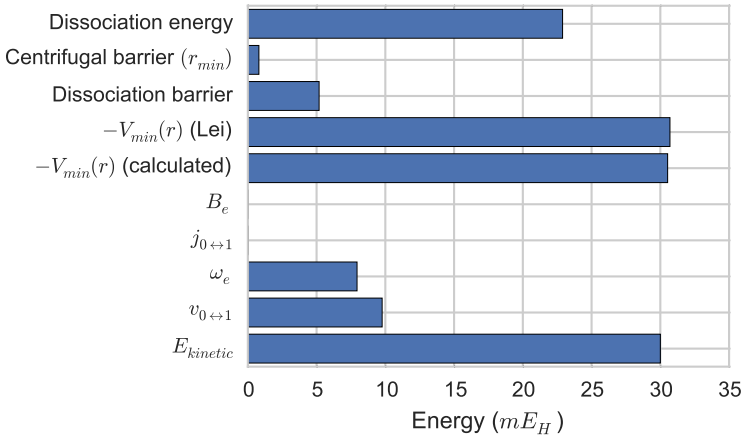


Figure 13: An example of the energy distribution at $J = 12$, $\nu = 1$, $j = 1$ and the highest kinetic energy used, to give an idea of order of magnitude.

4.2 State-resolved probabilities

Using time-dependent wavepacket methods, each calculation has as pure initial state and produces data for all output channels. A range of energies is computed each time, depending on initial kinetic energy and the spread of the wavepacket. Ω is not of interest in this work, so the output consists of ν and j channels at a range of energies with Ω contributions summed.

These channels have been visualized as a function of energy by giving each j level a color, and plotting all channels on stacked on top of each other. This way, each rainbow corresponds to a ν level, with each color a j sublevel (the highest j levels are gray). See, e.g., Figure 15, Figure 17, Figure 18, and the front page.

A useful way to detect inaccurate probabilities is to sum probabilities from all output channels for each energy. This should be close to unity, because one of the channels must be occupied at any energy, see section 14 (An example of integral norm deviations for all the partial waves for at $\nu_i = 1$, $j_i = 5$, and kinetic energy $5.0mE_H$, and how the usable energy range is extracted from this.). If the norm deviates too much, the probabilities for that energy are discarded. A reason this can happen is because the norm of the wavefunction in energy space is not sufficient to calculate probabilities. There can be other reasons, like including too few channels, but those have been checked for.

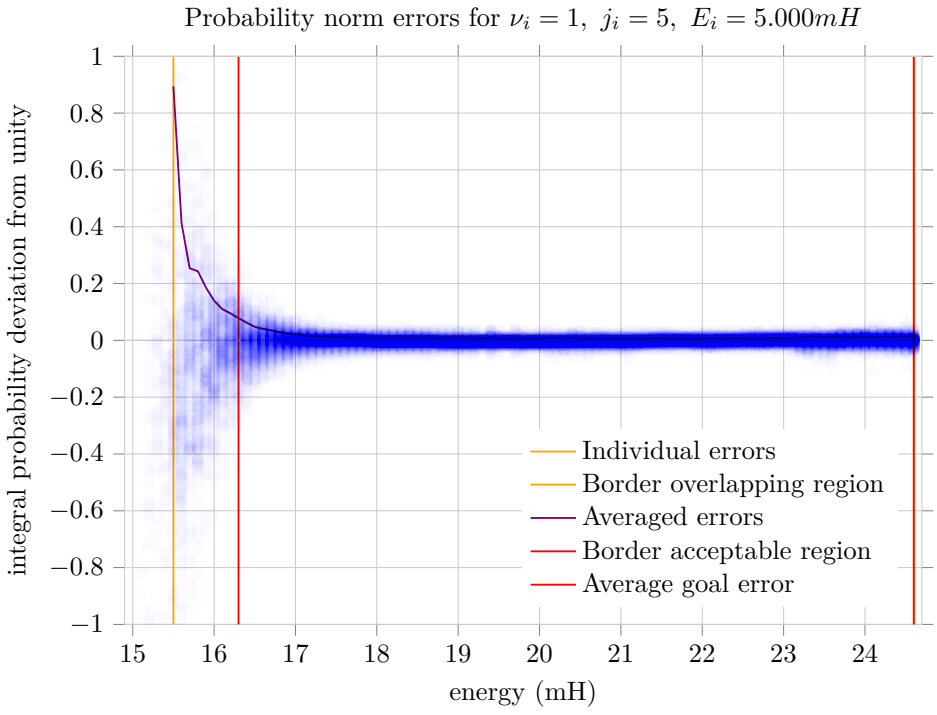


Figure 14: An example of integral norm deviations for all the partial waves for at $\nu_i = 1$, $j_i = 5$, and kinetic energy $5.0mE_H$, and how the usable energy range is extracted from this.

Another way to check is to create jobs at different initial energy distributions whose tails have some overlap. The probabilities in the overlapping region should then be the same for both jobs. This is done in later figures.

4.2.1 Effect of J

In Figure 15, it is seen that reactivity goes up as a function of energy and down as a function of J. This effect is strong: as energy increases, the highest occupied

J state also rapidly increases.

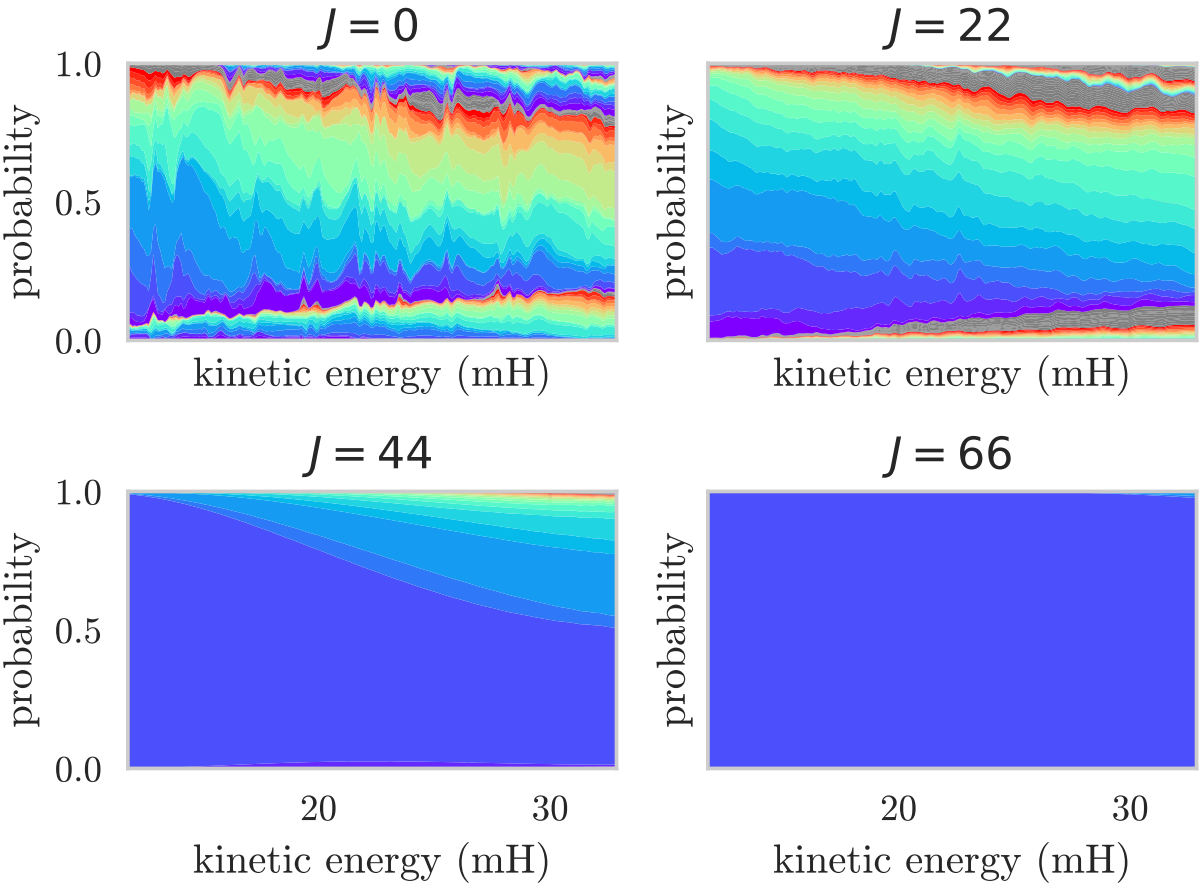


Figure 15: Transition probability as a function of initial kinetic energy for all the ν, j channels, as explained in section 4.2 (State-resolved probabilities). The initial state is $\nu = 1$, $j = 2$, $\Omega = 0$, and positive parity. Each rainbow is a vibrational final state, with rotation level varying from purple (low) to gray (high).

This happens because a large total angular momentum at fixed CO angular momentum j indicates a large orbital angular momentum. This in turn means that the impact parameter is large and that the closest approach between the particles is large (or at least, would be large in the absence of interaction).

This also explains the steep decrease in transition probability for higher J states, which is visible in 16.

Note that J is the total angular momentum, having contributions from the angular momentum of CO itself, and the rotation of H and CO around each other. However, one cannot sum these magnitudes, as they are vector quantities.

At near-head-on collisions, CO rotation improves reactivity. At high total angular momentum, reactivity is lower if j is small, because in that case nearly

all the angular momentum is in the centrifugal barrier (see Equation (2)).

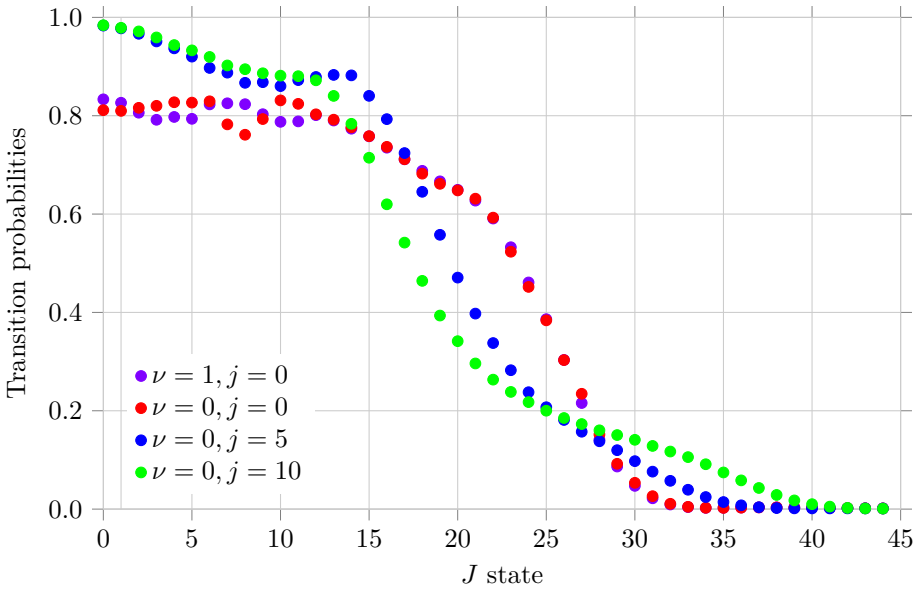


Figure 16: An example of the effect of angular momentum (J) on transitions from initial state $\nu = 1$, $j = 1$, and kinetic energy $4.6mE_H$. At low J, representing near-head-on impact if j is small, the barrier has little effect. At high J, representing glancing collisions if j is small, the distance is too high.

4.2.2 Effect of Ω

The effect of both J and Ω on reactivity is also shown in Figure 17. This also shows the effect of energy and Ω, and shows individual output states.

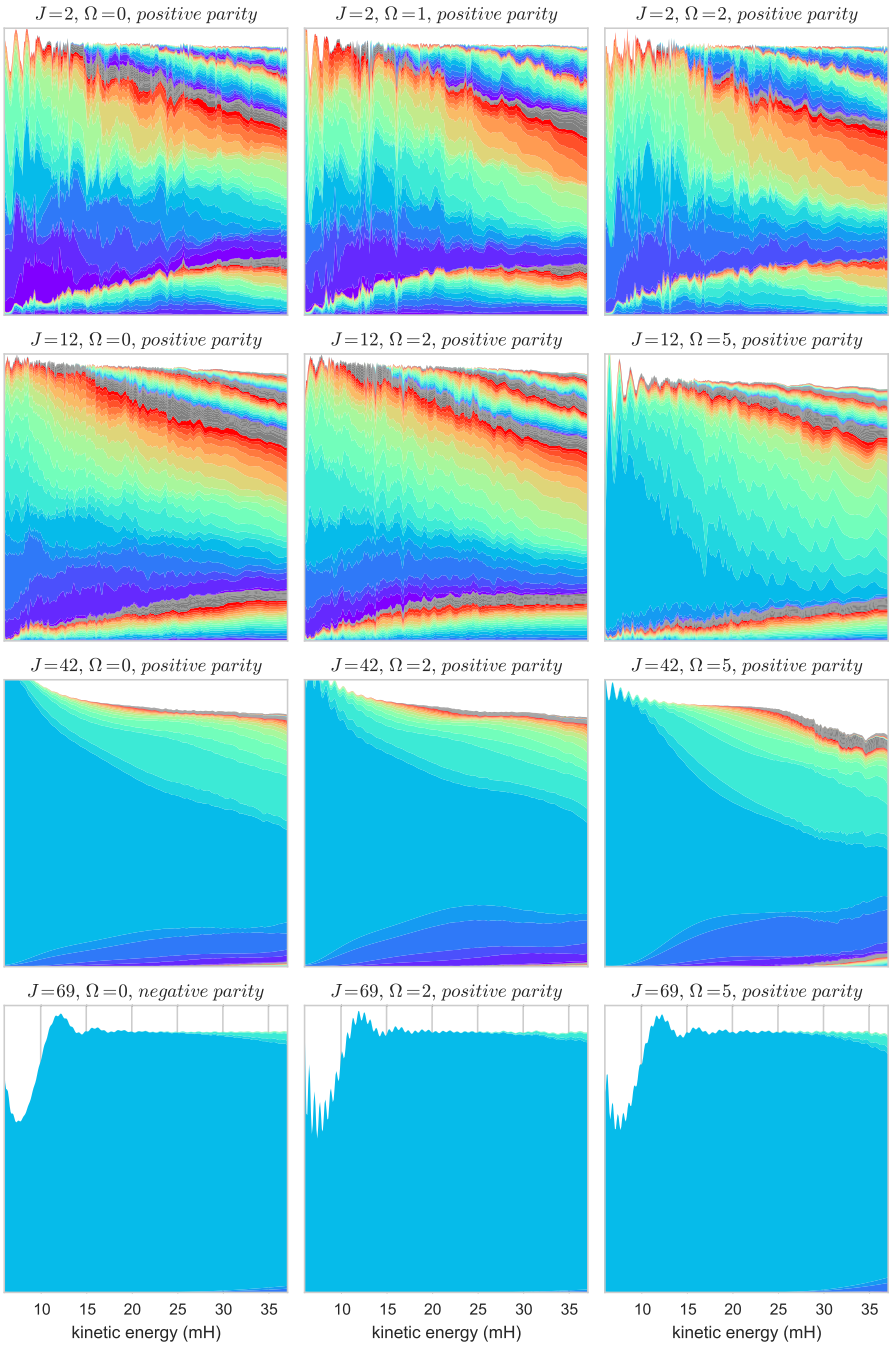


Figure 17: Transition probability for various partial waves. Initial kinetic energy is $30mE_H$, $\nu_i = 1$, and $j_i = 5$ for each job. It it seen that results are inaccurate at too low or high energy, as norm departs from unity. It is also seen that reactivity goes up with energy, and down with J.

It appears that lower Ω states induce more rotational transition, if all else is

equal, especially for the second row, where $J = 12$. This is more obvious in 19.

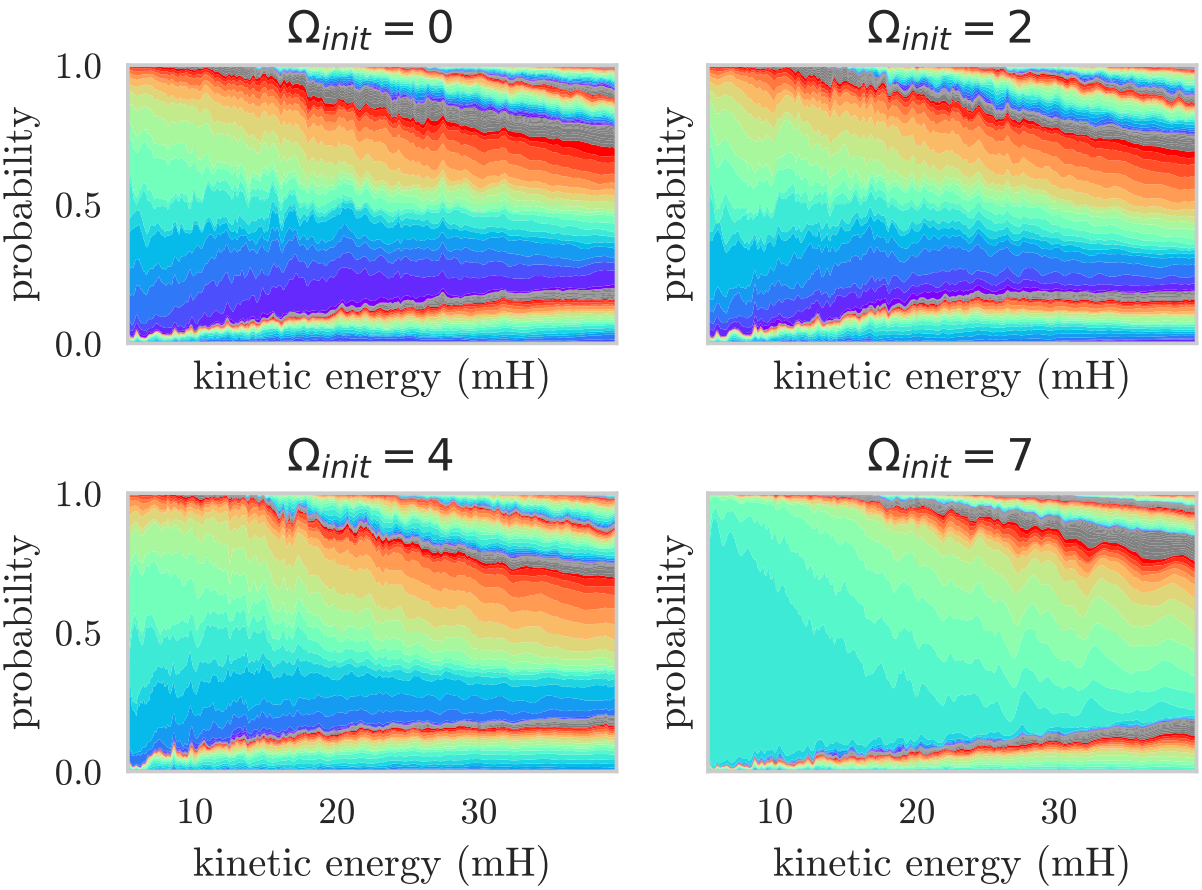


Figure 18: Transition probability as a function of initial kinetic energy for all the ν, j channels, as explained in section 4.2 (State-resolved probabilities). The initial state is $\nu = 1$, $j = 7$, $J = 8$, and positive parity. Each rainbow is a vibrational final state, with rotation level varying from purple (low) to gray (high).

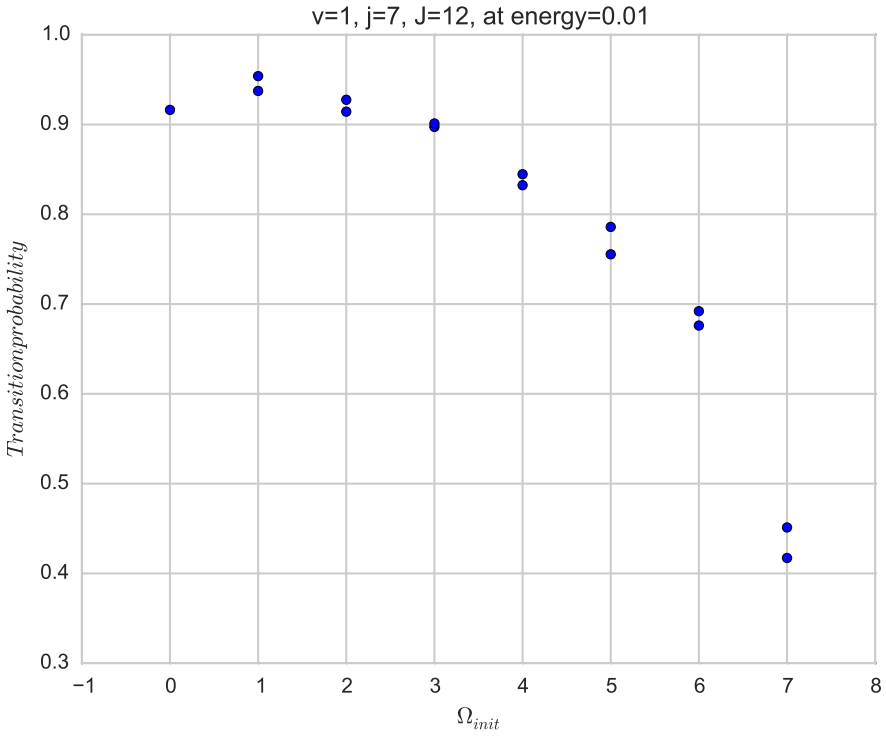


Figure 19: Transition probability as a function of Ω.

As mentioned in section 2.3.1 (Embedding), low Ω means the plane of rotation contains the intermolecular axis R and the angular momentum is perpendicular to R. This may be a more effective configuration to transfer angular momentum than if rotation were around the intermolecular axis R and the angular momentum is parallel to R.

4.2.3 Effect of Δj

A number of effects are visible for j transitions in Figure 20.

The figure shows the transitions in angular momentum of CO, within a given vibrational level. It is clear that transitions of $|\Delta j| = 2n$ are more likely than $|\Delta j| = 1 + 2n$, especially for small n. This is recognizable from homonuclear diatoms, where odd Δj is not allowed. Since the masses of carbon and oxygen are similar, it follows a weakened version of the homonuclear pattern. This agrees with [19, fig 3.3-3.5].

Higher j levels are more occupied, which is likely due to degeneracy. This is more pronounced when starting at lower j levels (not shown here).

It is also evident that large changes in j are less likely, probably because of the large amount of angular momentum that would need to be transferred. This was also found by [6].

Finally, very high levels of j are hardly reached, because they take a large amount of energy, which may not be available or may go into other states.

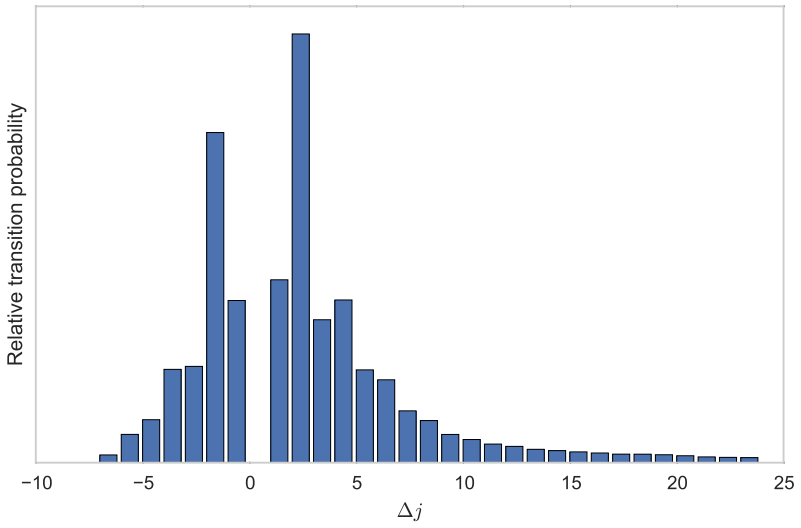


Figure 20: Squared transition matrix values as a function of Δ j. Initial kinetic energy 20 mE_H, $\mu_i = \mu_f = 1$ and $j_i = 7$, summed over J, parity, and Ω including degeneracy. $\Delta j = 0$ is not shown.

4.2.4 Effect of energy

The effect of the total energy on quenching is shown in Figure 21.

The spread in probability is seen to be large, especially for medium energies. This means that reactivity is not dictated by just J and total initial energy.

Note that the lower quenching at high energies is not due to lower reactivity, but rather due to the higher probability of transitioning to a higher state. The $\mu = 0$ states have low quenching because there are few lower states. This can be seen in Figure 22.

The figure also shows that rotational energy differences are minor compared to vibrational. Differences between jobs are slightly smaller than a vibrational transition. Since the energy differences between jobs were chosen to that energy ranges overlap slightly, this suggests that the kinetic energy spread is roughly half a vibrational level.

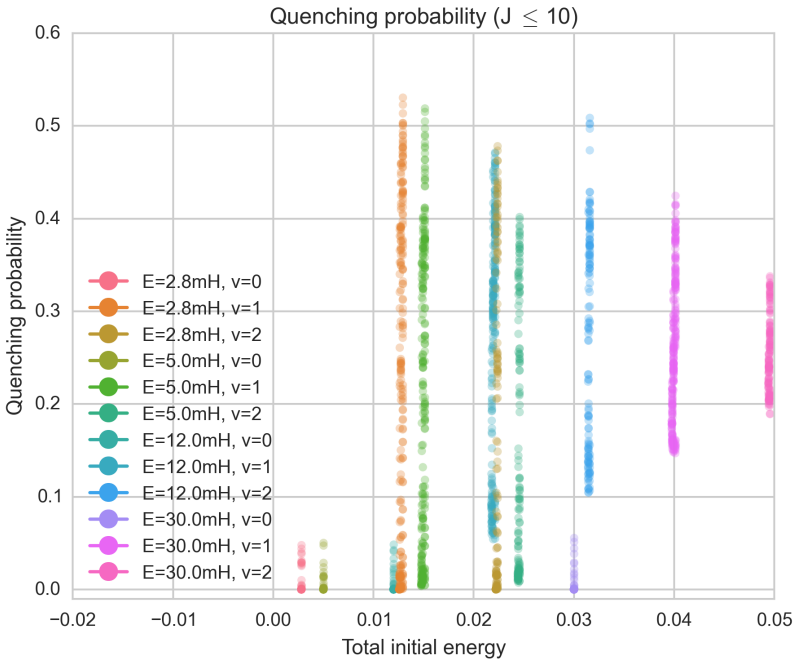


Figure 21: The probability of going from the initial state to a lower rovibrational state, shown against the total energy. This is done at $J \leq 10$ and at the initial kinetic energy, indicated by color.

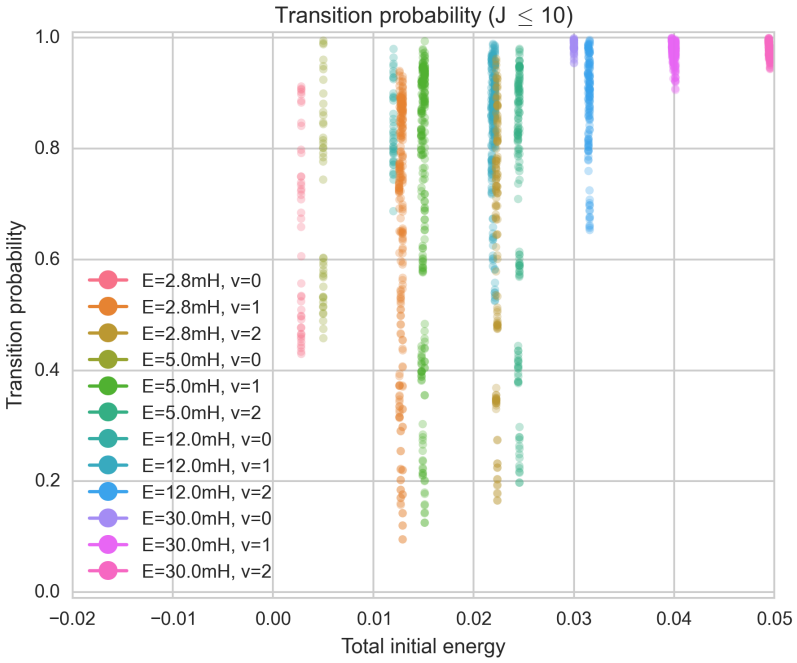


Figure 22: The probability of of not staying in the initial rovibrational state, shown against the total energy. Similar to Figure 21.

4.3 Cross sections

4.3.1 Transform to cross sections

State-to-state cross sections are calculated from transition probabilities according to Equation (17). In the present work, the output of the calculation has already been summed over the outgoing Ω states, so the equation simplifies a little:

$$\sigma_{vj\to v'j'}(E_{vj}) = \frac{\pi}{(2j+1)\,k_{vj}^2} \sum_{J=0}^{J_{max}} (2J+1) \sum_{p}^{+,-} \sum_{\Omega=0\vee 1}^{\min(j,J)} |\underline{\underline{\boldsymbol{T}}}^{J,p,\Omega}_{vj\to v'j'}(E_{vj})|^2. \quad (22)$$

4.3.2 Probability range selection by wavefunction spread

In going from probabilities to cross sections, the omega, parity, and J results are summed over, leaving one result per initial rovibrational state with a range of kinetic energies.

This introduces a problem. Each individual run has a range of energies. Of course, results need to be present at an energy for every run in order to combine. But probabilities may not be reliable for all energies, largely due to the wavefunction not having enough amplitude at energies far from the mean kinetic energy.

One attempted solution was to base which probabilities are used on the spread of the wavepacket. The spread in the R coordinate is a parameter, and is related to the spread in energy, as described in section 2.7.3 (Initial R wavefunction). The jobs for a given rovibrational state all have the same energy and spread, so this is easy. A result is shown in Figure 24.

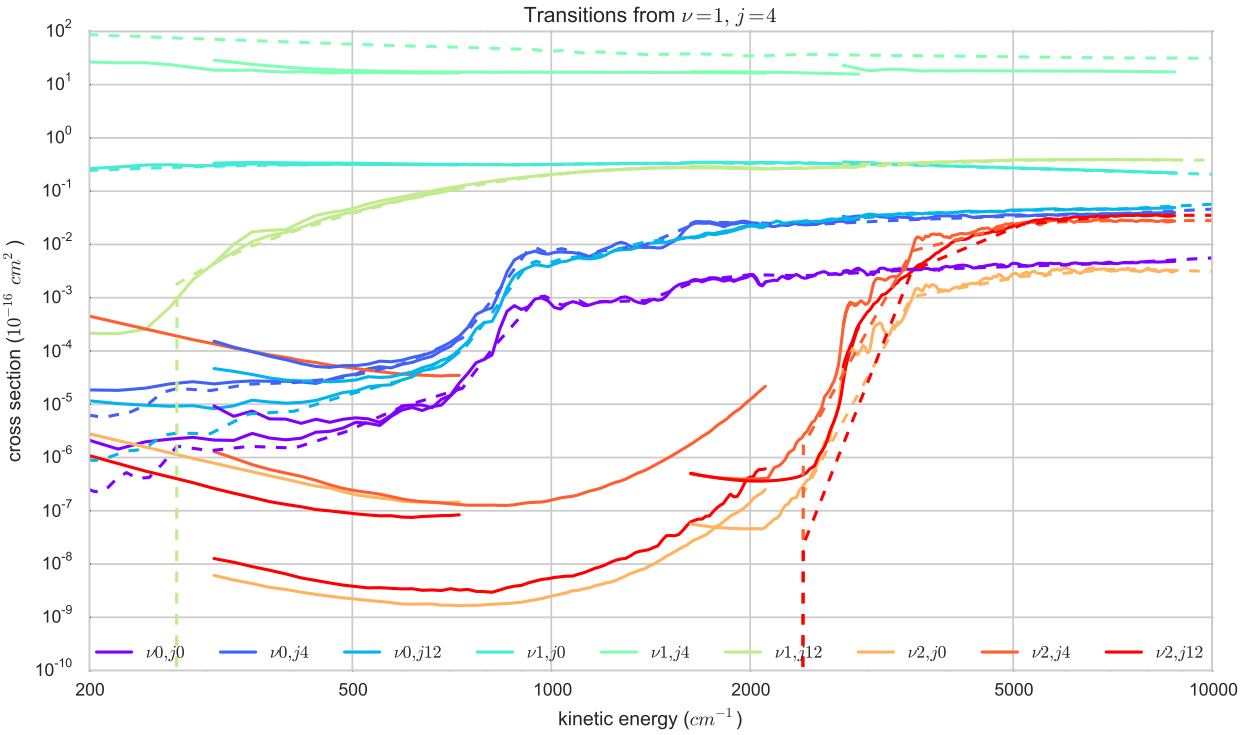


Figure 23: All the cross sections output by the program as continuous lines, compared to results by Song[19] in dashed lines. The cross section for the initial state, as well as the results for closed channels, are inaccurate as discussed in section 4.3.4 (Convergence of cross sections). These will be omitted from future figures.

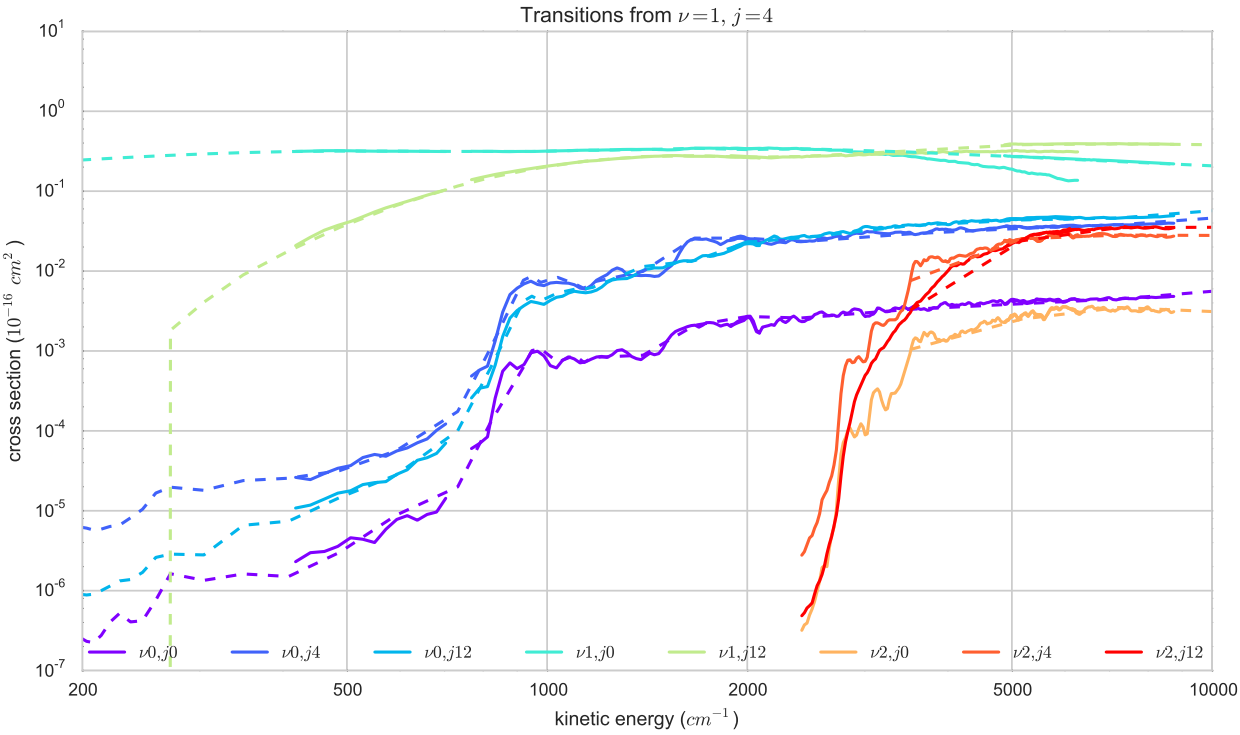


Figure 24: Cross sections, with ranges selected by cutting probabilities off at $0.2\sigma_E$ at the low end, and $1.0\sigma_E$ at the high end. Results by Song[19] in dashed lines. Compare to Figure 25, which combines probabilities based on probability norm.

Some observations can be made:

- The cutoff of 0.2 kinetic energy standard deviation on the low end, and 1.0 on the high end, was chosen to obtain reasonable results. The physical or numerical reason why 0.2 and 1.0 would give reasonable results are not clear.
- This methods cuts of quite some probabilities that seem reliable (by norm, by overlap with other results, and by comparison to earlier work).
- There are still a few cases in the figure where the range seems too large, e.g. the excitation to $\nu = 2$ around $3000cm^{-1}$.

4.3.3 Probability range selection by probability norm

An alternative way to attempt to determine which probabilities are reliable, is by seeing how much the norm deviates from unity at the energy. Results where the total of all output channels is not close to 100% correspond to the wavefunction not being conserved, and are not reliable. Unfortunately the reverse is not necessarily true - a conserved wavefunction is not a guarantee of reliability, merely a minimum requirement.

This method is visualized in section 14 (An example of integral norm deviations for all the partial waves for at $\nu_i = 1$, $j_i = 5$, and kinetic energy $5.0mE_H$, and how the usable energy range is extracted from this.), and the result is in Figure 25.

It is more complex than using the spread of the wavefunction. The selected range has to be continuous even if there are narrow fluctuations. A maximum average norm deviation is set as a goal, and the largest continuous range with that deviation is computed. This was initially done per partial wave, but this was changed to operating on the average deviation of all partial waves in a rovibrational state and energy to be more robust.

Do note that the exact probability norm deviation at which the cutoff happened was tuned by hand for each rovibrational state and initial energy.

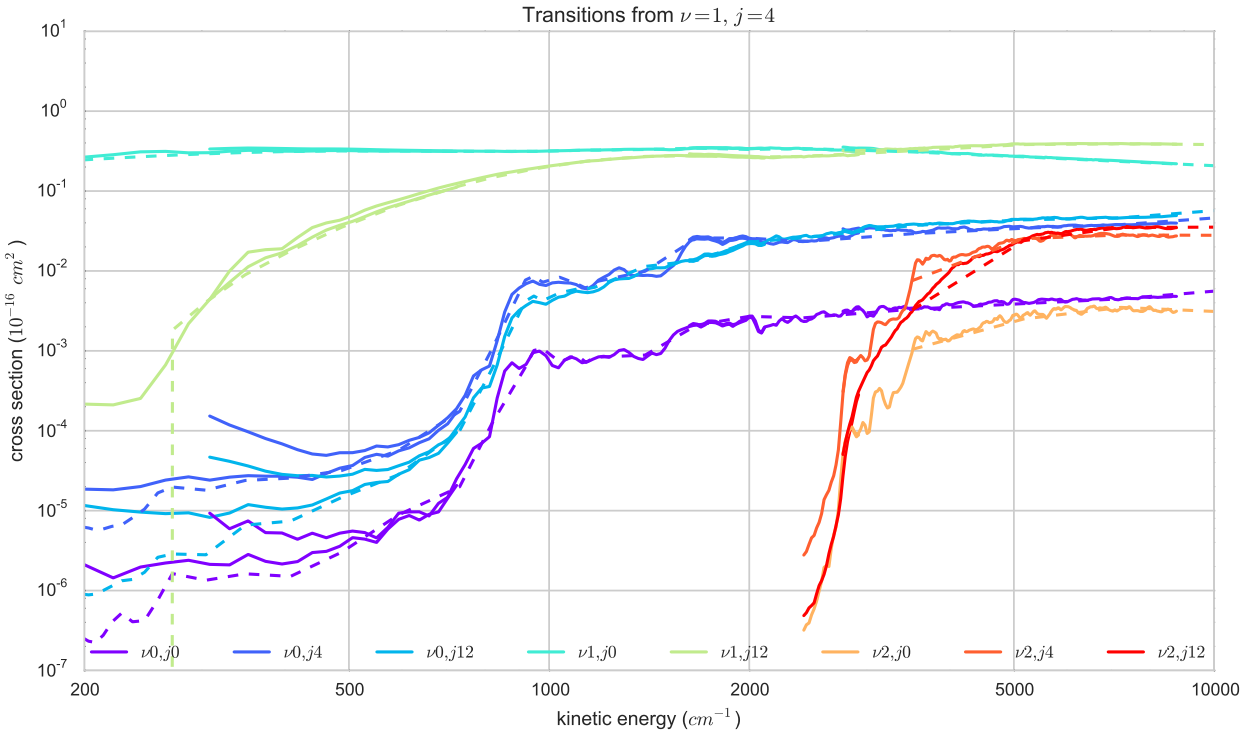


Figure 25: Cross sections, with ranges selected by choosing the range of probabilities with lowest average error. Compare to Figure 24, which combines probabilities based on wavefunction spread.

- This process throws away less of the seemingly reliable parts of the results.
- There are still quite a few cases in the figure where the range seems too large, e.g. again the the excitation to $\nu = 2$ around $3 \cdot 10^3 cm^{-1}$. This is even more pronounced if the cutoff deviation is increased slightly, and is more visible in figures on later pages. It is unclear why these probabilities, so close to the center of the wavefunction, behave erratically.
- Selecting the error cutoffs based on figure Figure 14 is a fairly subjective

process.

- Overall, both a more automatic process and one with fewer errors remains desirable.

4.3.4 Convergence of cross sections

The cross sections, before filtering, are visible in Figure 23. The dashed line represents results from Song [19] if available, which were computed with time-independent methods.

The probability of remaining in the initial state is consistently about a factor two lower in this work. This is likely the case because elastic cross sections take substantially more J states to converge than inelastic ones. As such, the cross sections for the initial state are probably not converged, and they are removed in further figures as they are not the main interest of this work.

There is output for vibrational excitations at low energy, since the program produces numbers for the whole energy range. But such an excitation at low energy is chemically impossible. It is indeed seen that these results look unreasonable, and they have been discarded after Figure 23.

4.3.5 Comparison to earlier work

For the results, see Figure 25, Figure 26 and Figure 27, which are filtered as described in section 4.3.4 (Convergence of cross sections).

In most cases where both are available, the results from this work match very well with those obtained by [19]. The increase in reactivity around $10^3 cm^{-1}$ for $v_i = 1$ is particularly interesting, and the good match there is a good sign for the reliability of both these result and those by Song *et al.*

The goal of extending results to higher energies was not yet reached. Given the infrastructure established though, it is likely to require limited human effort, but a substantial amount of computation time. These computational requirements go up quickly, as expensive higher J states quickly become necessary.

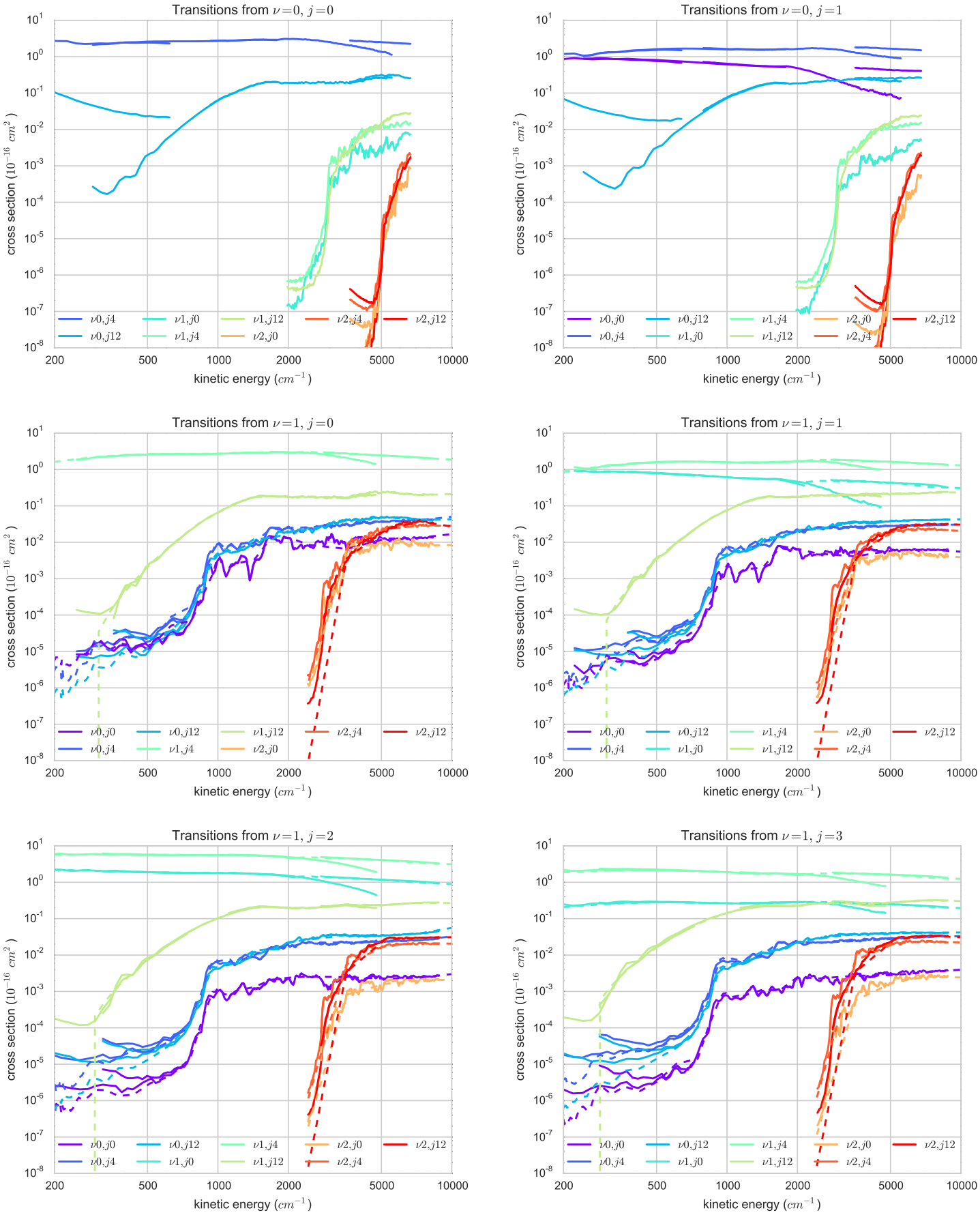
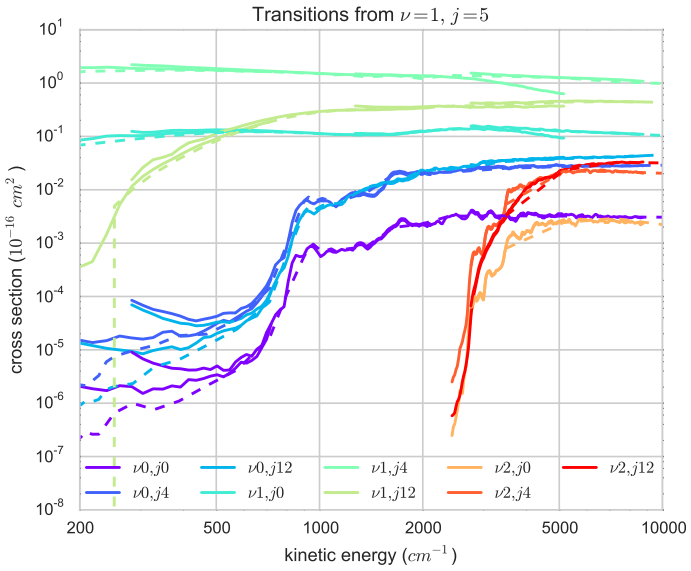
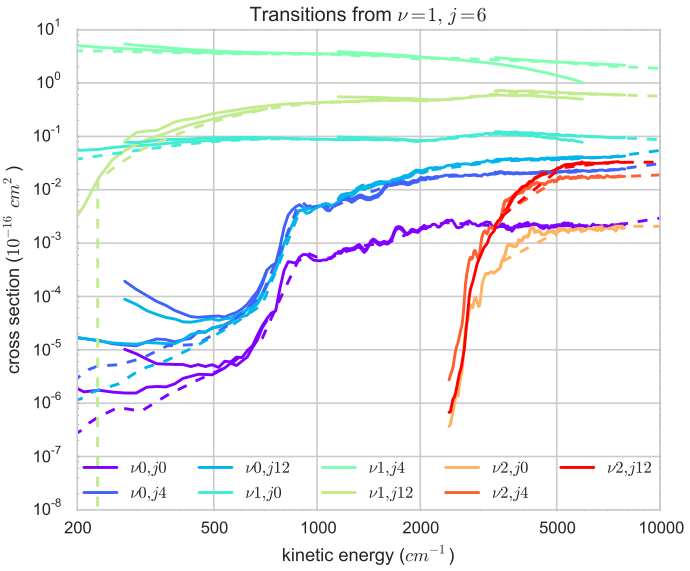
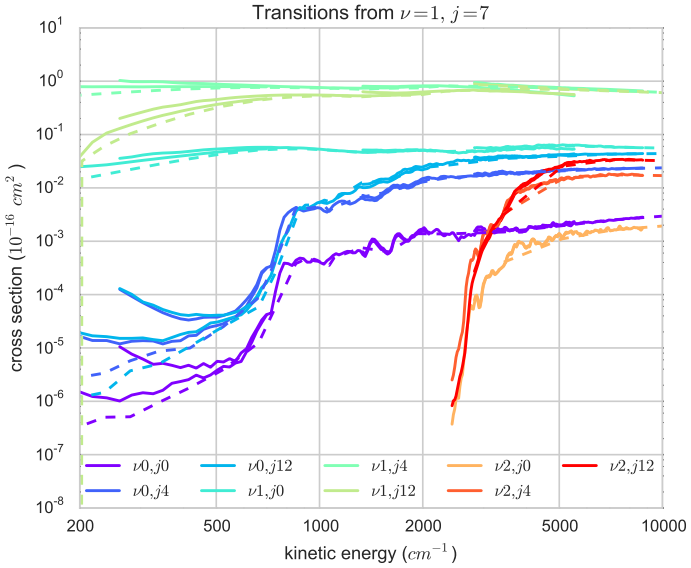
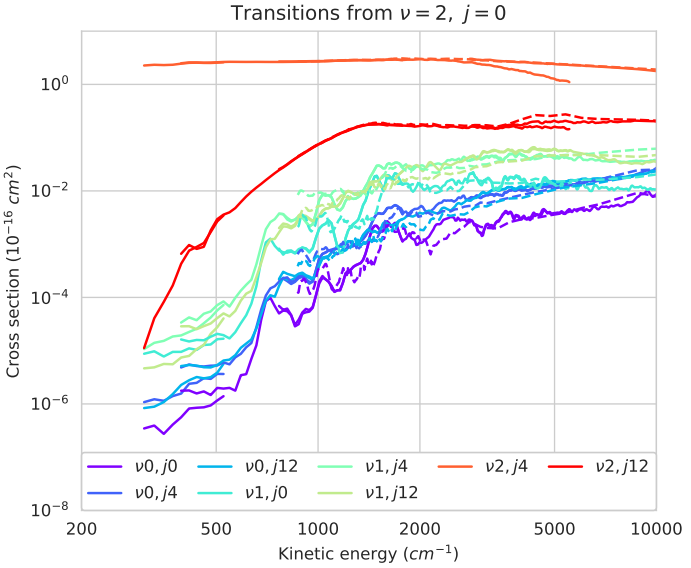


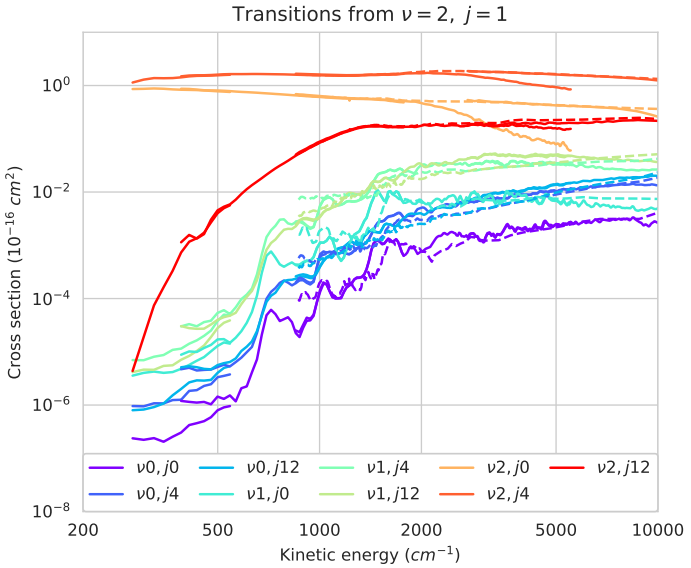
Figure 26: Cross section results per initial state. Continued in Figure 27.











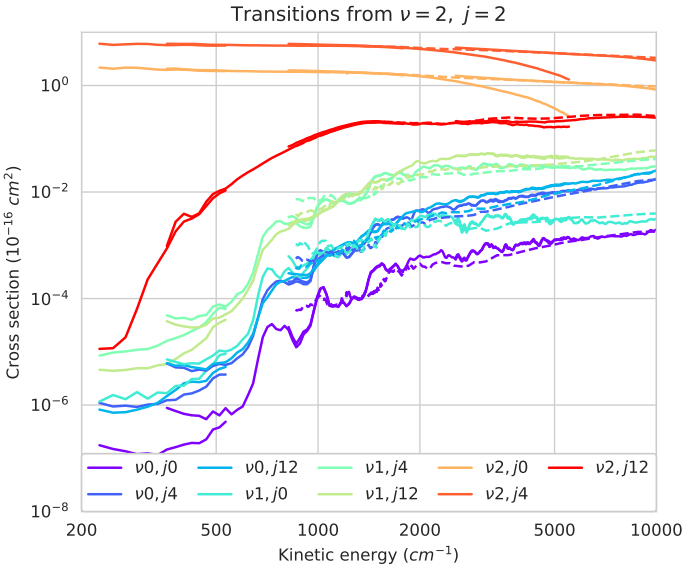


Figure 27: Cross section results per initial state.

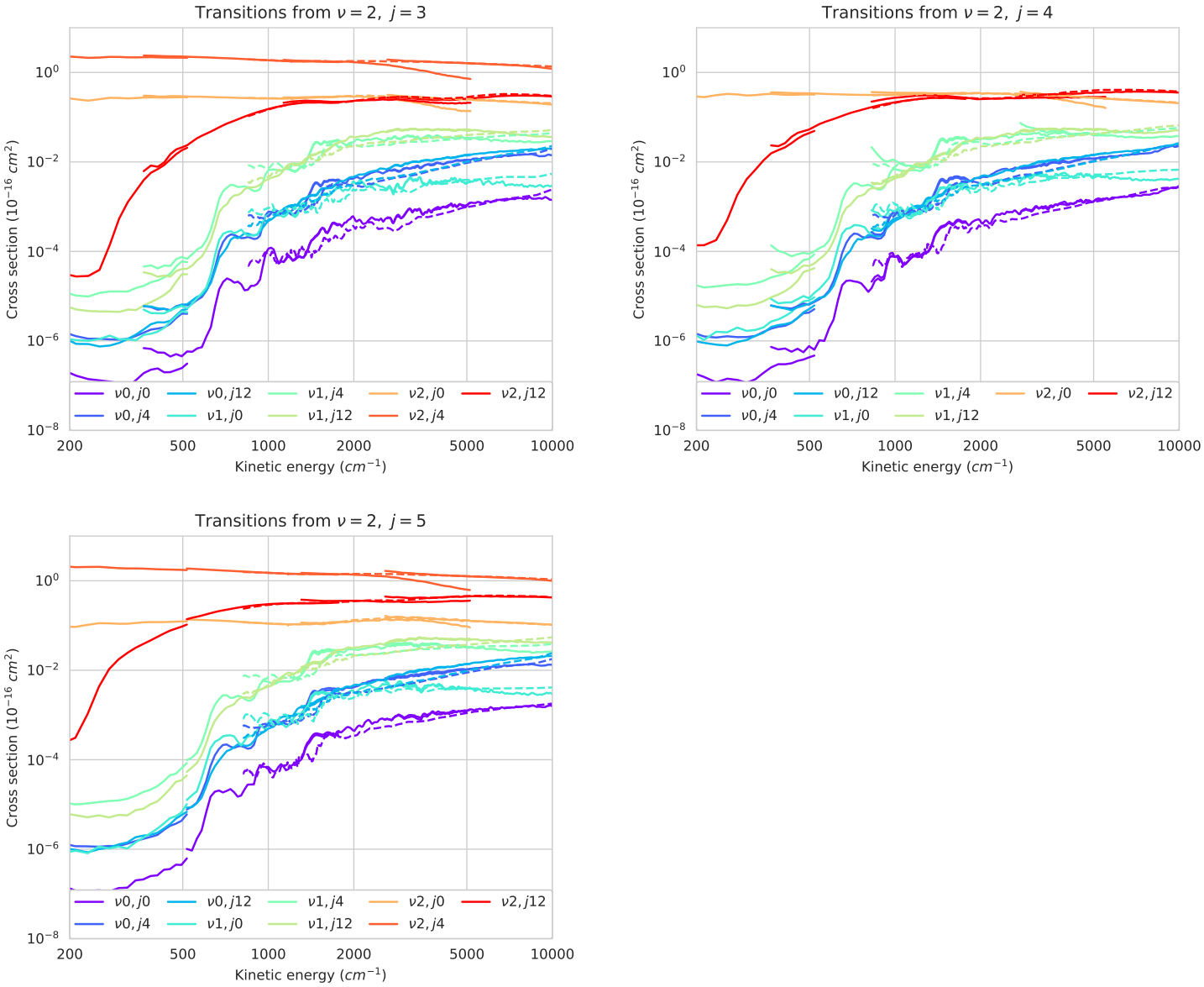


Figure 28: Cross section results per initial state.

Figure 29 compares individual transitions, based on Song [19, fig 4.1].

At medium and high energy, agreement is good. The shape agrees well, and some of the small peaks are reproduced, to an extent.

At the lowest energies, the results are less ideal. This is somewhat expected, as this is where the time-independent method excels, while time-dependent the wavepacket method is stronger at higher energies.

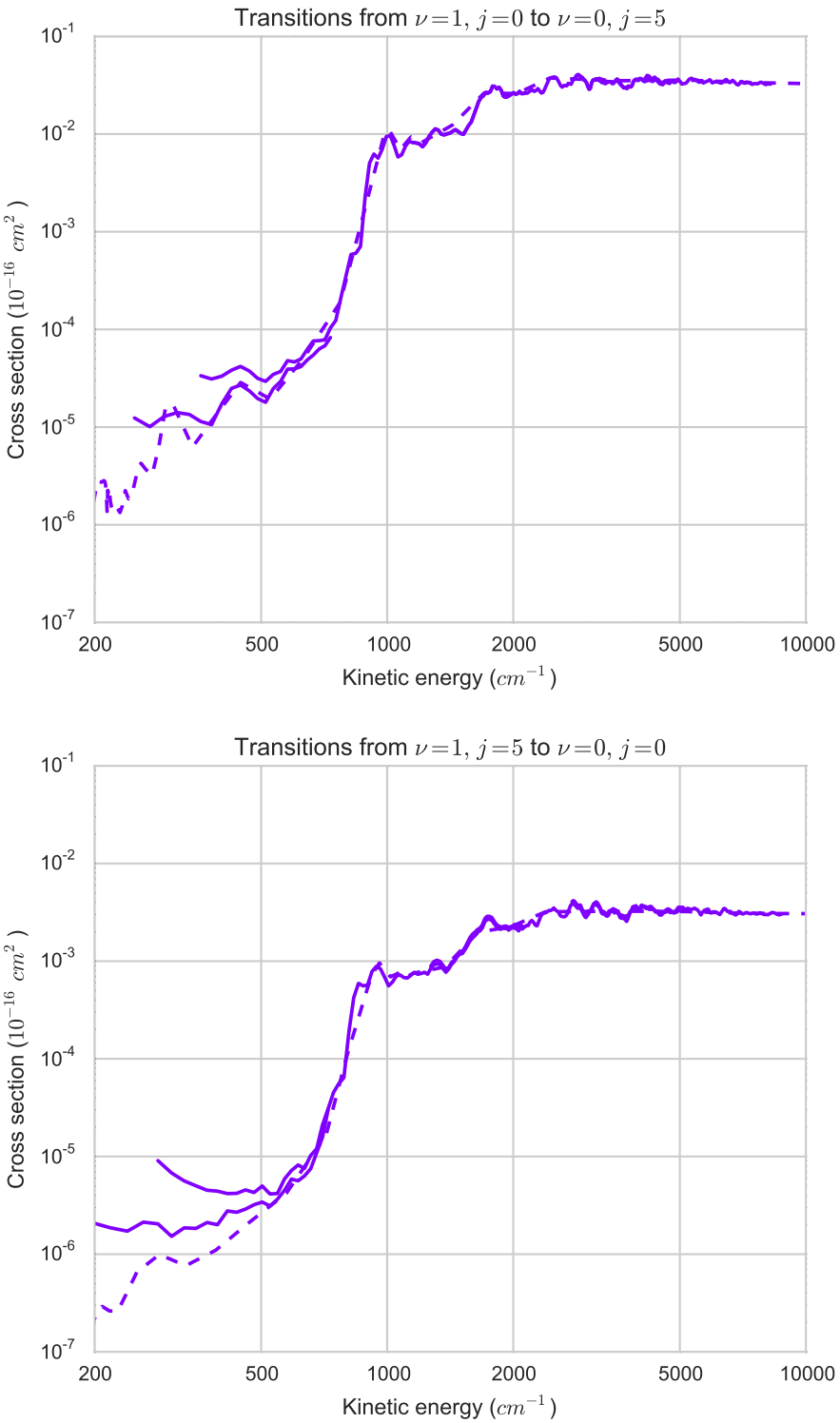


Figure 29: Cross sections for individual transitions, current work in continuous line and [19, fig 4.1] dashed.

4.4 Performance

Since accuracy and performance were improved simultaneously, it is difficult to determine precise performance gains. However, even with this reservation, a rough estimation of a lower bound may be of use. Careful tuning of parameters, together with optimizations added to the code, resulted in calculations being about a factor 8 faster, as seen in section 3.2 (Performance). At the same time, accuracy was improved, this being judged by the state-to-state probabilities summing to 1.

To get an idea of where most computation time is spent, Figure 30 contains a visual representation averaged over a number of typical jobs.

From the figure, it seems that some computation time can be saved by storing the reduced potential grid for specific potential and grid parameters. This step depends largely on the speed of the potential, which is evaluated on the order of 50.000 times.

Most time is spent on integration, which is not reusable, and is already highly optimized. It was indirectly decreased substantially by increasing the size of timesteps, as described in section 2.10.3 (Range of eigenvalues).

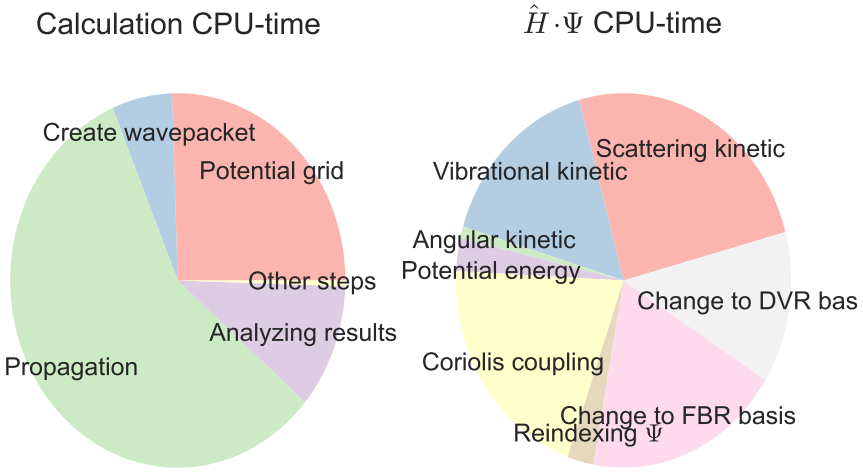


Figure 30: The breakup of CPU time use. Left is the overall calculation, right is the computing of the effect of $\hat{H}$ of Ψ, which is 80% of the propagation step on the left. Results depend on physical and other parameters, so this should only be used quantitatively. For example, the time spend on integration is proportional to the number of iterations. For specialized results, the code was expanded to include these benchmarks for each run if `_comp_cpp_defs` contains `TIMING`.

The performance could possibly be improved even further by reducing the basis. In this work, each r, R grid point had fixed number of j and Ω states. By adding some code, it should be possibly to trim ϑ basis functions per r-point. This would be made much more convenient by using vibrational and rotational eigenfunctions for these coordinates. One could then trim those rovibrational states that have energies too high to be accessible.

5 Conclusion

5.1 Accuracy

Results from this work match well those from earlier work well, especially for energies of around $10^3 cm^{-1}$ and above. The difference between the results is on average 31.3% for inelastic cross sections to all final states with $v \leq 3$ and $j \leq 10$. But if only energies over $10^3 cm^{-1}$ are considered, the difference is only 7.3% on average. At low energies, there is a tendency for results from this work to be higher than those from earlier work, but at higher energies there is no bias.

Reaction rates for the earlier study have been compared to experimental data for the transitions available, also to good results. It seems probably that the results are indeed accurate.

The method is promising for higher energies, and the infrastructure has been created. However, more computation would be needed, for which there would not be any earlier results to compare.

5.2 Methods

The code has been improved and expanded substantially: several problems were fixed, readability and robustness of the code were improved, bookkeeping code to handle many calculations was added, results are automatically extracted from various log and output files, there are many plotting routines and many parameters have (semi-)automatic values and increased checks.

5.3 Future

With some processing, the results should be useful for application in astronomy, as described in 1.2. It is also good to know that earlier results (section 1.3 (Earlier work)) match well with the ones obtained here in the intermediate energy region.

The project has potential to further extend the range of H-CO results. The method applies well to higher states and energies so far, and can perhaps be successful for a wider range. The main obstacle is the great number of J states needed as energy increases, but this is physical problem and not one of the method. It is possible to use approximations to make this take less time to compute in the future.

The approach may also be useful for other atom-diatom non-reactive scattering problems, or even reactive ones. The Fortran code supports reactive scattering. Progress on parameter tuning and Python code was directed towards non-reactive scattering, but many aspects are analogous. The time-independent approach at lower energies and this wavepacket approach at higher energies complement each other well, and serve as a mutual validation.

5.4 Reflection

There were more obstacles than were expected. As such, the project required more time and effort.

Fortunately the project was successful in the end. It agrees with earlier results and improved the code and parameters for reuse.

As a personal lesson, I got a little carried away with the computational side of things[6].

5.5 Acknowledgements

I would like to thank prof. dr. ir. Gerrit C. Groenenboom and prof. dr. Anthony J. H. M. Meijer for their advice, wisdom and support. The use of the scattering code by dr. Meijer and the potential energy surface by dr. 宋磊 were essential to this work.

6 Attachments

6.1 Animation

An animation of the wavefunction propagation is attached. It shows the square of the wavefunction every 20 iterations, projected onto the r and R coordinates.

It corresponds to a job with average energy $12mE_H$, $v_i = 1$, $j_i = 2$, $\Omega_i = 0$ and negative parity.

At the top, the classical position, the initial spread, the flux surface and the start of absorption are shown, respectively. In the figures below that, the wavefunction is projected on Ω and j as a function of time.

Only the initial part of the propagation is shown; the calculation goes on for much longer to almost fully absorb the wavefunction and to resolve resonances. This is very dull, and hard to see due to small norm, so this is omitted from the video.

6.2 Manual

This is a fairly detailed manual of how to use the software. This manual was created as part of the project.

The calculation can be controlled using many parameters. These parameters are described here, including information for easily adapting them to new problems. The final values that were used after several rounds of careful tuning are also included (after the name).

6.2.1 Implementation

The process is like this:

- A job script is created in Python, that specifies the non-standard parameter values.
- Python code adds all the unspecified parameters, computes dependent parameters and does input validation.
- A directory with source files is created for each run. `readygo.sh` and `conf.inp` are generated dynamically to contain all the parameters. This directory is self-contained and and could be run with `readygo.sh`.

[6] I've since become a professional Java programmer.

- Typically, Python schedules the jobs on one of the supported queues, like `slurm`, or starts them on various nodes directly.
- Each run compiles the code (with its own flags) and runs it. It reads input from `conf.inp`, and from environment set in `readygo.sh`. There is extra code in `readygo.sh` to do restarts, avoid using NFS, clean up temporary files, kill deadlocked processes, merge results, and more.
- Python is used to extract results from output files and create any figures and other reports, for many runs at once. Extracted results are stored in an efficient format, making subsequent analysis much faster.

Altogether, one never needs to start jobs manually, but one always can. This way it is convenient to manage hundreds of jobs, but also to debug individual ones at a low level.

There are some parameters in `conf.inp` that are not tunable from Python, because so far there has not been a need to change them.

`conf.inp` contains information about which parameters correspond to which variables in Fortran, which can be useful when debugging.

6.2.2 Physical parameters

Have a look at section 2.6.1 (States) for the meanings of quantum numbers.

Most of these physical parameters do not list a value, because the physical values were different between jobs.

- **`energy`**
  The average initial kinetic energy of the wavepacket. The wavepacket does not have a single well-defined kinetic energy, the spread is determined by `wp_R_spread`.
  Remarks:
  - The program outputs an initial translational energy, which may be different for high J states. This happens when the wavepacket is inside the centrifugal barrier, which should not happen. The solution is to make the R-grid longer and move `wp_R_center` outwards for bigger J if possible. It may be acceptable to ignore small deviations.
- **`init_v`**
  ν_{init} is the initial vibrational state of the diatom, with zero as ground state.
  Remarks:
  - You can easily see the vibrational state from a plot of the initial wavepacket, since it has a number of nodal planes along r equal to the ν level.
  - The Fortran code starts numbering of ν states at 1 as the ground state, but the Python code start at 0.
  - The deprecated parameter `init_vib` is the initial vibrational state, but with 1 as the ground state. Do not set both `init_v` and `init_vib` at the same time.

- **`init_j`**
  j_{init} is the initial rotational state of the diatom, with zero as ground state.
  Remarks:
  - The deprecated parameter `init_rot` is identical, but should not be used.
- **`init_omega`**
  Ω_{init} is the initial Ω state. Omega is at least 0 or 1 depending on the parity and J, and is at most j_{init} or J, whichever is lowest. You generally need to calculate and average all Ω_{init} states to get integral results.
  Special values:
  - `none` - This will set Ω_{init} to the lowest allowed value (0 or 1).

  Remarks:
  - The $\Omega = 0$ state does not always exist, depending on J and parity as described in section Equation (7).
- **`J`**
  J is the total angular momentum quantum number for the calculation, which stays constant. You generally need to calculate and average J states until they stop contributing to get integral results (that is, until transitions stop happening for the kinetic energy range).
  Remarks:
  - A higher J means more Ω levels exist.
  - J states have a degeneracy of $2J + 1$ due to the M quantum number, which has no effect besides this.
  - If J gets high enough, eventually there are no transitions anymore. Physically this is because the the atom and molecule are too far away from each other (big impact parameter and thus ℓ). In the non-inertial coordinates used, this is visible as the centrifugal barrier getting too high. This depends strongly on other parameters, most importantly j_{init} and kinetic energy.
  - Transition rate as a function of J is not necessarily monotonically decreasing, and could even increase after reaching zero due to glory scattering.
  - If you calculate cross sections or transition probabilities that are correct at lower energies, but get too low as energy increases, then perhaps more J states should be included. Results must be converged as a function of J_{max}.
- **`parity`**
  `parity` determines the parity of the wavefunction under inversion. The input value should be -1 or $+1$, equal to $(-1)^p$ rather than p. You generally need to calculate and average results for both parity states (if they exist) to get integral results.
  Special values:

  - `none` - Sometimes only one parity state is allowed, as described in section 2.6.4 (Restrictions). Parity can be set to `none` to choose an allowed value. But this should onyl be used for exploration; when calculating results, one should know which parity one needs.

  Remarks:

  - In the code, the variable `parity` is 1 or 0 again, like the quantum number, but the input should be -1 or $+1$.

- **`molecule_formula`**
  Diatom formula, in the format `12C 16O`.

- **`atom_formula`**
  Atom formula, in the format `1H`.

- **`partity_inv_sym`** `all`
  Can be used in case the system has symmetry properties; one of `all` (used), `odd`, `eve`.

- **`potential_name`**
  The keyword for the potential, e.g. 'hco' for the Song *et al.* HCO potential. A new potential will have to be implemented and given a name in the Fortran code before using it here.

6.2.3 Influential parameters

- **`r_dvr_min`**: `1.8`
  The lower end of the vibrational (r) grid, in bohr.
  Remarks:

  - Setting this too high will cut essential grid points, which yields very inaccurate results. There are not necessarily any clear warning signs, so avoid it carefully.
  - Including too low r values introduces very high potential elements in the Hamiltonian if `potential_cut` is not set correctly, which leads to very small timesteps.
  - A reasonable initial value can be found using the classical turning point. There is not yet automatic setting to determine that value.
  - Tuning this can be done by making the grid large and saving the wavefunction to see which r values it reaches.

- **`r_dvr_max`**: `-2.75`
  The higher end of the vibrational (r) grid, in bohr.
  Special values:

  - `negative` - A negative value will use the absolute, and add 0.1ν bohr. So -2.75 at $\nu = 2$ will use $+2.95$ bohr. Unlike at `r_dvr_min`, the potential is not steep at the maximum, so the vibrational wavefunction will expand this direction for higher vibrational levels.

  Remarks:

  - Setting this too low will cut essential grid points, which has the same bad effect as for `r_dvr_min`.
  - A reasonable initial value can be found using the classical turning point. There is not yet automatic setting to determine that value.
  - Tuning is similar to `r_dvr_min`, but depends more on ν levels.
  - Setting this higher to accommodate vibration during integration expands the r grid along the whole R axis, even though the higher vibration levels might only occur during interaction (at low R). There is currently no feature to prevent this.

- **`R_dvr_min`**: `2.0`
  The lower end of the scattering (R) grid, in bohr.
  Remarks:
  - Similar to `r_dvr_min`. The closest approach may depend somewhat on j and energy.

- **`R_dvr_max`**:
  The end of the R grid. This is a dependent parameter, see section 6.2.4 (Dependent parameters).
  Remarks:
  - Note that this qualitatively different from `r_dvr_max`, since the wavefunction can escape in R but not in r.

- **`R_end_abs_dist`**: `4.0`
  The size (bohr) of the absorption region in the R coordinate, which is at the end of the grid. Closely related to `R_abs_strength`.
  Remarks:
  - `wp_R_center`, `R_flux_wp_dist`, `R_abs_flux_dist` and this can together be used to set all R initial positions. This makes it so that if the absorption region is too small, only one coordinate needs to be updated, making tuning easier. Absolute positions should not be used together with this.
  - Absorption is harder when outgoing kinetic energy is higher. Tune at high energy, or make it adaptive (this or `R_abs_strength`).

- **`R_abs_flux_dist`**: The distance between the absorption point and the flux surface. This is a dependent parameter, see section 6.2.4 (Dependent parameters).

- **`R_flux_wp_dist`**: `1.5`
  The distance (bohr) in the R coordinate between the center of the initial wavepacket and the flux surface. The initial wavepacket should not touch the flux surface, as it has to interact before being analyzed.
  Special values:
  - `none` - set the value to 3.54 times `wp_R_spread`, which leaves 0.01% overlap on each side of a Gaussian distribution.

  Remarks:

- `wp_R_center`, this, `R_abs_flux_dist`, and `R_end_abs_dist` can together be used to set all R initial positions (instead of using absolute values). This makes it so that if the flux surface is too close to the wavepacket, only one coordinate needs to be updated, making tuning easier. Absolute positions should not be used together with this.
- The flux surface is located at a grid point, so the actual distance may be half a grid spacing difference.
- This mostly depends on the spread of the wavepacket. It was made automatic, but only after the calculations in this work were done.

- **`wp_R_center`**: The position of the center of the wavepacket. This is a dependent parameter, see section 6.2.4 (Dependent parameters)

- **`R_abs_strength`**: `0.007`
  The strength factor that determines how fast the wavepacket is absorbed in the absorption region, as given in Equation (16) (section 2.8.4 (Absorption)).
  Remarks:
  - Absorption needs to be stronger when outgoing kinetic energy is higher. It is best to tune at high energy. An alternative is to make `R_end_abs_dist` larger, which is more costly due to a larger grid, but less likely to result in reflections.[7]. See Figure 5.

- **`r_dvr_spacing`**: `-0.15`
  `r_dvr_spacing` is the distance in the vibrational (r) coordinate in bohr. This determined the number of grid points `r_dvr_count` in that dimension. Special values:
  - `positive` - The number of grid points is set to the amount needed to have at most the requested grid spacing (so the amount is rounded up).
  - `negative` - Like positive, but the spacing is decreased by a factor $1 + 0.5\nu_{\text{init}}$, equivalent to 50% more grid points for each vibrational level. A higher vibrational level has a lower wavelength and as such needs a bigger grid to describe.

  Remarks:
  - The grid is equidistant since a sync DVR is used.
  - If this is set, `r_dvr_count` should be `none`. Using spacing is typically preferable, since grid spacing rather than size determines accuracy.
  - This parameter has a big effect on performance since it makes the wavefunction and Hamiltonian bigger.
  - It also has a big effect on accuracy, with results rapidly becoming completely wrong when the wavefunction cannot be described accurately anymore.

[7] There is some code to set absorption strength automatically based on absorbing 99.0% on the way out (not counting reflection). This is a discrete iteration using an assumed timestep, and should be treated as an approximation.

- When tuning, find the largest spacing for which changes do not influence the probabilities significantly anymore.
- As a future change, this could be determined automatically based on the De Broglie wavelength:

$$\lambda = \frac{2\pi}{\sqrt{2\mu E_{kin}}}. \tag{23}$$

The kinetic energy in the minimum of the potential should be used. The Nyquist-Shannon sampling theorem imposes a minimum of two points per wavelength to represent the wave, but this may not be sufficient for accurate results.

- **R_dvr_spacing**: `-0.22`
`R_dvr_spacing` is the distance in the scattering (R) coordinate in bohr. This determined the number of grid points `R_dvr_count` in that dimension. Special values:
  - `positive` - The number of grid points is set to the amount needed to have at most the requested grid spacing (so the amount is rounded up).
  - `negative` - Like positive, but the spacing is decreased by a factor $1 + 0.5\nu_{\text{init}}$, equivalent to 50% more grid points for each vibrational level. Although this coordinate does not describe vibrational motion, vibrational energy may be converted to kinetic during scattering.

  Remarks:
  - If this is set, `R_dvr_count` should be `none`.
  - The notes at `r_dvr_spacing` also apply here.
  - If `R_flux_wp_dist`, `R_abs_flux_dist`, and `R_end_abs_dist` are used, `R_dvr_spacing` must be used as well (rather than `R_dvr_count`).

- **jmax**: `-50`
$j_{\max}$ is the highest rotational state to include in the angular basis. The minimum and step size can vary depending on Ω and parity symmetry, so this does not always correspond to the number of angular basis functions, but it has a close relation.
Special values:
  - `negative` - The value will be set to |`jmax`| + `J`, since a larger basis is needed for higher angular momentum. There is a maximum of 70 because of numerical issues for too high j, and because it seems sufficient in practise.

  Remarks:
  - The calculation needs $j_{\max} > j_{\text{init}}$, probably by a wide margin depending on the available energy.
  - `proj_max_rot` must not be higher than $j_{\max}$ since that would mean a rotational state that was not present in the basis is projected for analysis. It does not makes no sense to analyze the flux of such a state.

- Without neglecting Ω states, the calculation needs $j_{\max} \geq J$. If a smaller value is needed, use `omegamax` to neglect Ω states.
- There seem to be convergence problems starting around $j_{\max} = 80$. This happens in the Fortran routine `chkortassl`, which deals with the orthogonality of associated legendre polynomials. Since no such value was needed in the project, the issue was not investigated in detail.

- **`omegamax`**: `none` for most normal calculations, `65` for a few; a set of control calculations was done at `omegamax = 16` with very similar results. $\Omega_{\max}$ is the maximum Ω level to include, above which Ωs will be completely neglected. Since every Ω has its own process, this saves a process for every level neglected.
  Special values:
  - positive - The lowest of `omegamax`, `J` or `jmax`, the last one raising a warning.
  - `none` - This will include every Ω state if $J > j_{\max}$, a warning is raised otherwise.
  - negative - Ω states up to $j_{\text{init}} - \Omega_{\max}$, up to J. This works well if big j transitions are unlikely.

  Remarks:
  - Since $\Omega \leq j$ (and Ω states only directly couple to neighbors), many Ω are virtually unreached if $J \gg j_{init}$.
  - $\Omega = 0$ does not always exist, so $\Omega_{\max}$ is not always the number of Ω states.
  - `proc_count` should be left `none` if `omegamax` is set.
  - If $\Omega_{\max} > j_{\max}$, including when set automatically, a warning is raised.
  - Make sure to look at big j transitions (large $|j_{final} - j_{initial}|$) when tuning, as these are most affected by missing Ωs (e.g. Figure 9).
  - Looking at j and Ω occupation for the dumped wavefunction is also very informative (see `vartheta_omega_basis_use`).
  - If the physical system has more than one Ω state, at least two must be included, which is a technical limitation to detect MPI problems.

- **`time_duration`**: `-25.000`
  The maximum propagation time in atomic units (unless `norm_stop_crit` is reached first).
  Special values:
  - negative - `time_duration` is set to its absolute value, plus the time it would take a classical particle with kinetic energy `energy` to reach the minimum and then maximum of the grid in the absence of any interaction. So if the wavefunction has to travel further or has lower energy, the propagation time will be proportionally larger.

  Remarks:

  - Setting this means `time_step_count` will be automatically computed, so do not set both. This computation happens in the Fortran code because `time_step_min/max` must be computed first.
  - This has an almost linear effect on performance, since the majority of the calculation is spent propagating the wavepacket.
  - Two jobs with the same `time_duration` can require quite different numbers of iterations, because the time per step is different (and not known beforehand).
  - It is easy to tune this by looking at the (non)react probability and the remaining norm (which should approach 100% and 0 respectively).
  - Note that a large part of the integration will be spent on the last few percent of the wavepacket. This contains wavefunction parts that moves slowly or interacted for a long time. Therefore it is important for low energies and resonances.
  - The time duration may not be reached in case the computation is stopped early due to the wavepacket being mostly absorbed (`norm_stop_crit`).
  - This is not exact in an insignificant way, because time step are discrete (several atomic units).

- **`norm_stop_crit`**: `0.01`
  When the remaining norm of the wavepacket falls below this value, the propagation is terminated. (If this value is not reached before that, the propagation is terminated after `time_duration`).
  Special values:

  - `0` - Setting this to `0` means the full propagation time will always be used (this is not a magic value; 0 is just unreachable).

  Remarks:

  - Although this seems like a better way to determine when to stop than setting the propagation time, the last fraction of the wavefunction can be very hard to absorb, so do not rely on this alone.

- **`wp_R_spread`**: ranging from `0.15` to `0.42` depending on energy
  The spread of the Gaussian wavepacket in the R coordinate. See section 2.7.3 (Initial R wavefunction) for details.
  This parameter is deceptively hard to tune, because:

  - There is no safe-but-slow option, it can be both too low and too high.
  - Wrongly tuned values are easy to miss when not comparing `wp_R_spread` explicitly (and even then).
  - Its optimal value depends strongly on `energy`.

  Remarks:

  - A wavepacket that is wide in R is narrow in energy, as explained in section 2.7.3 (Initial R wavefunction).

  - A wide spread in E space means that there is a wider range of reliable output probabilities, since only those energies that have sufficient wavefunction amplitude are reliable. A good indication of reliable results is the total probability (summed over all output channels at fixed kinetic energy).
  - If the energy spread is wide relative to the average energy, then parts of the wavefunction may have near-zero or even negative energy. They could then be absorbed directly and silently ruin results, or move so slowly that they are not absorbed within `time_duration`.
  - Observed effects of too wide energy spread (low R spread) are norm decreasing very un-monotonically, and probability peaks shifting to other energies. Overall it is not easy to detect.
  - Too narrow energy spread may fail to converge (which is not meant to imply that all values that do converge are good values).
  - The energy range where predictions are reliable is very roughly 1 standard deviation in energy space. Often the range is wider on the high-energy side.
  - This parameter's optimum depends strongly on kinetic energy, but also on vibrational level.

- **`prob_E_spacing`**: `0.0001` at most energies
  The energy spacing in E_h at which energy-dependent state-resolved probabilities are produced.
  Remarks:
  - Finer energy resolution is usually needed at lower energies.
  - Setting this will automatically set `prob_E_count`, so do not set both.
  - Using fixed energy spacing is highly recommended to make combining results from different jobs easy. Therefore, using spacing rather than count is especially recommended when `prob_E_max` is dynamic.
  - The energy dependent probability is a Fourier transform of the time-dependent flux (section 2.9.1 (Analysis approach)). This imposes a limit on the amount of information available; too fine a spacing does interpolation without adding more information. This could perhaps be made automatic, but that would interfere with the point above.
  - This has minimal impact on computation time, but takes quite some storage space, because the energy-dependent state-to-state probabilities take most of the diskspace. For single calculations it is negligible, but it takes gigabytes total when computing many partial waves.

- **`proj_max_vib`**: `7`
  The highest vibrational state to calculate state-resolved probabilities for, with `0` being the ground state.
  Remarks:
  - The note about space use for `prob_E_spacing` applies here as well.
  - High rovibrational states may not be accurate. A file with norms for each generated projection channel is produced. Check that they are all close to unity.

  - Energy is conserved, so the highest occupied output channel depends on the initial energy (rotational, vibrational, and kinetic).
  - When summing probabilities from different jobs, only the lowest `proj_max_vib` among jobs is used. So it would be best to keep this constant, or update the Python code if it is important to have different values.

- **`proj_max_rot`**: `30`
  The highest rotational state to calculate state-resolved probabilities for, with `0` being the ground state.
  Special values:
  - `none` - Equal to `jmax`. This is likely higher than useful, since more energy is available during interaction than asymptotically, and `jmax` must be sufficient to describe both.

  Remarks:
  - See `proj_max_vib`, it is exactly analogous.

- **`chebychev_input_count`**: `700` used, `300` also works
  The number of Chebyshev iterations performed for the first integration step, as described in section 2.8.2 (Integration).
  Remarks:
  - Note that a $\hat{H}\Psi$ evaluation is needed for each step, so this adds a noticeable amount of time to the calculation. However, it is not part of the rest of the integration, so its constant computation cost is overshadowed by parameters that impact that process.
  - A reasonable indication of the value needed can be obtained by looking at the logfile of a calculation. It will show the norm converges during this process, so choose enough steps for convergence.
  - The number of iterations needed for convergence depends on the spectral range of the Hamiltonian; which should not vary a lot between most runs.

- **`potential_cut`**: `0.7` (possibly too high)
  Any potential energy values above `potential_cut` will be reduced to `potential_cut`. This decreases the spectral range of the Hamiltonian. A large spectral range makes timesteps small and propagation slow.
  Remarks:
  - Setting this value too low may make regions accessible that energetically should not be.
  - Setting this to a high value is a safe choice as golden standard, but could have a rather large effect on performance.
  - This depends on the energy available to the wavepacket. This parameter could perhaps be automated in the future.
  - If a region is not visited, when possible, it is preferable to remove it rather than (or in addition to) cutting its potential, but that is harder.

- Can be tuned by comparing decreasing values until the probabilities get influenced.

- **`proc_count`**
  Number of processes, which can just be determined based on `J` and `parity` if set to None. If set to a lower number, this causes the highest Ω levels which do not have a process to be neglected.
  Remarks:

  - To prevent accidentally neglecting Ω states, when using this option, **`_comp_cpp_defs`** must contain **`ALLOW_NEGLECT_OMEGA`** (this option is listed later).

6.2.4 Dependent parameters

These parameters can be set when necessary, but a value can also be derived based on other parameters. In typical use, it is recommended to use the automatic values, as these work well in most cases and save effort tuning what are already at least 22 parameters.

- **`R_abs_flux_dist`**: `none`
  The distance between the absorption region and the flux surface in bohr.
  Special values:

  - `none` - Set to $1.5 \cdot \Delta R$ (the R grid spacing).

  Remarks:

  - `wp_R_center`, `R_flux_wp_dist`, this and `R_end_abs_dist` can together be used to set all R initial positions. This makes it so that if the flux surface is too close to the absorption region, only one coordinate needs to be updated, making tuning easier. Absolute positions should not be used together with this.
  - Nothing of interest happens between the flux surface and the absorption region, so these can be close together.
  - The flux surface is located at a grid point, so the actual distance may be up to half a grid spacing higher or lower.

- **`wp_R_center`**: `none`
  The position of the initial wavepacket in the R coordinate.
  Special values:

  - `none` when using distance coordinates - Position the wavepacket such that the centrifugal barrier is no more than 10% of the kinetic energy. There is a minimum separation of 10 bohr to account for the interaction potential. This automatically moves the wavepacket further away for higher J states (stronger centrifugal barrier). Note that larger distances also need longer integration times, so this combines well with automatic values of `time_duration`.
  - `none` when using absolute positions - Positioned four wavepacket spreads away from the flux surface. Not recommended.

  - `< 1` - Set as a fraction of the grid, 0.8 will position it at 80% of the R grid length.

  Remarks:

  - Should be large enough to be outside the centrifugal barrier and the interaction potential.
  - Should be far enough away from the flux surface for the initial wavepacket to not overlap with it.
  - When distance coordinates are used (`R_flux_wp_dist`, `R_abs_flux_dist`, `R_end_abs_dist`), this is the anchor that determines the position of the flux surface, the absorption region and the end of the R grid.

- **`prob_E_min`**: `0.0`
  The lowest energy for which energy-dependent state-resolved probabilities are produced.
  Special values:

  - `none` - The minimum will be set to the average energy minus two times the energy spread.

  Remarks:

  - Note that this refers to total initial energy. If the initial internal energy is 3 and you set this parameter to 1, then the points until 3 will not produce output. Instead the grid will start at 3 and you can shift it to be initial kinetic energy instead of total.
  - In most cases you can just leave this as 0.0, as internal energy will be skipped regardless, and it is a bit tricky to tune beforehand.

- **`prob_E_max`**: `none`
  The highest energy for which energy-dependent state-resolved probabilities are produced.
  Special values:

  - negative - The absolute value will be used, plus the initial energy (kinetic and internal). So this effectively sets maximum as how much kinetic energy above the wavepacket energy to include.
  - `none` - The initial energy plus 1.3 times the energy spread will be used. This is a fairly good prediction of the accurate region.

  Remarks:

  - See the notes about performance and spacing at `prob_E_min`.

- **`R_flux_point`** / **`R_abs_pos`** / **`R_dvr_max`**
  These values can be used, along with `wp_R_center` and `R_dvr_count`, to position the wavepacket, flux surface, (start of the) absorption region and the end of the grid.
  Special values:

  - fraction - Setting these to a value between 0 and 1 will set them to that fraction of the grid.

It is recommended to use relative distances instead (`R_flux_wp_dist`, `R_abs_flux_dis`, `R_end_abs_dist`), as it is the distance that determines accuracy, rather than relative or absolute position on the grid.

- **`time_step_min` / `time_step_max`**
  These values can be used to set the timestep, which effectively scales the Hamiltonian. However, as described in section 2.10.3 (Range of eigenvalues), code was added to determine this automatically, and it is no longer recommended to set manual values for these parameters.

6.2.5 Unimportant parameters

- **`pot_poly_count`**: `30`
  This used to be the number of polynomials to use for the Gauss-Legendre integration that constructs the potential matrix, see section 2.10.2 (Computing potential energy), but it is not used anymore.

- **`cut_inner`**: `0.021`
  Determines the region where points will be removed because atoms are too close together. See section 2.10.1 (Grid filtering).
  Remarks:

  - Usually only cuts few points for a well-tuned grid.

- **`cut_outer`**: `0.2`
  Determines the region where points will be removed because atoms are too far apart. See section 2.10.1 (Grid filtering).
  Remarks:

  - This is not very useful for non-reactive scattering, because regions where there atoms are far apart would indicate a broken molecular bond (reactive).

- **`prob_save_step`**
  This determines how often the wavepacket it saved (once every `prob_save_step` iterations).
  Special values:

  - `none` - This will set a high value, which will be decreased as necessary to avoid aliasing.

  Remarks:

  - If a chosen value is found to cause aliasing, it is decreased automatically. This happens in the Fortran code, which means the value set does not always match the value used.
  - This determines the resolution of the resulting energy grid. The spacing is set by `prob_E_spacing`, but enough timepoints must be available to obtain accurate energy data.
  - This parameter is an exception, and can be updated during the calculation even if `NO_PARAM_CHANGE` is set. There is a special `NO_PARAM_CHANGE_SAVECRIT`, if desired.

- **mpi_mode**
  Should always be `mpi` if you want state-to-state probabilities, since the analysis is not implemented for `nompi` mode.

- **omega_pp**
  This is the number of Ω states per processor, with values 1 and 2 being accepted. **This should always be one!** The value two seems to have bugs and is not well-tested. There is little incentive, performance or otherwise, for fixing this. The aim was for omegas per process to me matched, with highest and lowest together for ideal load balancing.

6.2.6 Bookkeeping parameters

- **keep_files**
  If set to True, keep many files that are not strictly essential. Very useful for debugging. In absolute terms the disk use is not very high (MBs), but it is a large fraction that adds up if you have many jobs.

- **scratch_cwd**
  This runs the whole calculation on a scratch disk (may need to tweak 'readygo.sh' a tiny bit). Doing this may save space during computation, but especially prevents any possible NFS problems that may occur. The code is copied at the start of the calculation, and the results are copied back after it terminates, so you should not notice much.

- **write_wavefunction**
  If True, dumps the wavepacket to the disk every `coef_save_step`. This takes large amounts of space (GBs) and should only be `True` if you plan to visualize the propagation process. Note that the compilation flag `POP_BIN_FILES` determines the dump format.

- **coef_save_step**
  How often integration info is printed (in steps) and, if dumping wavepacket is enabled, how often that is done.

- **log_shell_cmds**
  If you suspect that there is a problem in the run script (`readygo.sh`), turning this on will log all shell commands for debugging.

- **selective_error_wipes**
  In the phase where some jobs had non-deterministic crashes (see `FLUX_PLAINTEXT`), this setting was added to detect specific crashes and restart them automatically. It should probably not be used anymore.

- **compile_deb_mode**
  Whether to compile in optimized mode (`opt`), debug mode (`deb`) or mixed mode (`mixed`). In mixed mode, you can use `_comp_deb_subset` to name the files that need to be debugged, with the rest being compiled in optimized mode. Debug mode changes several compiler flags, and adds the `DEBUG` define. Enabling this may slow code down a lot (which mostly matters for files like 'integrate.F' and 'hpsiclc.F'), and may also give very verbose debug output depending on the file ('prbeclc.F' has tons of debug code for example).

- **_comp_deb_subset**
  If `compile_deb_mode` is set to `mixed`, then this should be a comma-separated string of files to compile in debug mode.
- **compiler_type**
  The `gfortran` (for portability) and `ifort` (for speed, recommended) compilers are implemented, you can implement others in the `SCons` file.
- **_comp_extra_flags**
  Extra flags to pass to SCons for compiling, see `scons -h` for options.
- **_comp_cpp_defs**
  There are a few flags which should always be set. If you're compiling with SCons, this will be done automatically (see `scons -h` for help). But there are also a few that you may want to use to tune behavior. Those can be added as a comma-separated string for `_comp_cpp_defs`:
  - **DEBUG** - Substantial debug code is left in the codebase, and can be enabled with the DEBUG cpp flag. You should probably use `compile_deb_mode` and `_comp_deb_subset` to enable this for specific files, which will also take care of compiler flags.
  - **TIMING** - If provided, the major blocks of the program are timed, with results written to "progress.time" (for the first process). The performance penalty is very minimal, so you can use it by default.
  - **NO_PARAM_CHANGE** - Sometimes the Fortran code is helpful and updates invalid parameters to valid ones. But then you think you are doing different calculation than you actually are, which can be near-impossible to find out. If you use `NO_PARAM_CHANGE` then many (not all, e.g. `NO_PARAM_CHANGE_SAVECRIT`) of these updates will become errors, so you are probably calculating what you think you are. I recommend always using this.
  - **STOP_EARLY** - In a few situations, something that goes wrong early in the program can cause problems later. With this flag, a few of those situations are tested for, and the program is terminated early instead of late. It is not very important, but it can save some time.
  - **FLUX_PLAINTEXT** - This one is weird. During integration, the code writes flux data to disk, to be loaded back during flux analysis. For some computations, valid values were written but invalid ones were read when using the default Fortran unformatted files (sometimes the same parameters would have different effects). It sounds weird, and that is exactly the reason it took weeks to pin down. It was found that using plain text instead of Fortran unformatted solved the issue in every case, at the cost of temporary disk space. This flag switches to that plain text mode, and it should probably always be used. If one finds that identical calculations yield unreasonable values in some cases (depending on parameter, mostly at high J) then turn this on before spending weeks debugging!
  - **ALLOW_NEGLECT_OMEGA** - This must be set if you want to make approximations that neglect Ω states. This was added because you

could easily, silently neglect omega states without noticing. Now you need to request that approximation explicitly by using this flag to skip tests.

- `POP_BIN_FILES` - Normally, population dumps are written to one Fortran unformatted file. With this flag, dumps as plain binary files, with one file for each omega at each snapshot (many files). Disk use and performance are similar, so use the format you prefer. Note that dumps are only made if `write_wavefunction` is True.

6.3 Code

The code is attached digitally, as described in section 3.1 (The software).

6.3.1 Tuning parameters

What follows is a general outline of the tuning process. It is hoped that this will allow the code to be reused for other atom-diatom scattering simulations. The code supports reactive scattering, although this list is based on the experience from this report, where scattering is non-reactive.

1. If at all possible, start from one of the configurations known to produce reasonable results if possible. It is very hard to optimize if the situation before and after a change are both so wrong that one cannot say which is better.

2. Adapt parameters from section 6.2.2 (Physical parameters). Small changes to the code will be needed to use a new potential. A moderate energy is safest to start with. Low rovibrational excitation and low J allow for the fastest tuning.

3. Turn on debugging features such as `keep_files` and `write_wavefunction`. There is no need to enable debug mode in the compiler unless there appear to be bugs.

4. Leave parameters that have automatic values the way they are, including those in section 6.2.4 (Dependent parameters), and `omegamax = proc_count = none`.

5. For the parameters that have a costly but accurate 'direction', increase the parameter into that direction. There are many parameters like this: making the grid spacing smaller and the grid larger (R, ν, and j) will not decrease accuracy. Making the interaction region and absorption region larger is also safe, as is increasing the propagation time (with `time_duration` and `norm_stop_crit`). Others are low `prob_E_spacing`, high `proj_max_vib` and `proj_max_rot` and high `chebychev_input_count` and `pot_poly_count`. `potential_cut` can also be higher, but that can be costly.

6. Start the search for parameters that can be inaccurate both when too large and too small. Perhaps the most sensitive one is `wp_R_spread`. If this is too high in relation to energy, then parts of the wavefunction may have negative or near-zero energy, while other parts have too low intensity

to obtain meaningful results. If this is too low, the envelope may be too narrow compared to the wavelength, or only a narrow band of energies may have reliable results. Another parameter is `R_abs_strength`, which will cause reflections if too low (at end of grid) or too high (from the absorption barrier itself). It is possible to inspect the wavefunction dumps to detect too weak absorption.

7. After successful results have been obtained, tune further to increase performance, by slowly moving the parameters that have a costly but accurate 'direction' back away from the costly end. This should be continued until results are starting to change.

8. If desired, experiment with neglecting Ω states.

9. These steps will have to be repeated when calculating different physical configurations, especially energy. It will be faster after the first time, as changes are smaller.

Always make small, measured changed. Make sure it is easy to revert changes that do not improve results. Using a version control system for the source code has proven useful.

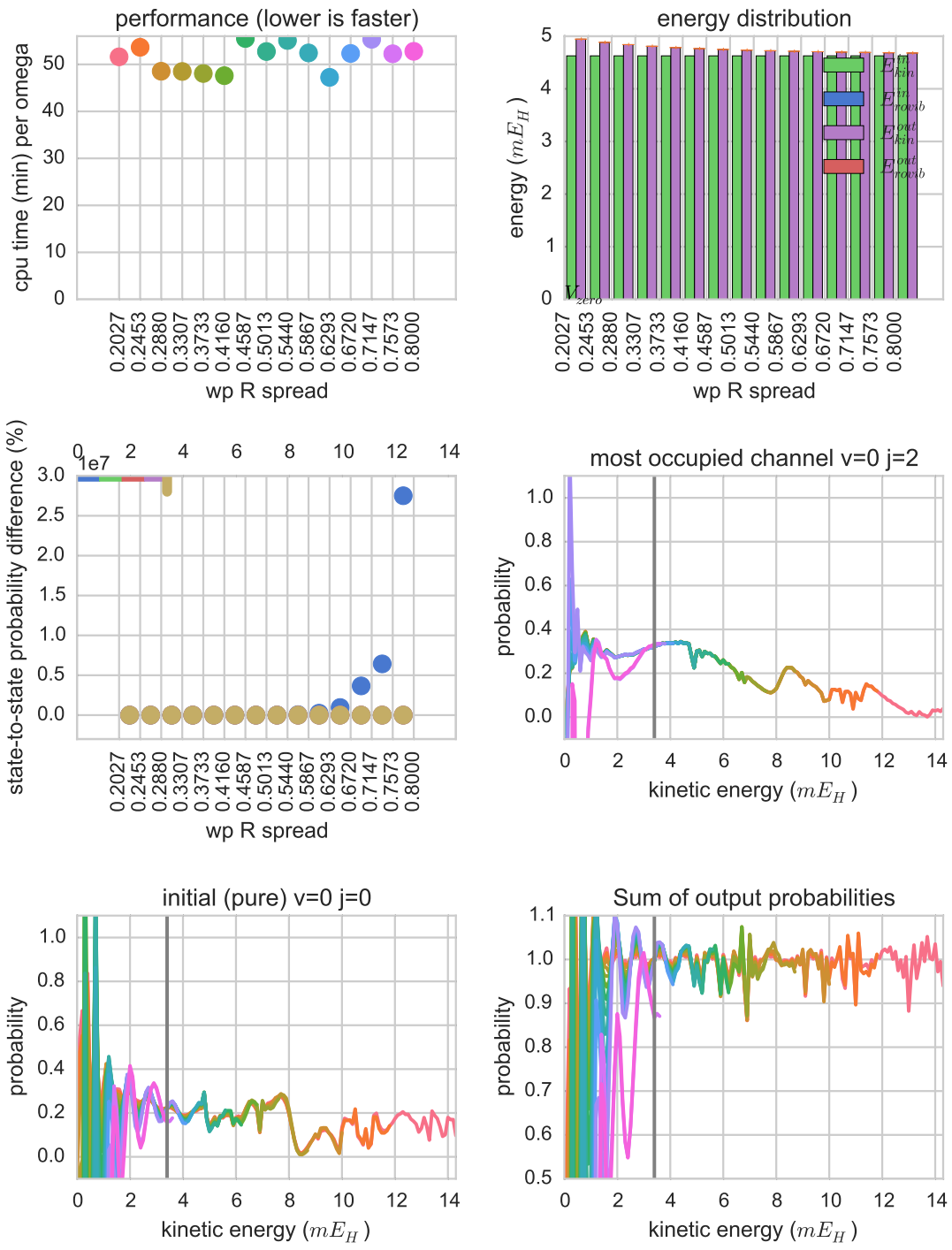


Figure 31: See the other half of this plot, Figure 32, for info.

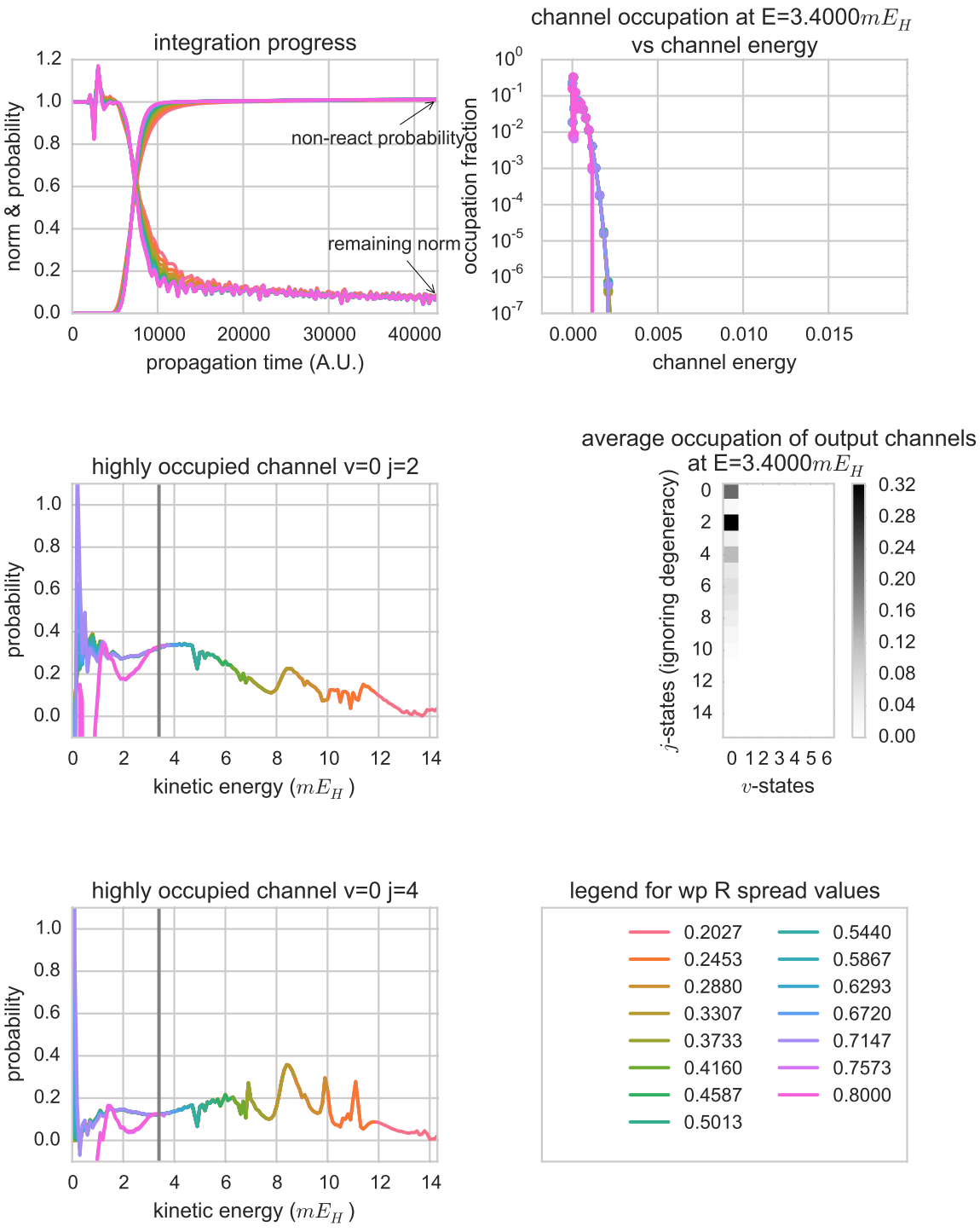


Figure 32: This, along with Figure 31, is an example of the tuning overview plot where the spread of the wavepacket in R varies - possibly the most problematic parameter. An explanation is in Figure 35.

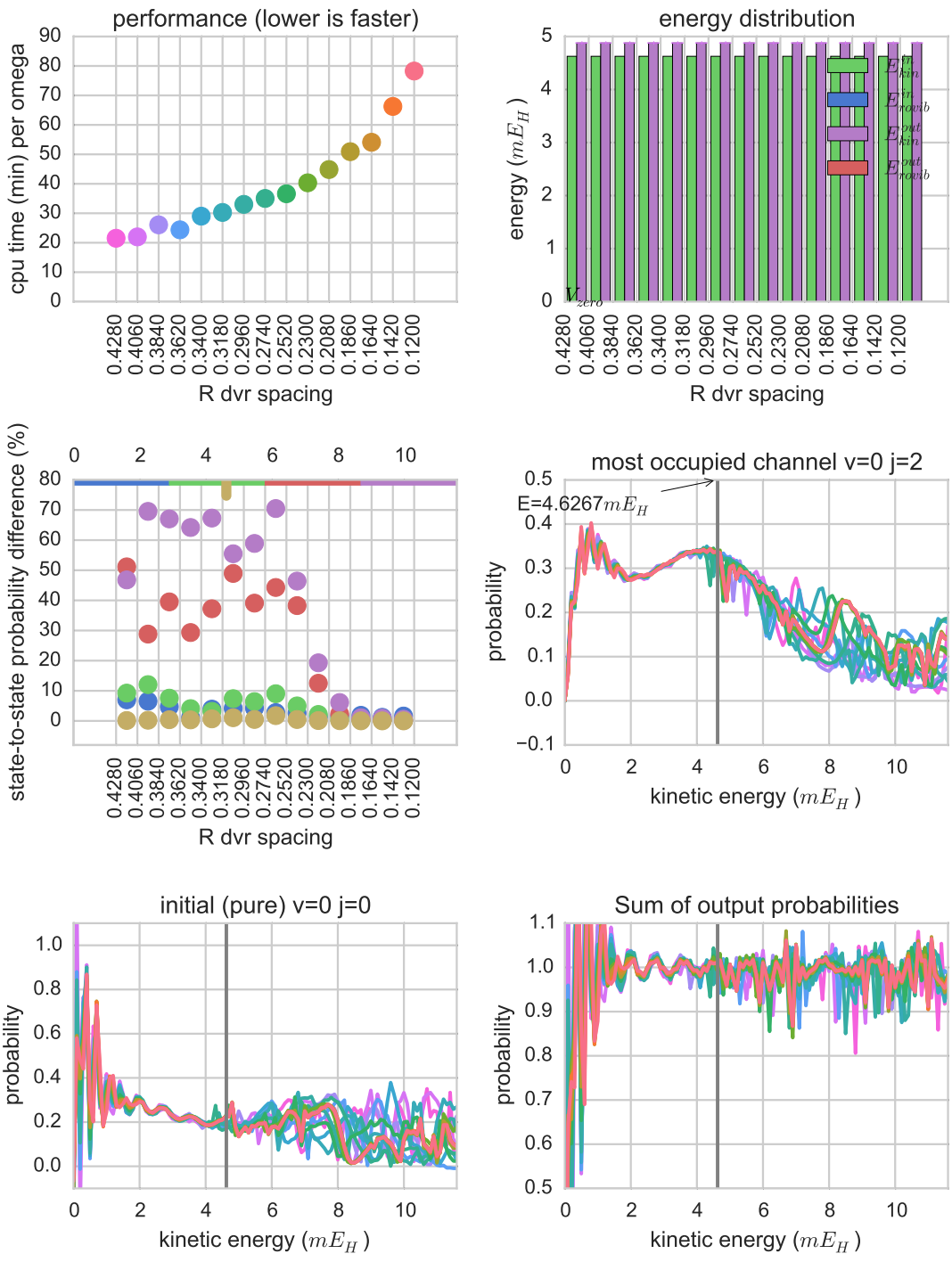


Figure 33: See the other half of this plot, Figure 34, for info.

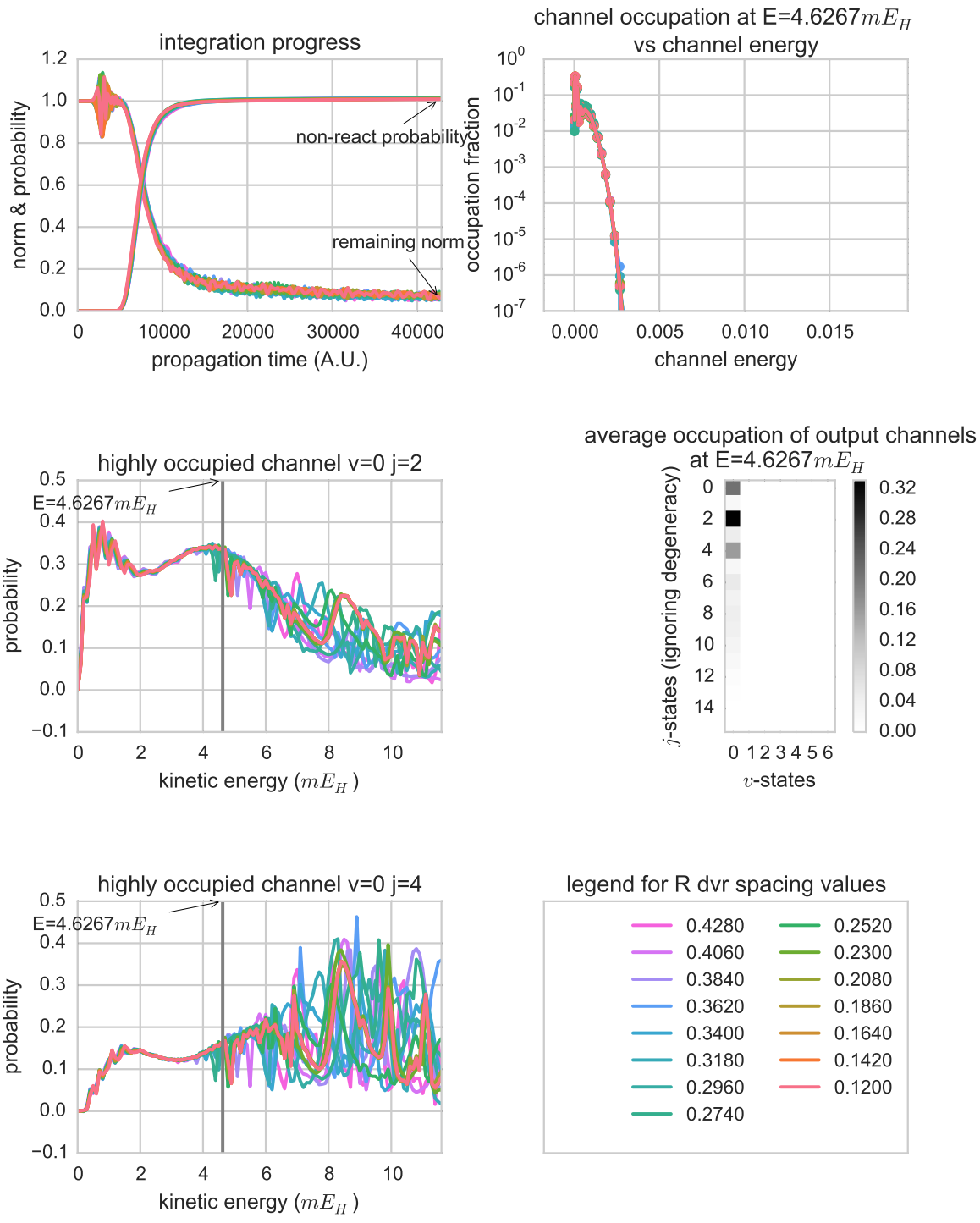


Figure 34: This, along with Figure 33, is an example of the tuning overview plot where the R grid spacing varies. An explanation is in Figure 35.

A	B
E	F
I	J

C	D
G	H
K	L

Figure 35: Legend for overview plots (Figure 31, 32, 33, 34), the two blocks representing the first and second page, and each letter representing a subplot by position. Except for **B** and **E**, each color is a job, with a legend in **L**.

- **A**: Computation (CPU) time as a function of the parameter.
- **B**: Incoming and outgoing distribution of kinetic versus rovibrational energy. Useful for example in the case of J, where incoming energy depends on J if the initial wavepacket is inside the potential.
- **C**: Norm (start at top) and non-react probability (starts at bottom) as a function of integration time. Useful to check convergence.
- **D**, **H**: Occupation of exit channels, per job and averaged respectively, at a fixed output energy that can be controlled.
- **E**: Compares the difference in probabilities between adjacent parameter values. Each color corresponds to an energy segment (legend at the top of the subplot). If subsequent parameter values yield the the same probabilities, the dots will be near-zero, and the result has likely converged. Not all energy regions are necessarily of interest.
- **F**, **G**, **I**, **K**: These are all probability distributions as a function of energy for a specific channel. Depending on configuration, it shows the initial state, ground state, most occupied state and other highly occupied states. The vertical line is the initial energy. This can be used to determine in which energy region which jobs have converged. The line need not necessarily be smooth, but it should be between 0 and 1, and small parameter changes should not lead to large probability changes.
- **J**: The sum of the state-to-state probabilities. This should be close to one in the converged region, and should not strongly depend on parameter changes.

Perhaps the most important metric to quickly judge whether parameters are approximately correct is whether the state-to-state probabilities as a function of energy sum to unity for some range of energies. Note that this should not be expected to be unity for all energies, and can indeed behave erratically near extrema. This is the case because results are only accurate in the range where the wavefunction has sufficient amplitude in energy space.

It is also very useful to see whether changing a parameter changes the results substantially. For accurate values, there should be an equilibrium, and small parameter changes should hardly affect results.

Many different functions have been made to visualize scattering results. One in particular combines various different plots that are useful to help tune parameters[8]. Two examples are shown in Figure 31 and 33.

[8]The tuning plot is found in `views.overview.param_overview`. Calculations can be created with `views.param_stability.generate_param_compare_jobs`.

6.3.2 Code structure

The core of the simulation is in a directory of Fortran90 files that can be built using SCons. There are various CPP rules and compile flags, be sure to rebuild for different settings.

The entry point of this code is `master.F`. The code is organized in modules with explicit imports, so one can follow the steps from the main file.

There are wrappers around this, mostly written in Python2 and some Bash, that help determine derived parameters, set up MPI, and extract results (whether successful or not).

There is also a fair amount of Python code to make starting reproducible jobs easy, using code similar to this:

```
def generate_jobs():
    Estates = [
        dict(energy=0.0028, wp_R_spread=0.4200, Jmax=25),
        dict(energy=0.0050, wp_R_spread=0.3200, Jmax=30),
        dict(energy=0.0300, wp_R_spread=0.1500, Jmax=65),
    ]
    # For each combination of v, j, and energy
    for Estate in Estates:
        for v in [0, 1]:
            for j in [0, 1, 2, 3, 4, 5]:
                # ShefJsumJob automatically queues
                yield ShefJsumJob(
                    name='v{0:d}j{1:d}_E{2:05.0e}'.format(
                        v, j, Estate['energy']),
                    batch_name='results7',
                    # These are the parameters for the jobs, with
                    # the rest defined by 'defaults_version'
                    subs=dict(
                        energy=Estate['energy'],
                        wp_R_spread=Estate['wp_R_spread'],
                        init_v=v,
                        init_j=j,
                    ),
                    # Sets the unspecified parameters to the
                    # 28th carefully tuned iteration
                    child_kwargs=dict(defaults_version=28),
                    # How many parity, omega, and J states to create
                    Jmax=Estate['Jmax'] + j,
                )

if __name__ == '__main__':
    # This places jobs in an internal queue, which can be
    # fed into slurm, qsub or something else
    queue = do_queue(
        jobs=generate_jobs(),
        # Choose a Python function to visualize results
        summary_func=analyze_results,
    )
```

This type of job file is likely the only thing that must be changed to perform further calculations.

As seen in the code, calculations are easy to create and customize. After specifying the jobs, they can be controlled from the command line:

1. Run jobs through the Slurm or Qsub systems, possibly a limited amount at a time.
2. Monitor the status of jobs.
3. Investigate, restart or clean up failed jobs easily.
4. Extract results from all the log files, cache them and show any of the available plots.

Many Fortran simulations must be combined to obtain cross sections and rates. Python code is available to automate this.

Finally, a lot of Python code is available to visualize or otherwise process results.

For example, to start up to 16 simultaneous jobs and monitor their status:

```
python jobs/code_demo.py --calc --limit=16 --monitor --verbose
```

The jobs are completely reproducible, with each creating its own directory with copies of source and configuration to prevent side effects.

To get started, one needs:

- ifort or gfortran. Other compilers should work, but flags have to be adapted in the SCons build file.
- MPI (Message Passing Interface) library matching the compiler.
- Blas and Lapack libraries (MKL implementation was used in this work).
- All libraries included in the `LD_LIBRARY_PATH` environment variable.
- Python 2 (standard on most linux distributions).
- The SCons build tool from scons.org.
- All Python packages in `...code/other/requirements.pip`.
- The Python packages and the `...code` directory included in the `PYTHONPATH` environment variable.
- Possibly change configuration in `...code/run/settings.py` or in environment, e.g. `CALC_DIR`.

There are many jobs in `...code/jobs/` that can be used as reference. They can be started from the command line as shown above.

6.4 Changes

This is a summary of the changes made to the software as part of this project.

6.4.1 Obstacles & improvements

This section offers a short overview of some of the bigger obstacles encountered and improvements made. Reading this is not important for understanding the rest of this report.

- Various parameters now have their values determined partially or entirely automatically. This drastically reduces the search space while tuning.
  - Perhaps most important, the timestep is now determined automatically using the Lanczos algorithm to determine the spectral range, see section 2.10.3 (Range of eigenvalues). This was a hard to tune parameter.
  - All the parameters from section 6.2.4 (Dependent parameters): `R_abs_flux_dist`, `wp_R_center`, `prob_E_min`, `prob_E_max`, `R_flux_point`, `R_abs_pos`, `R_dvr_max`, `time_step_min` and `time_step_max`.
  - Instead of providing the initial wavepacket position, the flux surface, the absorption region and the end of the grid for R (in different units), they can be provided as (fractions or) relative positions.
  - One can provide grid spacings instead of grid point counts, for r, R, and the output energy grid. It is likely that one wants to keep the same grid point density even if the grid size changes, so this saves effort.
  - One can provide integration time in atomic units instead of iterations, since step size is hard to know beforehand, and time is more meaningful: the duration of a chemical interaction should not depend on simulation implementation.
  - The norm below which the integration is considered complete was implemented (the parameter `norm_stop_crit` existed, but did not do anything before).
  - There is logic to choose an absorption strength based on a certain percentage absorbed. This is approximate, since this happens before the timestep is known exactly.
  - The size of the angular basis can be set relative to the total angular momentum.
- A lot of parameter tuning happened to get physical results.
- A lot more parameter tuning to achieve good performance.
- To determine which features most affect the the success of results, Random Forest classification was used. This was done by predicting known outcomes, and looking at which features the random forest model gave most weight[43]. Results were mixed.
- Fortran code has been converted from fixed-form Fortran77 to free-form Fortran90: resolved the many compiler warnings, indentation was completely overhauled, syntax was modernized, got rid of all implicitly defined variables and data blocks, precisions is no longer hard-coded, got rid of header files, use '`modulo`' function instead of '`(x/2)*2 != x`', '`>`' instead

of '`.gt.`', '`!`' instead of '`c...`', added in/out intent in some places, add parameter names at calling sites if confusing, removed `goto`s, used string formatting and using stderr output for better logging.

- The Fortran code was organized into modules, making all dependencies explicit so that function calls can be checked at compile time. This includes blas/lapack routines.
- Unused code was removed, e.g. code for outdated machine architectures, things that are no longer relevant or reachable, and commented-out code. For example, cray architecture, diagdvr, parts of files.
- The effects of all parameters were documented.
- After computation, results are extracted from various log files and combined into Python data structures. These also include input parameters for easy visualization.
- Results are augmented with derived properties.
- The code is in under source control in a git repository so that the history is easily available.
- Easily construct jobs for ranges of parameter values.
- Easily construct and manage all the calculations needed for a specific J_{max} value (`ShefJsumJob`).
- Because of a very elusive, non-deterministic problem with invalid flux values:
  - Checks were added so that these invalid values cannot accidentally be used in results.
  - For analysis code: all arrays initialized as signaling NaN, turned on crash on NaN, checked all parameters for NaNs, compile Blas/Lapack in debug mode.
  - Found a bug with partly uninitialized array (`stateres_edep_nonreact_prob_partial`), which was resolved.
  - Most file writing during the simulation was changed to a scratch disk instead of NFS, and are copied back afterwards if necessary.
  - All write operations, even to different files, were made sequential. Because the problem, while non-deterministic, has a strong correlation with J.
  - Flux writes and reads were changed from Fortran-unformatted to plain text, since the NaNs were eventually found to originate here. That is, there were no NaN values before writing or in the file, but they were appeared after reading, usually near each other.

  After this last change, the problem disappeared, so this has been made the default. No satisfactory explanation was found for why this caused the problem.

There were also problems with wave function dumps in Fortran unformatted, these were also changed to a new format (plain binary). These are only for visualization, they are not used to calculate cross sections.

- A lot of time was spend looking for the source of small numerical differences, which were eventually found to be caused by NAG and MKL being slightly different. The problem is very minimal if the grid spacing is large enough.
- Solved a problem with aliasing in `mpi_allreduce` when calculating lowest rovibrational energy
- Reproducibility was improved. Each job now has an independent directory from which it can be ran without any further configuration. There is also an error if parameters change after the job directory is created (so that the job and results always correspond the input chosen at the start).
- The build process was rewritten from Makefile to SCons (scons.org), removing 5 levels of recursion and adding features and robustness.
- MPI use has been made more robust: it was easy to run with 1 process by accident and get incorrect but reasonable results.
- The ability to detect problems early in the computation was added for several scenarios, saving computation time and getting faster feedback for failed jobs.
- Code was added to run everything from a physical disk instead of network disk automatically, but copy results back. This was necessary due to unconfirmed NFS issues.
- Improved compiler flags (gfortran, ifort) for better performance and error detection.
- Logging was expanded and improved. Quite some debug logging is only shown when compiled in debug mode.
- Renamed identifiers and added quite some comments in the Fortran code for clarity.
- Add benchmarking for the various steps in the propagation, compensated for slower and faster computer nodes.
- Automatic compression and conversion of results, and moving of non-essential files to scratch disk, because of quota issues.
- Functionality to detect deadlocks and terminate the job (useful during debugging; there should currently be no more deadlocks).
- Multithreading of blas/lapack operations interfered with the queue system and was actually slower when combined with MPI parallelism, so it was removed.
- Code added to extract which part of the output probabilities and energies are sufficiently converged, as judged by the cumulative probability.

- Code added to calculate the aggregated cross sections from the probabilities of many individual jobs.
- Infrastructure was created in the form of a reusable library (fenpei[40]), to easily run and compare many jobs. Reproducibly starting or analyzing several dozen jobs to compare a parameter takes only minutes and is very reliable.
- Added code to write channel energies and norms to file.
- Added functionality to extract and visualize wavefunction density during the propagation.
- Compared two methods of selecting which range of probabilities to use: based on wavefunction spread, or based on norm deviations.
- Various different visualizations and ways to process output were created. To highlight a few of the possibilities:
  - Analyze stderr log messages and try to determine what went wrong. Correlate these problems with input parameters.
  - Stacked overview with a different color for each output channel (Figure 18). Can be renormalized, and can combine different calculations.
  - 1D, 2D and mutual difference plots to compare the effect of any scalar input parameter on any output parameter.
  - Previews to determine which part of the output energy grid is usable.
  - Functions to show the desired results: cross sections and reaction rates.
  - The effect of partial waves on transition probability, optionally per channel, either at fixed energy or as a surface for all energies (Figure 16), with degeneracy optionally included.
  - Bar plot of energies involved: kinetic, rotational, vibrational, centrifugal: Figure 13.
  - Performance as total time, computation time per unit simulated time, or split up for different parts of the calculation (Figure 30).
  - Visualize wavefunction during the propagation, projected on any two basis quantities (R, r, j, Ω)[9].
  - Show indicators of convergence, like cumulative flux absorption and remaining wavefunction norm.
  - All the other subplots in Figure 35. This plot is interactive: individual results can be shown or hidden, and reference energy can be changed.
  - A non-visual function to automatically test for anomalies in the output that could indicate that something has gone wrong.

[9] Visualizing the wavefunction needs large dump files, which are only produced when turned on through a parameter.

- A downside is that some changes, in particular scaling $\hat{H}$ automatically, led to the inability to restart jobs that had crashed. Restart easily lead to results that are wrong in subtle ways, and they are not very common: if a job crashes, it is likely that it must be changed, invalidating stored restart data. Failures due to hardware or power failures are very rare these days, so little effort was invested into restoring restart functionality.

It is perhaps fair to say that the refactoring and usability expansion efforts have been somewhat excessive, having consumed too much time. However, the code should now be much more convenient and efficient to work with and extend.

Fairly extensive lab notes are also available upon request.

6.5 Data

The raw output is attached in compressed form (13GB), which includes input, log files, and output for each of the around 15.500 runs. This is the format that the Python code reads.

The data about each of the jobs is semicolon-separated format in `probabilities.csv`. The probabilities themselves are in one binary file per job, totaling about 7GB (details in `probabilities.csv`).

The same format is also used for the cross sections extracted from these probabilities, in `cross_sections.csv`.

This data took several CPU-years to compute. This data is stored by the Theoretical Chemistry department of the Radboud University Nijmegen.

References

(1) Song, L.; Balakrishnan, N.; van der Avoird, A.; Karman, T.; Groenenboom, G. C. *The Journal of Chemical Physics* **2015**, *142* 04303, DOI: http://dx.doi.org/10.1063/1.4921520.

(2) Van Dishoeck, E. F. *Faraday Discuss.* **2014**, *168*, 9–47.

(3) McBane, G. C.; Kable, S. H.; Houston, P. L.; Schatz, G. C. *The Journal of Chemical Physics* **1991**, *94* 1141-1149, 1141–1149.

(4) Green, S.; Pan, B.; Bowman, J. M. *The Journal of Chemical Physics* **1995**, *102* 8800-8806, 8800–8806.

(5) Green, S.; Keller, H.; Schinke, R.; Werner, H. *The Journal of Chemical Physics* **1996**, *105* 5416-5422, 5416–5422.

(6) Chu, S.-I.; Dalgarno, A. *Proc. R. Soc. A* **1975**, *342*, 191.

(7) Werner, H.-J.; Bauer, C.; Rosmus, P.; Keller, H.-M.; Stumpf, M.; Schinke, R. *J. Chem. Phys.* **1995**, *102*, 3593.

(8) Keller, H.-M.; Floethmann, H.; Dobbyn, A. J.; Schinke, R.; Werner, H.-J.; Bauer, C.; Rosmus, P. *J. Chem. Phys.* **1996**, *105*, 4983.

(9) Green, S.; Pan, B.; Bowman, J. M. *J. Chem. Phys.* **1995**, *102*, 8800.

(10) Green, S.; Keller, H.-M.; Schinke, R.; Werner, H.-J. *J. Chem. Phys.* **1996**, *105*, 5416.

(11) Yang, B.; Stancil, P. C.; Balakrishnan, N. *J. Chem. Phys.* **2005**, *123*, 094308.

(12) Shepler, B. C.; Yang, B. H.; Kumar, T. J. D.; Stancil, P. C.; Bowman, J. M.; Balakrishnan, N.; Zhang, P.; Bodo, E.; Dalgarno, A. *Astron. Astrophys.* **2007**, *475*, L15.

(13) Von Rosenberg, C. W.; Taylor, R. L.; Teare, J. D. *J. Chem. Phys.* **1971**, *54*, 1974.

(14) Glass, G. P.; Klronde, S. *J. Phys. Chem.* **1982**, *86*, 908.

(15) Kozlov, P. V.; Makarov, V. N.; Pavlov, V. A.; Shatalov, O. P. *Shock Waves* **2000**, *10*, 191.

(16) Ndengue, S. A.; Dawes, R. In *71st International Symposium on Molecular Spectroscopy*, 2016.

(17) Walker, K.; Song, L.; Yang, B.; Groenenboom, G.; van der Avoird, A.; Naduvalath, B.; Forrey, R.; Stancil, P. In *APS Division of Atomic, Molecular and Optical Physics Meeting Abstracts*, 2015.

(18) Song, L.; Balakrishnan, N.; Walker, K. M.; Stancil, P. C.; Thi, W. F.; Kamp, I.; van der Avoird, A.; Groenenboom, G. C. *The Astrophysical Journal* **2015**, *813*, 96.

(19) Song, L. Ab Initio Study of Inelastic Collision Processes for Astrochemical Applications., (PhD thesis), Ph.D. Thesis, Radboud University Nijmegen, 2016.

(20) Meijer, A. J.H. M.; Goldfield, E. M. *The Journal of Chemical Physics* **1998**, *108*, 5404–5413.

(21) Laganà, A.; Lendvay, G., *Theory of Chemical Reaction Dynamics*; NATO science series: Mathematics, physics, and chemistry; Springer: 2004; Chapter Timedependent wavepacket calculations for reactive scattering (chapter 8).

(22) Gonsalves, R. J. Rotations and Rigid Body Motion. http://www.physics.buffalo.edu/phy302/topic3/ (accessed 02/19/2017).

(23) Pack, R. T. *The Journal of Chemical Physics* **1974**, *60*, 633–639.

(24) Piela, L., *Ideas of Quantum Chemistry, Appendix I: Space- and Body-Fixed Coordinate Systems*; Elsevier: 2013.

(25) Balint-Kurti, G. G.; Palov, A., *Theory of Molecular Collisions*; Theoretical and Computational Chemistry Series; The Royal Society of Chemistry: 2015, P001–282.

(26) Meijer, A. J.H. M.; Goldfield, E. M. *The Journal of Chemical Physics* **1999**, *110*, 870–880.

(27) Song, L.; van der Avoird, A.; Groenenboom, G. C. *The Journal of Physical Chemistry A* **2013**, *117*, PMID: 23597133, 7571–7579.

(28) *Tutorials in Molecular Reaction Dynamics*; Brouard, M., Vallance, C., Eds.; The Royal Society of Chemistry: 2012.

(29) Light, J. C.; Carrington, T. In *Advances in Chemical Physics*; John Wiley and sons: 2007, pp 263–310.

(30) Balint-Kurti, G. G. *International Reviews in Physical Chemistry* **2008**, *27*, 507–539.

(31) Schatz, G. C. *The Journal of Physical Chemistry* **1996**, *100*, 12839–12847.

(32) Golub, G. H.; Van Loan, C. F., *Matrix Computations (3rd Ed.)* Johns Hopkins University Press: Baltimore, MD, USA, 1996.

(33) Walker, K. M.; Song, L.; Yang, B. H.; Groenenboom, G. C.; van der Avoird, A.; Balakrishnan, N.; Forrey, R. C.; Stancil, P. C. *The Astrophysical Journal* **2015**, *811*, 27.

(34) Gray, S. K.; Balint-Kurti, G. G. *The Journal of Chemical Physics* **1998**, *108*, 950–962.

(35) Corden, M. Speed of REAL*8 and COMPLEX*8 number multiplication. https://software.intel.com/en-us/forums/intel-visual-fortran-compiler-for-windows/topic/71486 (accessed 06/21/2017).

(36) Chen, R.; Guo, H. *Computer Physics Communications* **1999**, *119*, 19 –31.

(37) Billing, G. *Computer Physics Communications* **1984**, *32*, 45 –62.

(38) Hui, Z. J. Z., *Theory and application of quantum molecular dynamics*; World Scientific: 1998.

(39) Goldfield, E. M.; Meijer, A. J.H. M. *The Journal of Chemical Physics* **2000**, *113*, 11055–11062.

(40) Verleg, M. Fenpei. https://github.com/mverleg/fenpei (accessed 06/23/2019).

(41) Groenenboom, G. C.; Colbert, D. T. *The Journal of Chemical Physics* **1993**, *99*, 9681–9696.

(42) Goldfield, E.; Gray, S. *Computer Physics Communications* **1996**, *98*, 1 –14.

(43) Rogers, J.; Gunn, S. In *Subspace, Latent Structure and Feature Selection*, ed. by Saunders, C.; Grobelnik, M.; Gunn, S.; Shawe-Taylor, J., Springer Berlin Heidelberg: Berlin, Heidelberg, 2006, pp 173–184.

(44) *Theoretical Chemistry: Theory of Scattering, Papers in Honor of Henry Eyring*, 21 Jan 2016; Henderson, D. W., Ed.; Elsevier: 2016.