atoms

MDPI

Article

PyQCAMS: Python Quasi-Classical Atom–Molecule Scattering

Rian Koots [1,2] and Jesús Pérez-Ríos [1,2,*]

[1] Department of Physics, Stony Brook University, Stony Brook, NY 11794, USA
[2] Institute for Advanced Computational Science, Stony Brook University, Stony Brook, NY 11794, USA
* Correspondence: jesus.perezrios@stonybrook.edu

Abstract: We present Python Quasi-classical atom–molecule scattering (PyQCAMS v0.1.0), a new Python package for atom–diatom scattering within the quasi-classical trajectory approach. The input consists of the mass, collision energy, impact parameter, and pair-wise/three-body interactions. As the output, the code provides the vibrational quenching, dissociation, and reactive cross sections along with the rovibrational energy distribution of the reaction products. We benchmark the program for a reaction involving a molecular ion in a high-density ultracold gas, $RbBa^+ + Rb$. Furthermore, we treat $H_2 + Ca \rightarrow CaH + H$ reactions as a prototypical example to illustrate the properties and performance of the software. Finally, we study the parallelization performance of the code by looking into the speedup of the program as a function of the number of CPUs used.

Keywords: atom–molecule scattering; quasi-classical trajectory calculations; molecular dissociation; vibrational quenching; reactive scattering

1. Introduction

For decades, one of the main approaches to studying molecular dynamics has been via the quasi-classical trajectory (QCT) method [1,2]. This technique treats collisions semi-classically. The nuclear dynamics in the underlying potential energy surface are treated classically. However, the initial and final states are chosen following the Bohr–Sommerfeld quantization rule, yielding accurate predictions at significantly less cost than quantum dynamics as long as they fall within given conditions—i.e., a high collision energy with a large number of contributing partial waves [3]. QCT has been used in a multitude of scenarios relevant to chemical physics [4–10] and cold and ultracold chemistry [11–13], ranging from the ultracold to the hyperthermal regimes. In particular, it has been used to study the relaxation and reaction dynamics of cold atom–ionic molecule [11,12] and atom–molecule collisions [14].

We present an open-source object-oriented program written in Python to perform QCT calculations on atom–diatom systems called Python Quasi-Classical Atom–Molecule Scattering (PyQCAMS). While chemical dynamics programs such as Gaussian [15] and VENUS [16] are capable of performing QCT calculations, we introduce a more accessible, user-friendly, and dedicated path to performing these simulations. Our program consists of completely open-source software and relies mainly on the NumPy [17] and SciPy [18] packages. We use matplotlib [19] for data visualization, pandas [20] for data storage and analysis, and multiprocess [21,22] for parallel implementation.

The outline of this paper is as follows: In Section 2.1 we discuss the theory behind the quasi-classical trajectories method, including the initial conditions, trajectory reactions, and analysis. In Section 2.2, we discuss the PyQCAMS program as separated into the inputs, main code, and outputs. We also discuss the implementation and performance of the program. In Section 3, we provide a benchmark study on the reaction $RbBa^+ + Rb$ as well as an example of a typical workflow to study the reaction $H_2 + Ca$, where we outline how a user can obtain reaction rates and product distributions using the program.

check for updates

Citation: Koots, R.; Pérez-Ríos, J. PyQCAMS: Python Quasi-Classical Atom–Molecule Scattering. *Atoms* **2024**, *12*, 29. https://doi.org/10.3390/atoms12050029

Received: 22 January 2024
Revised: 9 April 2024
Accepted: 7 May 2024
Published: 11 May 2024

2. Materials and Methods

2.1. Theoretical Approach

In this section, we describe the basics of the QCT. The QCT method treats scattering processes semi-classically. The trajectories are calculated by solving Newton's equations of motion of the colliding nuclei. In the case of atom–molecule scattering, at the start of each trajectory, the rovibrational spectrum is calculated following a discrete variable representation (DVR) method using particle-in-a-box eigenfunctions [23]. The internal degrees of freedom of the final molecule are treated within the Wentzel–Kramers–Brillouin (WKB) approximation, such that the final rovibrational state (v', j') satisfies a quantum mechanically viable state. The classical Hamiltonian for a 3-particle atom–molecule system with masses m_i, $i = 1, 2, 3$ takes the following form:

$$H = \sum_{i=1}^{3} \frac{\vec{p}_i^2}{2m_i} + V(\vec{r}_1, \vec{r}_2, \vec{r}_3), \tag{1}$$

where $\vec{p}_i$ and $\vec{r}_i$ represent the momentum and position vectors of each atom with respect to the origin. This Hamiltonian is expressed in Jacobi coordinates as

$$H = \frac{\vec{P}_1^2}{2\mu_{12}} + \frac{\vec{P}_2^2}{2\mu_{3,12}} + V(\vec{\rho}_1, \vec{\rho}_2), \tag{2}$$

where $\vec{\rho}_1$ is the Jacobi vector associated with the molecule, and $\vec{P}_1$ is its conjugate momentum. $\vec{\rho}_2$ is the Jacobi vector connecting the atom with the center of mass of the molecule, and $\vec{P}_2$ is its conjugate momentum, as shown in Figure 1. Note that this Hamiltonian uses the reduced masses of the corresponding atoms: $\mu_{12} = (\frac{1}{m_1} + \frac{1}{m_2})^{-1}$ and $\mu_{3,12} = (\frac{1}{m_3} + \frac{1}{m_1+m_2})^{-1}$. Defining the coordinates in this way separates the center-of-mass degree of freedom from the relative one. The momentum associated with the center-of-mass degrees of freedom is a constant of motion, since the interaction potentials do not depend on the center-of-mass position, and is neglected in the analysis of the dynamics.

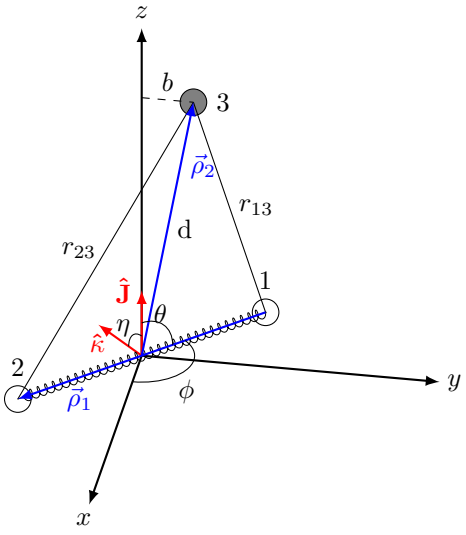

Figure 1. Jacobi coordinates of an atom–molecule system. The Jacobi vectors $\vec{\rho}_1$ and $\vec{\rho}_2$ are in blue. Here, the molecule is rotating in the x-y plane, with the angular momentum $\vec{J}$ along the z-axis, $\vec{\kappa} = \vec{\rho}_1 \times \hat{z}$. η is the angle between $\vec{\kappa}$ and $\vec{J}$, and ϕ and θ are defined as usual in spherical coordinates.

We trace the atoms' subsequent motion by solving Hamilton's equations of motion:

$$\frac{d\rho_{i,\alpha}}{dt} = \frac{\partial H}{\partial P_{i,\alpha}} \tag{3}$$

$$\frac{dP_{i,\alpha}}{dt} = -\frac{\partial H}{\partial \rho_{i,\alpha}} \tag{4}$$

for $i = 1, 2$ for each associated vector, and $\alpha = 1, 2, 3$ for each Cartesian component.

2.1.1. Initial Conditions

The initial orientation angles are randomly generated to initialize each trajectory. The momentum of the molecule, $\vec{P}_1$, is subsequently defined by the orientation angles. If we initialize the molecule at its classical outer turning point $|\vec{\rho}_1| = r_+$, the momentum has no radial component. Since the angular momentum $\vec{J} = \vec{r} \times \vec{p}$, and $\vec{\rho}_1 \perp \vec{P}_1$, we find the magnitude of $P_1 = \hbar\sqrt{j(j+1)}/r_+$, with the following components [3]:

$$\vec{P}_1 = P_1 \begin{pmatrix} \sin\phi\cos\eta - \cos\theta\cos\phi\sin\eta \\ -\cos\phi\cos\eta - \cos\theta\sin\phi\sin\eta \\ \sin\theta\sin\eta \end{pmatrix}. \tag{5}$$

The angles ϕ, θ, η are defined in Figure 1. The relative momentum between the colliding atom and the molecule center for a given collision energy, E_c, is $\vec{P}_2 = \sqrt{2\mu_{3,12}E_c}$. The initial vibrational phase is randomly sampled by choosing the initial distance between the atom and molecule as

$$R = R_0 + \frac{\zeta P_2 \tau_{v,j}}{\mu_{3,12}}, \tag{6}$$

where R_0 is a fixed far-away distance, where the interaction potential strength is negligible. The second term probes the molecule's vibrational phase since $\zeta \in [0, 1]$ is randomly generated following a uniform distribution, and $\tau_{v,j}$ is the vibrational period of the molecule corresponding to the quantum mechanical state (v, j). This is calculated as

$$\tau_{v,j} = \sqrt{2\mu_{12}} \int_{r_-}^{r_+} \left[E_{int} - V(r_{12}) - \frac{\hbar^2 j(j+1)}{2\mu_{12} r_{12}^2} \right]^{-\frac{1}{2}} dr_{12}, \tag{7}$$

where $V(r_{12})$ is the molecular potential energy, and r_{12} is the molecular separation. E_{int} is the internal energy of the molecule, which is calculated via a DVR method. The bounds of the integral are given by the classical inner and outer turning points of the bound state, r_- and r_+, respectively.

2.1.2. Final Products

For the atom–molecule reaction AB + C, we expect three possible final products (see Figure 2):

1. Inelastic collision (quenching): AB(v, j) + C $\rightarrow$ AB(v', j') + C;
2. Molecular formation (reaction): AB + C $\rightarrow$ AC + B or BC + A;
3. Dissociation: AB + C $\rightarrow$ A + B + C.

The final product is determined by the relative energy between each atom. The condition for whether two atoms are bound is determined by the effective potential:

$$E_{ij} < \begin{cases} V_{ij}(r_0) + \frac{j'(j'+1)}{2\mu_{ij} r_0^2} & j' \neq 0 \\ 0 & j' = 0 \end{cases}, \tag{8}$$

where r_0 is the local maximum of the effective potential. Here, $i, j \in [1, 3]$, $i \neq j$ represent each of the atoms. Two atoms are considered bound only if this condition is satisfied for just one pair of atoms, so that the other two pairs can be deemed unbound.

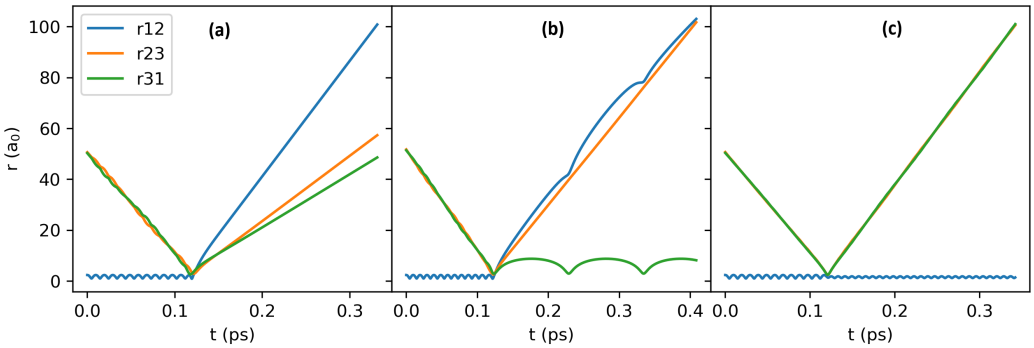

Figure 2. Different product outcomes for a given QCT calculation for H_2 + Ca at E_c = 50,000 K, b = 0 a_0. r_{12} represents the internuclear distance between each H, whereas r_{32} and r_{31} represent the internuclear distance between the colliding atom Ca (3) and each molecular atom (H(1) and H(2)). These trajectories show dissociation (**a**), reaction (**b**), and quenching (**c**). In panel (**b**), r_{31} oscillates around a fixed distance, an indicator that a new molecule is formed.

The final states (v', j') are calculated within the WKB approximation. The rotational quantum number is given by

$$j' = -\frac{1}{2} + \frac{1}{2}\sqrt{1 + 4\frac{\vec{J'} \cdot \vec{J'}}{\hbar^2}}, \tag{9}$$

where $\vec{J} = \vec{\rho}_i \times \vec{P}_i$ is the effective conjugate angular momentum of the Jacobi vector $\vec{\rho}_i$. This expression is equivalent to $\vec{J} \cdot \vec{J} = j(j+1)\hbar^2$. After rounding j' to the nearest integer number, the vibrational quantum number is given by [2]

$$v' = -\frac{1}{2} + \frac{\sqrt{2\mu_{ab}}}{\pi\hbar}\int_{r_-}^{r_+}\left[E'_{int} - V(r_{ab}) - \frac{\hbar^2 j'(j'+1)}{2\mu_{ab}r_{ab}^2}\right]^{\frac{1}{2}} dr. \tag{10}$$

Although these equations lead to continuous numbers, they need to be interpreted as integers before assigning them as quantum numbers. One may choose to round to the nearest number via histogram binning (HB) or through the Gaussian binning (GB) process, where each rovibrational action $(x = v', j')$ is weighted against its nearest integer x_t with a Gaussian [24,25]:

$$W(x', x_t) = \frac{1}{\sigma\sqrt{\pi}} e^{\frac{-|x'-x_t|^2}{\sigma^2}} \tag{11}$$

with $\sigma = 0.05$. The Gaussian weight of a trajectory ending in (v', j') is

$$W(v', j') = W(v', v_t)W(j', j_t). \tag{12}$$

2.1.3. Analysis

The cross-section and reaction rate coefficients for an atom–molecule collision are calculated from the QCT simulations. The first step is to calculate the opacity function, which is formally defined as

$$P_{q,r,d}(E_c, b) = \int P_{q,r,d}(E_c, b, \theta, \phi, \eta, \chi)d\Omega, \tag{13}$$

where $d\Omega = \sin\theta d\theta d\phi d\eta d\chi$, and $P_{q,r,d}$ represent the opacity function of a quenching, reaction, and dissociation product, respectively. This integral is evaluated via the Monte Carlo

technique over the randomly generated variables θ, ϕ, η, and χ, with an error associated with one standard deviation:

$$P_{q,r,d}(E_c, b) = \frac{N_{q,r,d}(E_c, b)}{N} \pm \frac{\sqrt{N_{q,r,d}(E_c, b)}}{N} \sqrt{\frac{N - N_{q,r,d}(E_c, b)}{N}}. \quad (14)$$

The final state-specific opacity of a reaction initialized in state (v_i, j_i) with collision energy E_c and impact parameter b, and ending in state (v', j') is calculated via the GB method [26]:

$$P_r(E_c, b; v', j') = \frac{\sum_{i=1}^{N_r} W_r^i(v', j')}{W} \pm \frac{\sqrt{\sum_{i=1}^{N_r} W_r^i(v', j')}}{W} \sqrt{\frac{W - \sum_{i=1}^{N_r} W_r^i(v', j')}{W}}, \quad (15)$$

where N_r is the number of reactive trajectories, and the total weight W is evaluated as

$$W = \sum^{N} W(v', j') + N_d, \quad (16)$$

where N is the total number of trajectories, and N_d is the number of dissociation trajectories. Each dissociation reaction has a weight $W_d = 1$ since there is no rovibrational state associated with this product.

The opacity function yields the state-specific scattering cross section at different collision energies E_c:

$$\sigma_{q,r,d}(E_c; v', j') = 2\pi \int_0^{b_{\max}^{q,r,d}} P_{q,r,d}(E_c, b; v', j') b db, \quad (17)$$

where $b_{\max}$ is defined such that $P(E_c, b_{max}) = 0$ for a given reaction channel. The rate coefficient is given by

$$k_{q,r,d}(E_c) = \sigma_{q,r,d}(E_c) \sqrt{\frac{2E_c}{\mu}}. \quad (18)$$

2.2. The Program

PyQCAMS is written in an object-oriented manner, containing two main classes: `Molecule` and `Trajectory`. This treats molecules and trajectories as objects, tracing relevant attributes for each calculation, as outlined in Figure 3. The program takes an input, where the user specifies the details of the reaction of interest. The variables are passed into their respective classes, where the trajectory calculations are performed. The results of each trajectory are then output into a user specified file. The details of each step are outlined in this section.

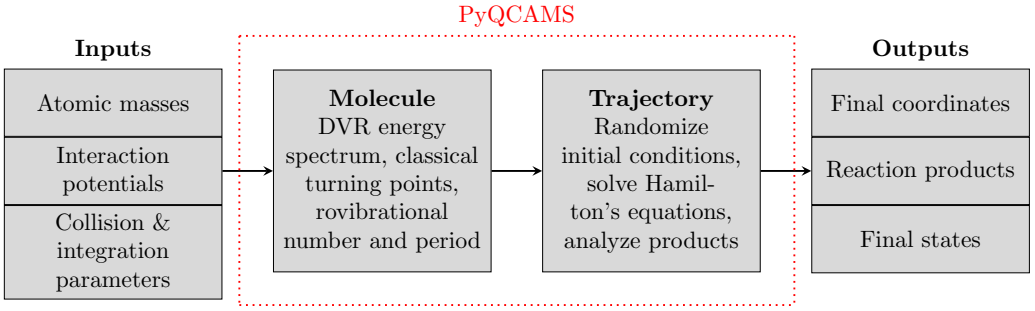

Figure 3. The general structure of the PyQCAMS program.

2.2.1. Input

For a given system, a new Python script should be created with a dictionary containing the calculation input, as shown in Listing 1. The script contains the masses of all three atoms, where atoms 1 and 2 form the initial molecule, and atom 3 is the colliding atom. Each trajectory is defined by the collision energy (E0) in Kelvin, the impact parameter (b0) in Bohr, and the initial distance between the atom and molecule (R0) in Bohr.

Listing 1. Python script for a full QCT calculation of the reaction H_2 + Ca. This example uses the Morse function from the potentials module, with appropriate parameters set. v12 always refers to the initial bound molecule. v123 is the three-body potential function, with associated derivatives. This example uses the Axilrod–Teller model potential but with a coefficient of 0; so, it includes no three-body effects.

```
import numpy as np
from pyqcams import qct, constants, potentials

# Atomic masses (amu converted to atomic units)
m1 = 1.008*constants.u2me # H
m2 = 1.008*constants.u2me # H
m3 = 40.078*constants.u2me # Ca

# Collision parameters
E0 = 40000 # collision energy (K)
b0 = 0 # Impact parameter (Bohr)
R0 = 50 # Initial distance (Bohr)

# Potential parameters in atomic units
v12, dv12 = potentials.morse(de = 0.16456603489, re = 1.40104284795, alpha =
    1.059493476908482)
v23, dv23 = potentials.morse(de = 0.06529228457, re = 3.79079033313, alpha =
    0.6906412379896358)
v31, dv31 = potentials.morse(de = 0.06529228457, re = 3.79079033313, alpha =
    0.6906412379896358)

# Three-body interaction
v123, dv123dr12, dv123dr23, dv123dr31 = potentials.axilrod(C=0)

# Initiate molecules
mol12 = qct.Molecule(mi = m1, mj = m2, Vij = v12, dVij = dv12,
                     xmin = .5, xmax = 30, vi = 1, ji = 0, npts=1000)
mol23 = qct.Molecule(mi = m2, mj = m3, Vij = v23, dVij = dv23,
                     xmin = 1, xmax = 40)
mol31 = qct.Molecule(mi = m3, mj = m1, Vij = v31, dVij = dv31,
                     xmin = 1, xmax = 40)

input_dict = {'m1':m1, 'm2':m2, 'm3':m3,
    'E0': E0, 'b0': b0, 'R0': R0, 'seed': None,
    'mol_12': mol12,'mol_23': mol23,'mol_31': mol31,
    'vt': v123, 'dvtdr12': dv123dr12,
    'dvtdr23': dv123dr23, 'dvtdr31': dv123dr31,
    'integ':{'t_stop': 2, 'r_stop': 2, 'r_tol': 1e-10,
        'a_tol':1e-8,'econs':1e-5,'lcons':1e-5}}

if __name__ == '__main__':
    bs = np.arange(0,5,0.25) # Range of impact parameters (Bohr)
    nTraj = 10000 # Number of trajectories
    short_out = 'short.txt' # Short output
    long_output = 'long.txt' # Long output

    # Run over a range of impact parameters
    for b in bs:
        input_dict['b0'] = b
        qct.runN(nTraj, input_dict, short_out = 'short.txt',
                long_out = 'long.txt')

    # Analysis
```

```
mu = m3*m1*m2/(m1+m2+m3)
analysis.opacity('samplelong.txt',GB=True,vib=True,rot=False,output='
    opacity.txt')
analysis.crossSection('samplelong.txt',GB=True,vib=True,rot=False,output=
    'sigma.txt')
analysis.rate('samplelong.txt',mu=mu,GB=True,vib=True,rot=False,output='
    rate.txt')
```

Next, the interaction potential between each atom and their derivatives is defined in atomic units. The user should define all two-body and three-body interactions (and their derivatives). These can be user-defined functions or chosen from the `potentials` module, which has a selection of analytical potentials and their derivatives for two-body and three-body interactions. One may fit ab initio potential curves and surfaces to a functional form for the trajectory calculation [27], which includes both two and three-body interactions. This functional form is provided in `potentials` as `poly2` and `poly3`, inspired by [28]. The potentials offered in `potentials` are

1. Two-Body Terms
   (a) Morse: $V(r) = D_e\left(1 - \exp^{(-\alpha(r-r_e))}\right)^2 - D_e$
   User defines: D_e, α, r_e;
   (b) Generalized Lennard–Jones: $V(r) = C_m/r^m - C_n/r^n$
   User defines: m, n, C_m, C_n;
   (c) Buckingham: $V(r) = ae^{-br} - C_6/r^6$
   User defines: a, b, C_6;
   (d) poly2: $V(r) = \frac{C_0 e^{-\alpha r}}{r} + \sum_{i=1}^{N} c_i \rho^i, \rho = re^{-\beta r}$
   User defines: C_0, α, β, c_i.
2. Three-Body Terms
   (a) Axilrod–Teller [29]: $V(r_{12}, r_{23}, r_{31}) = C\frac{3\cos\gamma_1\cos\gamma_2\cos\gamma_3+1}{r_{12}^3 r_{23}^3 r_{31}^3}$
   User defines: C;
   (b) poly3: $V(r_{12}, r_{23}, r_{31}) = \sum_{i,j,k}^{M} d_{ijk}\rho_{12}^i\rho_{23}^j\rho_{31}^k, \rho_{\mu\nu} = r_{\mu\nu}e^{-\beta_{\mu\nu}r_{\mu\nu}}$
   User defines: $d_{ijk}, \beta_{\mu\nu}$.

Three molecules representing the three possible reaction products should be initiated via the molecule class, with the following keyword arguments ('mi', 'mj', 'Vij', 'dVij', 'xmin', 'xmax'). Here, 'mi,mj' are the masses of atoms 'i' and 'j', 'Vij,dVij' are the interaction potential and its derivative between those atoms, and 'xmin,xmax' represent the range of the interaction potential. Note that since molecule '12' is initially bound, one should also define their initial vibrational and rotational quantum numbers, 'vi,ji'. A DVR method is used to calculate the energy corresponding to this rovibrational state, so the user can control the number of DVR points with the keyword 'npts', which is set to 1000 by default. If the rovibrational spectrum is known, the DVR method can be skipped by setting the binding energy of the molecule using the keyword argument 'Ei'.

The main input requirement of PyQCAMS is a dictionary containing all of the previously defined parameters necessary for a QCT calculation. In addition, the dictionary should contain the three-body interaction term 'Vt' and its partial derivatives 'dVtdr12, dVtdr23, dVtdr31'. To neglect the three-body interaction, just set these functions to 0. Finally, the parameters for integration should be entered in the nested dictionary 'integ'. The user can choose when to stop a trajectory using 't_stop', which is a multiplicative factor of the trajectory's time scale. Another stopping condition is the maximum distance between any two atoms, 'r_stop', which

is a multiplicative factor of the initial distance 'R0'. Finally, the absolute and relative tolerances of the integrator can be controlled with 'a_tol' and 'r_tol', respectively. For reproducibility, the user can input a 'seed' for the random number generator or leave it as null to generate a new trajectory. The keywords for this dictionary are described in Table 1.

Table 1. Keywords for the input dictionary required in the input file of PyQCAMS.

Keyword	Description (Unit)
m1, m2, m3	Atomic masses (a.u.)
E0	Collision energy (Kelvin)
b0	Impact parameter (Bohr)
R0	Initial distance (Bohr)
seed	Random number generator seed
mol_12	Molecule class object: of atoms one and two (initially bound)
mol_23	Molecule class object: of atoms two and three
mol_31	Molecule class object: of atoms three and one
Vt	Three-body interaction term V_t (Hartree)
dVtdr12	Partial derivative $\partial V_t/\partial r_{12}$ (Hartree)
dVtdr23	Partial derivative $\partial V_t/\partial r_{23}$ (Hartree)
dVtdr31	Partial derivative $\partial V_t/\partial r_{31}$ (Hartree)
integ[t_stop]	Stopping condition: multiple of collision timescale
integ[r_stop]	Stopping condition: multiple of R0
integ[r_tol]	Relative error tolerance for equation of motion
integ[a_tol]	Absolute error tolerance for equation of motion
integ[econs]	Conserved energy requirement (Hartree)
integ[lcons]	Conserved momentum requirement (a.u.)

Note that atoms "1" and "2" should always represent those of the initial molecule, so that atom "3" represents the colliding atom. This convention should also be followed when specifying the masses. Additionally, all values entered in the input file should be in atomic units except for the collision energy.

2.2.2. Main Code

The main code consists of two classes, `Molecule` and `Trajectory`. After defining the input dictionary using `Molecule` objects and other keywords, it can be passed into the `Trajectory` class, which computes a quasi-classical trajectory.

The `Molecule` class contains methods relevant to molecules, such as the DVR method for finding the energy spectrum of a bound molecule and a method for calculating the classical turning points of a bound molecule. The spectrum and turning points for different $E_{(v,j)}$ in different potentials can be stored separately for future use. The attributes of Molecule objects can be found in Table 2.

Table 2. Attributes of a PyQCAMS Molecule object other than those provided in their input. All values are in atomic units.

Keyword	Description
Ei	Initial binding energy (for molecule 12)
E	Final relative energy
Veff	Effective potential
re	Equilibrium distance
rp	Classical outer turning point
rm	Classical inner turning point
tau	Vibrational period
bdry	Local maximum of effective potential
bdx	Interparticle distance of effective potential local maximum
vPrime	Final vibrational number (not binned)
vt	Final vibrational number (rounded)
vw	Gaussian weight associated with vPrime
jPrime	Final rotational number (rounded)
jw	Gaussian weight associated with jPrime

The `Trajectory` class performs the main QCT calculation and is outlined in Figure 4. First, a random set of initial conditions is generated by the method `iCond`, following Section 2.1.1, yielding $(\vec{\rho}_1, \vec{\rho}_2, \vec{P}_1, \vec{P}_2)$. Next, the Hamilton's equations of motion are solved using a SciPy [18] integrator utilizing the adaptive Runge–Kutta 5(4) integration method [30]. After one of the stop conditions is met, the energy and momentum are checked for conservation. Finally, the product count $\vec{n}_f$ is created following Section 2.1.2. We first check each effective potential as to whether it can support bound states, then we check the final relative energies based on Equation (8). If more than one bound molecule is found, we denote it as an intermediate complex in the final output. If the trajectory results in a bound molecule, the final state $\vec{s_f}$ is calculated using the `Molecule` class method `turningPts`.

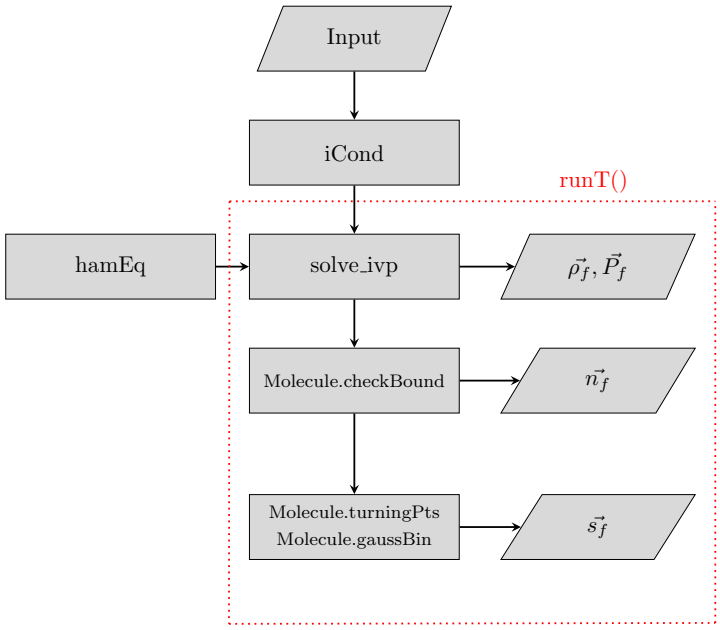

Figure 4. Outline of the `Trajectory` class. The components within the dashed line are contained in the `run_T` method of the `Trajectory` class. The outputs are the final position, momentum, product count, and state vectors. The product count vector is defined by which, if any, molecule is bound at the end of the trajectory. For a bound molecule, its final state and associated Gaussian weights produce the state vector.

2.2.3. Output

The Trajectory object has attributes (see Table 3) relevant to a trajectory that can be tracked to an output file.

Table 3. Useful attributes of the Trajectory class other than those provided in the input dictionary. While only some of these are obtained by the program by default, the user can choose to save any of these attributes when using `qct.runN`. All position, energy, momentum, and time attributes are in atomic units.

Attribute	Description
count	count vector of reaction products
fstate	state vector of reaction products
R	initial atom–diatom distance
ang	initial configuration of angles
w0	initial Jacobi vectors
wn	set of Jacobi vectors $\vec{\rho}_1, \vec{\rho}_2, \vec{P}_1, \vec{P}_2$ throughout the trajectory
t	trajectory time steps
delta_e	energy conservation
delta_l	momentum conservation

For a single trajectory, the default outputs are

1. Initial state (vi, ji, E_c, b).
2. Product count $\vec{n_f}$ = (n_{12}, n_{23}, n_{31}, n_d, n_c), where n is either 0 or 1. n_{ij} represents a bound molecule between atoms i and j, n_d represents dissociation, and n_c represents a three-atom intermediate complex at the end of the calculation.
3. Final state $\vec{s_f}$ = (v, $vw(v', v)$, j, $jw(j', j)$), where $xw(x', x)$ is the Gaussian weight given in Equation (11). For trajectories yielding $n_d = 1$ or $n_c = 1$, the final state outputs $\vec{s_f}$ = (0, 0, 0, 0).

A full QCT study requires many trajectories run over a range of impact parameters. The `qct.runN` method provides a way to run N trajectories for a given input, from which there are two modes of output:

1. Long output: Prints a single line per trajectory containing the default outputs. If any of the attributes provided in Table 3 are desired, place them as a list of strings in the 'attrs' argument of `qct.runN` method. This file is required as an input to the analysis functions provided by PyQCAMS.
2. Short output: Sums over the product count n_i of all trajectories for a given initial state (v_i, j_i, E_c, b). This can be useful for studies that do not involve state-specific results or GB.

A sample of the output files is shown in Figure 5.

vi	ji	e	b	n12	n23	n31	nd	nc	v	vw	j	jw
1	0	4×10^3	0.0	0	1	0	0	0	2.0	1.06×10^{-3}	14.0	1.35×10^{-14}
1	0	4×10^3	0.0	0	1	0	0	0	5.0	7.88×10^{-5}	42.0	1.95
1	0	4×10^3	0.0	1	0	0	0	0	10.0	1.63×10^{-8}	5.0	0.0224
1	0	4×10^3	0.0	1	0	0	0	0	5.0	3.03×10^{-3}	3.0	0.0126
1	0	4×10^3	0.0	1	0	0	0	0	6.0	1.73×10^{-6}	11.0	4.70
1	0	4×10^3	0.0	1	0	0	0	0	3.0	5.51	15.0	2.82×10^{-2}
1	0	4×10^3	0.0	1	0	0	0	0	3.0	1.98×10^{-4}	12.0	1.01×10^{-4}
1	0	4×10^3	0.0	0	0	0	0	0	0	0	0	0
1	0	4×10^3	0.0	1	0	0	0	0	10.0	3.47×10^{-2}	4.0	6.47×10^{-2}
1	0	4×10^3	0.0	1	0	0	0	0	6.0	3.12×10^{-2}	7.0	4.95×10^{-3}

vi	ji	e	b	n12	n23	n31	nd	nc	time
1	0	4×10^3	0.0	7	2	0	0	0	10.96
1	0	4×10^3	0.25	7	1	2	0	0	10.89
1	0	4×10^3	0.5	8	0	2	0	0	12.15
1	0	4×10^3	0.75	8	0	1	0	1	11.75
1	0	4×10^3	1.0	10	0	0	0	0	12.56
1	0	4×10^3	1.25	6	1	0	0	3	12.08
1	0	4×10^3	1.5	9	0	1	0	0	13.07
1	0	4×10^3	1.75	7	2	1	0	0	11.91
1	0	4×10^3	2.0	9	1	0	0	0	12.16
1	0	4×10^3	2.25	10	0	0	0	0	12.56

Figure 5. Sample of the long output (**top**) and short output (**bottom**) of the PyQCAMS program. Both outputs contain initial vibrational number "vi", initial rotational number "ji", collision energy "e" in Kelvin, impact parameter "b" in Bohr, and product count. 'n12' is the number of trajectories that ended with a molecule formed by atoms 1 and 2 and similarly for "n23" and "n31". ''nd" is the number of dissociation collisions, and 'nc' is the number of trajectories ending in an intermediate complex. The long output prints a new line for each trajectory, containing the final vibrational state "v", its associated Gaussian weight "vw", the final rotational state "j", and its associated Gaussian weight "jw". The short output sums over all trajectories for a given (vi, ji, E_c, b), and reports the total calculation time in seconds.

2.2.4. Analysis Functions

PyQCAMS is equipped with an `analysis` module that contains functions to analyze the output file. These functions are `opacity` to compute the opacity function, `crossSection` to compute the cross section in units of cm^2, and `rate` to compute the reaction rate in units of cm^3/s. These functions take the output of PyQCAMS as an input. The user has a choice of using HB or GB and can also choose to calculate state-specific results or neglect them.

2.2.5. Parallel Implementation and Performance

The code is best used in a parallel implementation, which dramatically speeds up the time per trajectory as the number of CPUs is increased (Figure 6). The `qct.runN` method automatically runs the trajectory in parallel using multiprocessing with all CPUS. The

number of CPUs is controllable and reverts to serial computation when the number of CPUs is set to 1.

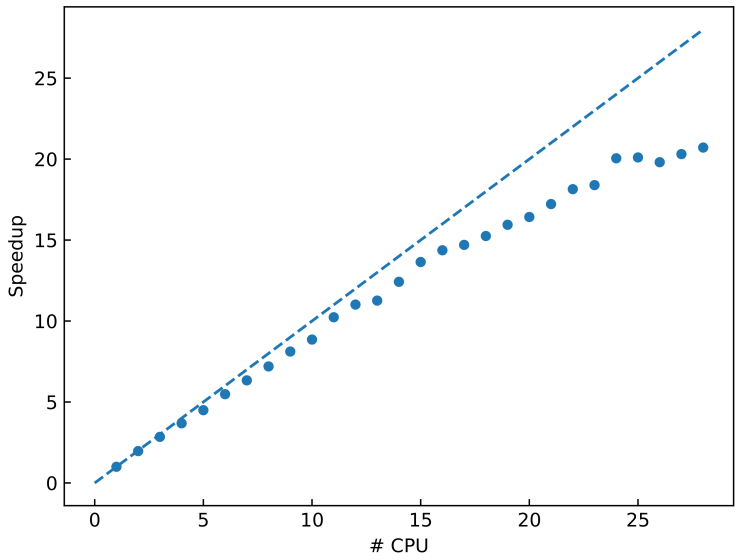

Figure 6. Calculation speedup as a function of the number of CPUs used in the parallel calculation, where the dashed line represents perfect linear speedup. These trajectories were run at a collision energy of 40,000 K for H_2 + Ca reactions, with a relative tolerance of 10^{-12} and absolute tolerance of 10^{-11}. These calculations were performed on a 28-core node on the SeaWulf cluster located at the Institute for Advanced Computational Science (IACS) at Stony Brook University.

Figure 6 shows the speedup of the program as the number of CPUs varies, averaged over 1000 trajectories, calculated as $S(N) = t(1)/t(N)$ for N CPUs. It is clear that parallel processing yields a speedup of up to 20 times, thus a significant decrease in the total calculation time.

The majority of the calculation time is spent during the solution of the Hamilton's equations of motion. Scipy's `solve_ivp` API allows the user to control the absolute and relative tolerances to control local error estimates. These can be controlled in the input dictionary and will have a large influence on the energy and momentum conservation and the time per trajectory. It is highly recommended to study the effect that these tolerances have on a system before running large calculations by tracking those attributes of the `Trajectory` object.

2.2.6. Visualization

The file `plotters.py` contains several methods for visualizing the results of a trajectory. Figure 2 was obtained using the `traj_plt` function, which requires a trajectory object as input. There is also a 3D plot generator `traj_3d`, which provides a trace of the event. The usage of these plotters is shown in the example Jupyter notebook.

3. Results

3.1. $RbBa^+$ + Rb

As a benchmark, we study a system of cold molecular ions embedded in an ultracold gas, namely the reaction $RbBa^+$ + Ba as studied in [12]. We use the same pairwise interaction potential between Rb and Ba^+, where it is assumed that the charge is localized in the Ba atom, with the Lennard–Jones potential: $V(r) = C_8/r^8 - C_4/r^4$. For Rb-Rb, we use the triplet ground state potential parameters obtained from [31] in the Lennard–Jones potential: $V(r) = C_{12}/r^{12} - C_6/r^6$, whereas [12] uses the full potential provided in [31]. We calculate the quenching cross section ($RbBa^+(v_i)$ + Rb $\rightarrow$ $RbBa^+(v \neq v_i)$ + Rb) for states initialized at various high-lying vibrational states (with j = 0) near the dissociation of $RbBa^+$. In Figure 7, we compare the quenching cross sections calculated by PyQCAMS to those presented in [12], where the cross sections were also found through the QCT method.

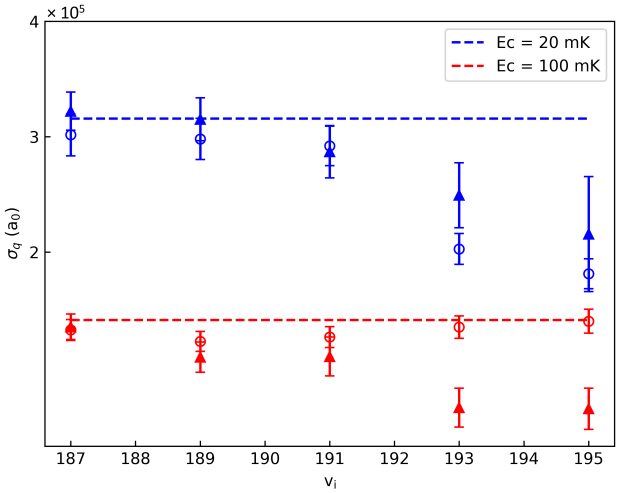

Figure 7. Quenching cross sections for various initial vibrational states, v_i, at collision energies E_c = 20 mK (blue) and 100 mK (red). The dashed lines represent the classical Langevin cross section for the collision energy. The hollow dotted points represent results from PyQCAMS, while the triangle points are the previously reported results [12].

The calculated quenching cross sections are plotted alongside the Langevin cross section $\sigma_L = \pi(4C_4)^{1/2}E_c^{-1/2}$. We see good agreement with both the Langevin prediction and the previous results at the lower vibrational states. For the higher vibrational states, we notice that, at $E_c = 100$ mK, the PyQCAMS results lie closer to the Langevin prediction than previously reported. These elongated states seem to pose a challenge for convergence, as shown in the 20 mK data for both sets of results. However, PyQCAMS seems to have performed better at 100 mK than previous calculations regarding the Langevin prediction, demonstrating the strength of the program.

3.2. H_2 + Ca

As an example, we demonstrate the calculation of the CaH formation rate as a result of the reaction H_2 + Ca → CaH + H. The first part of the code contains the inputs as described in Section 2.2.1. The potential ranges and parameters for H_2 and CaH were obtained from [32,33] and [34,35], respectively. Here, we chose the Morse potential to describe the interaction of both H_2 and CaH. We ran 10^4 trajectories in parallel, looped over 20 impact parameters. The script for this calculation is shown in Listing 1.

We repeated the code in Listing 1 over a range of collision energies E_c and three different initial vibrational levels of H_2. The rates for CaH formation are shown in Figure 8, where it is noticed that higher collision energies give rise to larger reaction rates, as expected in endothermic reactions. Additionally, as the initial vibrational level of H_2 is increased, the rates of reaction increase for a given collision energy. However, at very high collision energies such a trend changes due to the dominance of molecular dissociation processes. A more detailed study of this reaction will be published elsewhere.

We can also calculate the distribution of states of CaH and H_2, using the final state vector $\vec{s}$. Figure 9 shows these distributions at four different collision energies. We can see that the higher vibrational states fill up for both CaH and H_2 as the collision energy is increased, as is typical in endothermic reactions. All details of these calculations can be found in a Jupyter notebook example file.

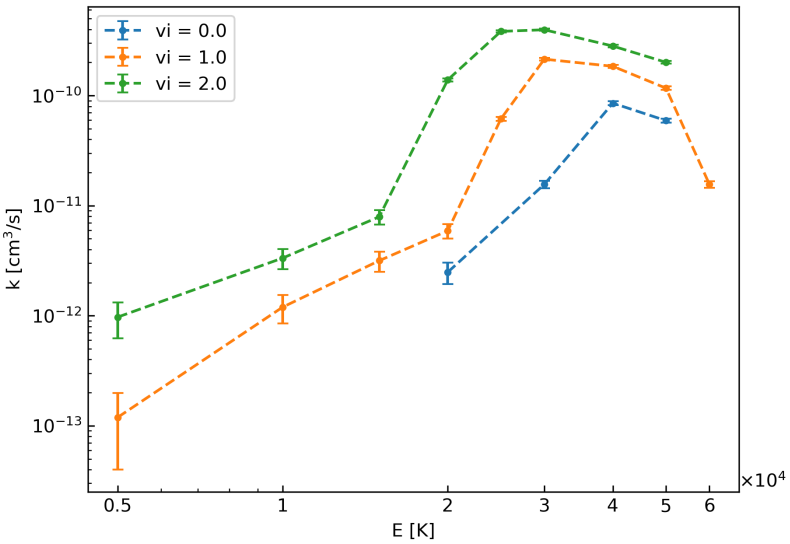

Figure 8. Rate of reaction of H_2 + Ca → CaH + H. Different collision energies were considered, and 10^4 trajectories were run at each energy. We used 20 evenly spaced impact parameters between 0 and 5 a_0 for each collision energy. Here, the H_2 molecule was initiated at $v = 0$, 1, 2, $j = 0$, and each pairwise interaction was defined by a Morse potential. From here, we see that CaH was most likely formed at a collision energy E_c = 40,000 K for $v = 0$, E_c = 30,000 K for $v = 1$, and E_c = 25,000 K for $v = 2$.

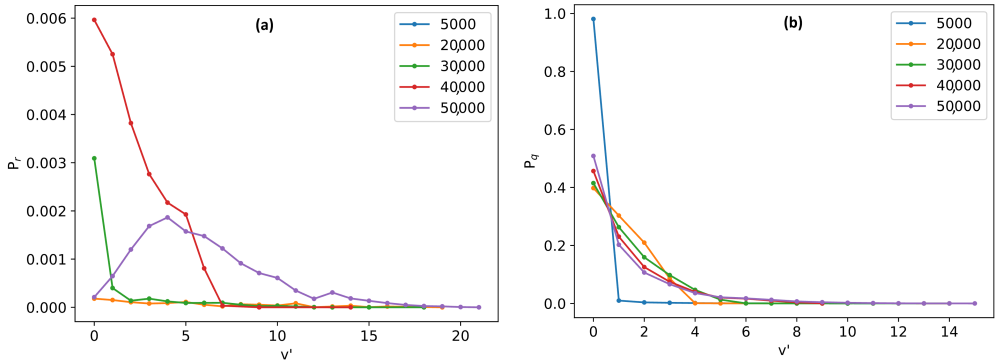

Figure 9. Probability distribution of the final vibrational states of CaH (**a**) and H_2 (**b**) as a result of the reaction H_2 + Ca, calculated by PyQCAMS. H_2 was initiated at $v = 0$, $j = 0$, with each pairwise interaction defined by a Morse potential. Each color represents a different collision energy E_c.

4. Discussion

We have presented a Python quasi-classical atom–molecule scattering program, PyQCAMS. The PyQCAMS program aims to provide an easy-to-use platform for calculating quasi-classical trajectories for atom–diatomic molecule systems, including the three most relevant potentials for diatomic molecules: Morse, Buckingham, and the generalized Lennard–Jones, as well as any user-defined analytic potentials supplied by the user. We discussed the underlying theory behind the program and the methods of the program as they pertain to the theory. As output, the user obtains the reaction probability per energy and impact parameter. Then, with this information, it is possible to calculate the cross section and energy-dependent rate constant using the provided analysis functions. Its object-oriented approach allows the user to study different properties of a given trajectory. The plotter tools make it easy to visualize the results of a trajectory, making it ideal for a new researcher studying trajectories or for presenting the topic in a classroom setting. Finally, we would like to emphasize that there are already codes

available for the calculation of reaction dynamics based in QCT methods, e.g., VENUS [36], Newton-X [37], or SHARC [38]. However, these programs are dedicated to more complex scenarios and are less versatile than the present code for atom–diatom collisions. Our code is fully written in Python, making it fully transparent and user-friendly. Furthermore, we offer direct plotting tools to rapidly diagnose any complication during the computation of a new system.

Author Contributions: Conceptualization, J.P.-R.; methodology, J.P.-R.; software, R.K. and J.P.-R.; formal analysis, R.K. and J.P.-R.; investigation, R.K. and J.P.-R.; data curation, R.K.; writing—original draft preparation, R.K. and J.P.-R.; writing—review and editing, R.K. and J.P.-R.; visualization, R.K.; supervision, J.P.-R.; project administration, J.P.-R.; funding acquisition, J.P.-R. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the United States Air Force Office of Scientific Research (grant number FA9550-23-1-0202).

Data Availability Statement: The source code and example Jupyter notebook can be found at https://github.com/Rkoost/PyQCAMS, accessed on 8 May 2024. A sample dataset to reproduce the figures in the example notebook can be found at https://figshare.com/s/8b923dab304005ae7a5c, accessed on 8 May 2024.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

PyQCAMS	Python Quasi-Classical Atom–Molecule Scattering
QCT	Quasi-classical trajectory
DVR	Discrete variable representation
HB	Histogram binning
GB	Gaussian binning
API	Application programming interface

References

1. Karplus, M.; Porter, R.N.; Sharma, R.D. Exchange Reactions with Activation Energy. I. Simple Barrier Potential for (H, H_2). *J. Chem. Phys.* **1965**, *43*, 3259–3287. [CrossRef]
2. Truhlar, D.G.; Muckerman, J.T. Reactive scattering Cross sections III: Quasiclassical and semiclassical methods . In *Atom-Molecule Collision Theory: A Guide for the Experimentalist*; Plenum Press: New York, NY, USA, 1979; pp. 505–561.
3. Ríos, J. Cold Chemical Reactions Between Molecular Ions and Neutral Atoms. In *An Introduction to Cold and Ultracold Chemistry: Atoms, Molecules, Ions and Rydbergs*; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 215–234.
4. Aoiz, F.J.; Herrero, V.J.; Sáez Rábanos, V. Quasiclassical state to state reaction cross sections for D+H_2(v = 0, j = 0)→HD(v′,j′) + H. Formation and characteristics of short-lived collision complexes. *J. Chem. Phys.* **1992**, *97*, 7423–7436. [CrossRef]
5. Aoiz, F.J.; Bañares, L.; Herrero, V.J. Recent results from quasiclassical trajectory computations of elementary chemical reactions. *J. Chem. Soc. Faraday Trans.* **1998**, *94*, 2483–2500. [CrossRef]
6. De Oliveira-Filho, A.G.S.; Ornellas, F.R.; Bowman, J.M. Quasiclassical Trajectory Calculations of the Rate Constant of the OH + HBr →Br + H_2O Reaction Using a Full-Dimensional Ab Initio Potential Energy Surface Over the Temperature Range 5 to 500 K. *J. Phys. Chem. Lett.* **2014**, *5*, 706–712. [CrossRef] [PubMed]
7. Nagy, T.; Vikár, A.; Lendvay, G. A general formulation of the quasiclassical trajectory method for reduced-dimensionality reaction dynamics calculations. *Phys. Chem. Chem. Phys.* **2018**, *20*, 13224–13240. [CrossRef]
8. Patra, S.; San Vicente Veliz, J.C.; Koner, D.; Bieske, E.J.; Meuwly, M. Photodissociation dynamics of N_3+. *J. Chem. Phys.* **2022**, *156*, 124307. [CrossRef]
9. Töpfer, K.; Upadhyay, M.; Meuwly, M. Quantitative molecular simulations. *Phys. Chem. Chem. Phys.* **2022**, *24*, 12767–12786. [CrossRef] [PubMed]
10. Goswami, S.; San Vicente Veliz, J.C.; Upadhyay, M.; Bemish, R.J.; Meuwly, M. Quantum and quasi-classical dynamics of the C(3P) + O_2(3σ-g) → CO(1σ+) + O(1D) reaction on its electronic ground state. *Phys. Chem. Chem. Phys.* **2022**, *24*, 23309–23322. [CrossRef]
11. Hirzler, H.; Pérez-Ríos, J. Rydberg atom-ion collisions in cold environments. *Phys. Rev. A* **2021**, *103*, 043323. [CrossRef]

12. Pérez-Ríos, J. Vibrational quenching and reactive processes of weakly bound molecular ions colliding with atoms at cold temperatures. *Phys. Rev. A* **2019**, *99*, 022707. [CrossRef]
13. Hirzler, H.; Trimby, E.; Lous, R.S.; Groenenboom, G.C.; Gerritsma, R.; Pérez-Ríos, J. Controlling the nature of a charged impurity in a bath of Feshbach dimers. *Phys. Rev. Res.* **2020**, *2*, 033232. [CrossRef]
14. Mota, V.C.; Caridade, P.J.S.B.; Varandas, A.J.C.; Galvão, B.R.L. Quasiclassical Trajectory Study of the Si + SH Reaction on an Accurate Double Many-Body Expansion Potential Energy Surface. *J. Phys. Chem. A* **2022**, *126*, 3555–3568. [CrossRef] [PubMed]
15. Frisch, M.J.; Trucks, G.W.; Schlegel, H.B.; Scuseria, G.E.; Robb, M.A.; Cheeseman, J.R.; Scalmani, G.; Barone, V.; Petersson, G.A.; Nakatsuji, H.; et al. *Gaussian 16*; Revision C.01; Gaussian Inc.: Wallingford, CT, USA, 2016.
16. Hase, W.L.; Duchovic, R.J.; Hu, X.; Komornicki, A.; Lim, K.F.; Lu, D.h.; Peslherbe, G.H.; Swamy, K.N.; Linde, S.; Varandas, A.; et al. A general chemical dynamics computer program. *Quantum Chem. Program Exch. Bull.* **1996**, *16*, 671.
17. Harris, C.R.; Millman, K.J.; van der Walt, S.J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N.J.; et al. Array programming with NumPy. *Nature* **2020**, *585*, 357–362. [CrossRef] [PubMed]
18. Virtanen, P.; Gommers, R.; Oliphant, T.E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nat. Methods* **2020**, *17*, 261–272. [CrossRef] [PubMed]
19. Hunter, J.D. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.* **2007**, *9*, 90–95. [CrossRef]
20. The Pandas Development Team. pandas-dev/pandas: Pandas, 2020. Available online: https://doi.org/10.5281/zenodo.3509134 (accessed on 21 January 2024).
21. McKerns, M.M.; Strand, L.; Sullivan, T.; Fang, A.; Aivazis, M.A.G. Building a Framework for Predictive Science. *arXiv* **2012**, arXiv:1202.1056.
22. McKerns, M.; Aivazis, M. Pathos: A Framework for Heterogeneous Computing, 2010. Available online: http://trac.mystic.cacr.caltech.edu/project/pathos (accessed on 21 January 2024).
23. Colbert, D.T.; Miller, W.H. A novel discrete variable representation for quantum mechanical reactive scattering via the S-matrix Kohn method. *J. Chem. Phys.* **1992**, *96*, 1982–1991. [CrossRef]
24. Bonnet, L.; Rayez, J. Quasiclassical trajectory method for molecular scattering processes: Necessity of a weighted binning approach. *Chem. Phys. Lett.* **1997**, *277*, 183–190. [CrossRef]
25. Bonnet, L.; Espinosa-García, J. The method of Gaussian weighted trajectories. V. On the 1GB procedure for polyatomic processes. *J. Chem. Phys.* **2010**, *133*, 164108. [CrossRef]
26. Bonnet, L. The method of Gaussian weighted trajectories. III. An adiabaticity correction proposal. *J. Chem. Phys.* **2008**, *128*, 044109. [CrossRef] [PubMed]
27. Aguado, A.; Paniagua, M. A new functional form to obtain analytical potentials of triatomic molecules. *J. Chem. Phys.* **1992**, *96*, 1265–1275. [CrossRef]
28. Aguado, A.; Tablero, C.; Paniagua, M. Global fit of ab initio potential energy surfaces I. Triatomic systems. *Comput. Phys. Commun.* **1998**, *108*, 259–266. [CrossRef]
29. Axilrod, B.M.; Teller, E. Interaction of the van der Waals Type Between Three Atoms. *J. Chem. Phys.* **2004**, *11*, 299–300. [CrossRef]
30. Dormand, J.; Prince, P. A family of embedded Runge-Kutta formulae. *J. Comput. Appl. Math.* **1980**, *6*, 19–26. [CrossRef]
31. Strauss, C.; Takekoshi, T.; Lang, F.; Winkler, K.; Grimm, R.; Hecker Denschlag, J.; Tiemann, E. Hyperfine, rotational, and vibrational structure of the $a^3\Sigma_u^+$ state of $^{87}Rb_2$. *Phys. Rev. A* **2010**, *82*, 052514. [CrossRef]
32. Yan, Z.C.; Babb, J.F.; Dalgarno, A.; Drake, G.W.F. Variational calculations of dispersion coefficients for interactions among H, He, and Li atoms. *Phys. Rev. A* **1996**, *54*, 2824–2833. [CrossRef] [PubMed]
33. Liu, J.; Salumbides, E.J.; Hollenstein, U.; Koelemeij, J.C.J.; Eikema, K.S.E.; Ubachs, W.; Merkt, F. Determination of the ionization and dissociation energies of the hydrogen molecule. *J. Chem. Phys.* **2009**, *130*, 174306. [CrossRef]
34. Shayesteh, A.; Alavi, S.F.; Rahman, M.; Gharib-Nezhad, E. Ab initio transition dipole moments and potential energy curves for the low-lying electronic states of CaH. *Chem. Phys. Lett.* **2017**, *667*, 345–350. [CrossRef]
35. Mitroy, J.; Zhang, J.Y. Properties and long range interactions of the calcium atom. *J. Chem. Phys.* **2008**, *128*, 134305. [CrossRef]
36. Hase, W.L.; Duchovic, R.J.; Hu, X.Y.; Komornicki, A.; Lim, K.F.; Lu, D.H.; Peslherbe, G.H.; Swamy, K.N.; Linde, S.R.V.; Varandas, A.J.C.; et al. *VENUS96: A General Chemical Dynamics Computer Program*; Wayne State University: Detroit, MI, USA, 1996.
37. Barbatti, M.; Ruckenbauer, M.; Plasser, F.; Pittner, J.; Granucci, G.; Persico, M.; Lischka, H. Newton-X: A surface-hopping program for nonadiabatic molecular dynamics. *WIREs Comput. Mol. Sci.* **2014**, *4*, 26–33. [CrossRef]
38. Mai, S.; Avagliano, D.; Heindl, M.; Marquetand, P.; Menger, M.F.S.J.; Oppel, M.; Plasser, F.; Polonius, S.; Ruckenbauer, M.; Shu, Y.; et al. SHARC3.0: Surface Hopping Including Arbitrary Couplings—Program Package for Non-Adiabatic Dynamics. 2023. Available online: https://sharc-md.org/ (accessed on 9 May 2024).