Quantum Mechanical Effusion Rates out of a 2D Particle in a Box

Mohan J. Shankar

Department of Chemistry
University of Virginia
2024

Quantum Mechanical Effusion Rates out of a 2D Particle in a Box

Mohan J. Shankar

A Thesis Presented to the Faculty of the
University of Virginia
for the

Distinguished Majors Program

Research under the supervision of:
Dr. Sergei Egorov

Department of Chemistry
University of Virginia
April, 2024

Contents

Abstract

In academia and industry, knowledge about rates is indispensable for scientists, providing insight on pathway favorability and regioselectivity among other things. Transition-State Theory (TST) is the standard framework to calculate these rates, but the theory provides erroneous results at low temperatures and low masses as quantum effects such as tunneling and zero-point energy have a non-negligible contribution. In this work, we modify the well-known two-dimensional particle in a box quantum system to compute effusion rates before comparing them to classical rates. This study shows that at low temperatures and in small systems, quantum mechanical effusion rates k_q greatly deviate from the classical values k_c where $k_q << k_c$, opposing intuition in the range of temperatures and gap sizes explored. Nonetheless, quantum rates approached classical results in accordance with the correspondence principle, corroborating the capabilities of this approach and highlighting the potential of future results.

List of Figures

Chapter 1
Introduction

For academic and industrial pursuits, the relative and absolute rates of reactions are important pieces of information to hold, often dictating pathways which can produce better yields and control over attributes such as regioselectivity [1]. Consequently, the problem of finding reaction rates is fundamental to our understanding of chemistry, resulting in the development of Transition-State Theory (TST) starting in the 1930's [2].

Within the framework of TST, reaction rates are found using classical mechanics operating under the assumption that a well-defined dividing surface separating reactants and products known as the "transition-state" exists. An example is shown below in Figure 1.1.

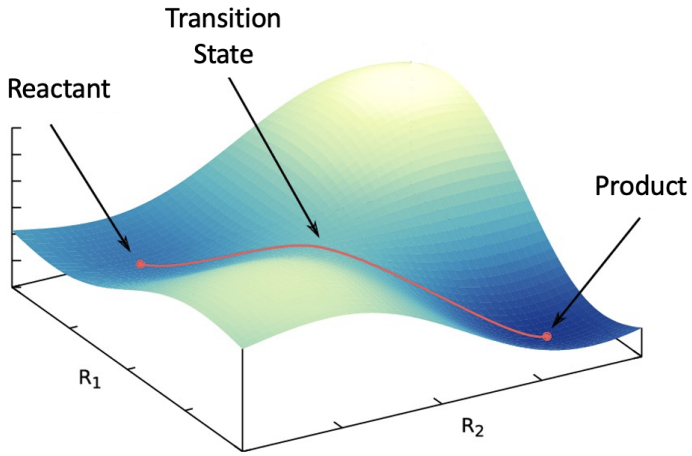

Figure 1.1: *Example of a transition state for an arbitrary, elementary reaction traveling along a potential energy surface.*

Furthermore, processes in TST are assumed to occur adiabatically, start in thermal equilibrium so they are amenable to treatment using the Boltzmann distribution, and have no recrossing (reactants to products is not reversed).

Classical TST has been very effective at calculating reaction rates for systems involving heavy atoms at high temperatures, but it often fails for the quantum regime where phenomena such as tunneling and zero-point energy occur [3, 4]. As such, there is a need to develop

a Quantum Transition-State Theory (QTST) dependent on quantum flux through a dividing surface in the absence of recrossing.

In this thesis, we study rate constants for quantum mechanical effusion of a particle out of a two-dimensional particle in a box with a hope of establishing of a baseline for QTST results. In Chapter 2, we discuss the theoretical minimum to understand the problem including the fundamentals of quantum mechanics, a brief discussion of classical TST, QTST, classical effusion, and Miller's correlation functions from which we obtain the quantum rates of effusion.

Chapter 3 elucidates the methodology through which the problem is solved, particularly focusing on a finite difference method to numerically solve the partial differential equation in addition to evaluation of the correlation functions. Chapter 4 is dedicated to the results and their discussion, focusing on the behavior of the correlation functions over a range of temperatures. Furthermore, the classical and quantum rate constants for a given wall thicknesses and gap sizes are shown. Finally, we present a summary of our findings and the most relevant conclusions of our work in Chapter 5.

Chapter 2
Theory

> God does not play dice with the universe.
>
> *A Letter to Max Born, December 1926*
> *from Albert Einstein*

As it turns out, quantum mechanics acts in a stark contrast to that of classical mechanics with the key difference being a probabilistic nature rather than a deterministic one. The theory works nonetheless, and a brief discussion of it begins below.

2.1 Introductory Quantum Mechanics

$\boldsymbol{F} = m\boldsymbol{a}$ stands in the pantheon of famous equations along with $e = mc^2$, but the former is not how Newton wrote it. Instead, force was expressed as the first derivative of momentum with respect to time shown as

$$\boldsymbol{F} = \frac{d\boldsymbol{p}}{dt} = \dot{\boldsymbol{p}} \tag{2.1}$$

or the second derivative of position with respect to time multiplied by mass,

$$\boldsymbol{F} = m\frac{d^2\boldsymbol{r}}{dt^2} = m\ddot{\boldsymbol{r}} \tag{2.2}$$

where the dot(s) denotes a derivative(s) with respect to time and bold letters vector quantities. The last thing to note is that $\boldsymbol{F} = -\nabla V(\boldsymbol{r})$ where ∇ denotes the gradient and $V(\boldsymbol{r})$ the potential energy of the system. Consequently, the position $\boldsymbol{r}(t)$ and velocity $\boldsymbol{v}(t)$ can be found by solving the differential equations with the necessary initial conditions of $\boldsymbol{r}(0)$ and $\boldsymbol{v}(0)$.

In a more sophisticated formalism of classical mechanics, the state of a system with n degrees of freedom is described by 3N coordinates $(\boldsymbol{q}_1, \boldsymbol{q}_2, ..., \boldsymbol{q}_{3N})$ and 3N momenta $(\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_{3N})$ resulting in a single point for 6N-dimensional phase space [5]. For this Hamiltonian formalism, the dynamics of the system are governed by the Hamiltonian,

$$\mathcal{H}(\boldsymbol{q}, \boldsymbol{p}) = \sum_{i=1}^{3N} \frac{\boldsymbol{p}_i^2}{2m} + V(\boldsymbol{q}) \tag{2.3}$$

which is the sum of all kinetic and potential energy in the system. The analogous differential equations governing the dynamics of the system are

$$\frac{\partial \mathcal{H}(\boldsymbol{q}, \boldsymbol{p})}{\partial \boldsymbol{p}_i} = \dot{\boldsymbol{q}}_i \tag{2.4}$$

and

$$\frac{\partial \mathcal{H}(\boldsymbol{q}, \boldsymbol{p})}{\partial \boldsymbol{q}_i} = -\dot{\boldsymbol{p}}_i \tag{2.5}$$

where $\boldsymbol{q}$ replaces $\boldsymbol{r}$ as generalized position and $\boldsymbol{p}$ generalized momentum.

Discussing the Hamiltonian formalism of classical mechanics makes the motivation of quantum a bit easier. Classical mechanics dictates that at any given time, the state of a particle can be specified by *x(t)* and *p(t)* as a point in phase space.

In quantum mechanics, the state of a particle is represented by a *ket vector* $|\psi\rangle$ in a Hilbert space. Of note is that the Hilbert space comes equipped with an inner product denoted as

$$\langle \phi | \psi \rangle = \int dx \, \phi(x)^* \psi(x) \tag{2.6}$$

after picking a particular basis, typically position or momentum. Note that the asterisk denotes the complex conjugate.

Here, the *bra vector* $\langle \phi |$ is an element of the ket vector's dual space while $\phi(x)$ and $\psi(x)$ are the respective *wavefunctions* of the bra and ket. The wavefunction is, perhaps, the most important mathematical object in quantum mechanics as its square $|\psi(x)|^2 = \psi^*(x)\psi(x)$ has a probabilistic interpretation. Consequently, the total probability of finding the particle must be unity,

$$\int_{\text{all space}} dx \, \psi^*(x)\psi(x) = 1 \, . \tag{2.7}$$

Continuing with outlining the parallels of classical and quantum mechanics, classical mechanics dictates that dynamical variables ω are a function of position in momentum such that $\omega = \omega(x, p)$. In quantum mechanics, x and p are represented by a Hermitian operator that acts on a ket via left multiplication:

$$\hat{\mathcal{A}} : |\psi\rangle \to |\psi'\rangle = \hat{\mathcal{A}} \, |\psi\rangle \ . \tag{2.8}$$

From here, one must note that in any measurement of the observable associated with the operator $\hat{\mathcal{A}}$, the only values that will ever be observed are the eigenvalues a_n which satisfy the eigenvalue equation:

$$\hat{\mathcal{A}} \, |\psi_n\rangle = a_n \, |\psi_n\rangle \ . \tag{2.9}$$

Lastly, the state variables change in time follow Eqn. 2.4 and 2.5 for classical mechanics. The quantum analogue sees the state vector, $|\psi(t)\rangle$ follow the *time-dependent* Schrödinger equation

$$i\hbar \frac{\partial}{\partial t} \, |\psi(x,t)\rangle = \hat{H} \, |\psi(x,t)\rangle \ . \tag{2.10}$$

Here, $\hat{H}$ is the quantum mechanical operator associated with energy aptly named the "Hamiltonian" and is analogous to the classical form, $\mathcal{H}$ as

$$\hat{H} = \hat{T} + \hat{V} \tag{2.11}$$

where $\hat{T}$ is the quantum mechanical operator for kinetic energy and $\hat{V}$ the operator for potential energy. From 2.10, the *time-dependent* Schrödinger equation

$$\hat{H}\,|\psi\rangle = E\,|\psi\rangle\ , \tag{2.12}$$

can be derived. Solving this equation will be one of the main focuses of this thesis.

2.2 TST and QTST

As mentioned in the introduction, TST follows the key assumption of reactions progressing under the assumption of a well-defined transition state existing. Furthermore, the process must be adiabatic, in thermal equilibrium, and progress without recrossing. The assumptions for a QTST are the same save for the addition of quantum phenomena. It is important to note the assumption of no recrossing is weak and is largely contingent on a good choice of transition state.

Nonetheless, from TST, the rate can be accurately approximated as the classical flux through the dividing surface. In particular, it was determined that TST captured the short-time ($\lim_{t\to 0_+} t$) limit of a classical flux-side time-correlation function which would be equal to the exact rate ($\lim_{t\to\infty} t$) in the absence of recrossing of the dividing surface [6, 7].

In accordance with linear operators corresponding to physical observables within quantum mechanics, Miller developed correlation functions which leverage the quantum mechanical flux operator and provide entirely accurate rates [8]. With this power comes the drawback of computational infeasibility for sufficiently complex systems, leading many to develop intermediate theories that balance computational cost and accuracy.

2.3 Effusion

In order to decide if a new theory is accurate, it is necessary to compare it to prior known results in order to corroborate claims. In the case of TST and consequently QTST, the known results come from the classical mechanical description of effusion, the phenomenon when a gas particle passes through a gap which is smaller than the mean free path.

The kinetic theory of gases stemming from classical mechanics and TST arrive at the same equation to calculate a rate constant

$$k = \sqrt{\frac{k_B T}{2\pi m}}\frac{A}{V}\,, \tag{2.13}$$

where k_B is the Boltzmann constant, T is the temperature, m the mass of the particle, A the area of the gap, and V the volume of the box. In the case of a two-dimensional system, A corresponds to the size of the gap opening and V the area of the box.

The assumptions of TST still hold in the case of deriving Eqn. 2.13 as the process is adiabatic, reliably described in thermal equilibrium allowing for the use of Boltzmann statistics, and there is no possibility of recrossing. As a result, comparison of quantum effusion rates to classical values offers a promising avenue for further development of a QTST.

2.4 Correlation Functions

The flux (number of particles per unit time) passing through some point x = s is defined in quantum mechanics using the flux operator,

$$\hat{F}(s) = \frac{i}{\hbar}\left[\hat{H}, \hat{h}\right] , \tag{2.14}$$

where

$$\hat{h} = h(x-s) = \begin{cases} 1, \text{ if } x > s \\ 0, \text{ if } x < s \end{cases}$$

is the quantum mechanical projection operator. For the sake of TST, we let s be the position of the dividing surface so that $\hat{h}$ yields 1 if the particle is on the product side and 0 otherwise. For a time-dependent wavefunction in the position basis, the flux operator is generally defined as

$$\langle\phi|\hat{F}|\psi\rangle = -\frac{i\hbar}{2m}\left\{\phi(s,t)^*\frac{\partial\psi(s,t)}{\partial s} - \frac{\partial\phi(s,t)}{\partial s}^*\psi(s,t)\right\} . \tag{2.15}$$

where the asterisk here denotes the complex conjugate. The important implication of these equations is that flux will be constant if $\psi(s,0)$ (a stationary state) is an eigenstate of the Hamiltonian [9].

Using correlation functions, the rates k of quantum mechanical reactions can be calculated as time goes to infinity. In particular, the rate can be expressed in terms of the flux-side C_{fs} and flux-flux C_{ff} correlation functions [8]:

$$k(T)\cdot Q_r(T) = \lim_{t\to\infty} C_{fs}(t) , \tag{2.16}$$

$$k(T)\cdot Q_r(T) = \int_0^\infty dt\; C_{ff}(t) . \tag{2.17}$$

$Q_r(T)$ is the partition function of the reactant area which can be calculated using the eigenvalues of 2D particle in a box

$$E_{n_x,n_y} = \frac{n_x^2\pi^2\hbar^2}{2mL_x^2} + \frac{n_y^2\pi^2\hbar^2}{2mL_y^2} , \tag{2.18}$$

where $\hbar$ is the reduced Planck constant, $L_{x,y}$ is the length of the box along the specified axis, and $n_{x,y} \in \mathbb{N}$ specifying the energetic state of the particle. With this found, the partition function can be calculated as

$$Q(T) = \sum_{n_x} exp\left[-\frac{\beta n_x^2 \pi^2 \hbar^2}{2mL_x^2}\right] \cdot \sum_{n_y} exp\left[-\frac{\beta n_y^2 \pi^2 \hbar^2}{2mL_y^2}\right], \tag{2.19}$$

where $\beta \equiv 1/k_B T$. Explicitly writing out the flux-side and flux-flux correlation function produces

$$C_{fs}(t) = tr\left[e^{-\beta\hat{H}/2}\hat{F}e^{-\beta\hat{H}/2}e^{+i\hat{H}t/\hbar}\hat{h}e^{-i\hat{H}t/\hbar}\right] \tag{2.20}$$

and

$$C_{ff}(t) = tr\left[e^{-\beta\hat{H}/2}\hat{F}e^{-\beta\hat{H}/2}e^{+i\hat{H}t/\hbar}\hat{F}e^{-i\hat{H}t/\hbar}\right]. \tag{2.21}$$

In order to explicitly evaluate these functions, we use a finite basis set of bound eigenstates of the Hamiltonian C_{ff} expressing it as

$$C_{ff} = \sum_{i,j} exp\left[-\frac{\beta(E_i + E_j}{2}\right] \cos\left(\frac{t(E_i - E_j)}{\hbar}\right) |\langle i|\hat{F}|j\rangle|^2 \tag{2.22}$$

where i, j are eigenstates of the Hamiltonian and E_i, E_j are their respective eigenvalues. In order to calculate the flux squared factors, we use

$$|\langle i|\hat{F}|j\rangle|^2 = |\langle i|s|\langle s|\hat{F}|s'\rangle\langle s'|j\rangle|^2 = \left|\int ds \int ds'\, \psi_i^*(s)\langle s|\hat{F}|s'\rangle\psi_j(s')\right|^2 \tag{2.23}$$

which can be expresed as

$$\int_{\partial R} ds \frac{\hbar^2}{4m^2}\left|\frac{\partial\psi_i(s)}{\partial n}\psi_j(s) - \frac{\partial\psi_j(s)}{\partial n}\psi_i(s)\right|^2 \tag{2.24}$$

where ∂R is the path along the dividing surface and $\frac{\partial}{\partial n}$ the derivative across the surface towards the product surface. In order to produce flux-side correlation functions to find the rate, we integrate Eqn. 2.22 to produce

$$C_{fs} = \sum_{i,j} exp\left[-\frac{\beta(E_i + E_j}{2}\right] \frac{\hbar \sin\left(\frac{t(E_i - E_j)}{\hbar}\right)}{(E_i - E_j)} |\langle i|\hat{F}|j\rangle|^2. \tag{2.25}$$

Comparing Eqn. 2.16 and Eqn. 2.25, the presence of the sinusoidal function results in the lack of a formal limit at infinite time. The issue is circumvented if the function is well-behaved as it can reach the "limit" at an intermediate time before the instability associated with discretizing the energy spectrum renders the expression inaccurate.

An example of a well-behaved C_{fs} and C_{ff} function are respectively shown Figure 2.1 and Figure 2.2 . In the case of Figure 2.1, a plateau is reached at t ∼2000 until t ∼6000 before falling into the unstable regime.

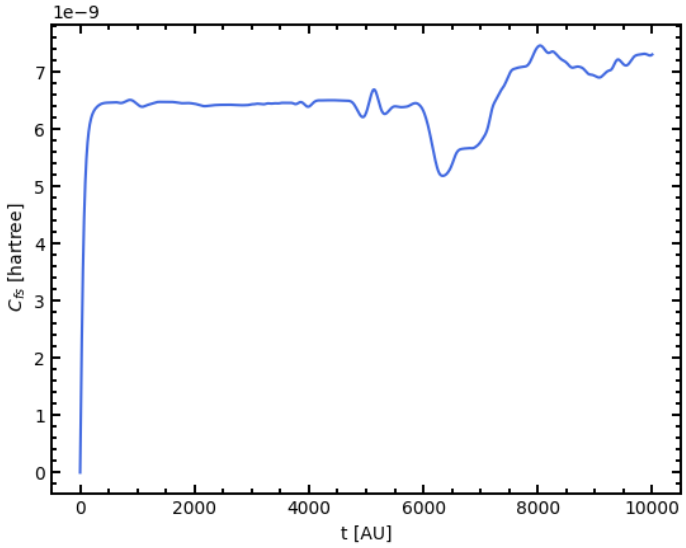

Figure 2.1: *An example of a well-behaved C_{fs} function at 1500K for a system with a 2pt gap and 2 pt wall thickness.*

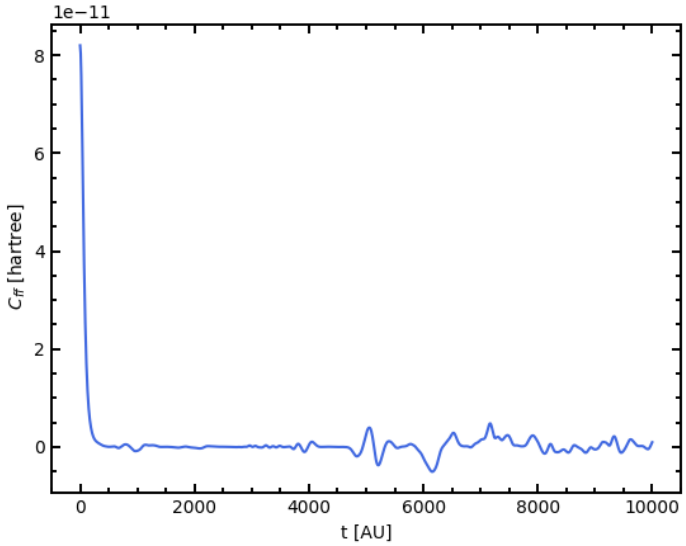

Figure 2.2: *An example of a well-behaved C_{ff} function at 1500K for a system with a 2pt gap and 2 pt wall thickness.*

Chapter 3
Methods

For this work, we employed a centered finite difference approximation to numerically describe the second derivative in the x and y direction. The weights were taken as the limit of the number of points tending to infinity, deemed appropriate as all points outside of the area considered were set to zero [10].

The modified particle in a box system is shown below in Figure 3.1. The box on the left is defined to be the "reactant" region, representing the initial position of the particle while the larger box on the right represents the space a particle escapes into. Consequently, the left box must be sufficiently small relative to the "product" box, thus their respective dimensions were set to (1 Å $\times$ 1 Å) and (1 Å $\times$ 8 Å) while the mass of the particle was defined as 1 a.m.u. (1.66×10^{-27} kg).

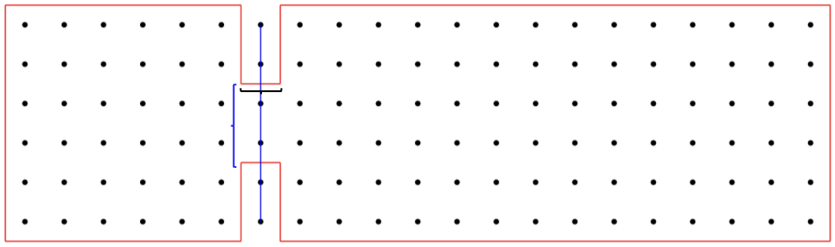

Figure 3.1: *Diagram of the system, not to scale. The black points represent how the system was spatially discretized. The red lines represent walls where the wavefunction and at and outside the walls are set to zero. The blue line represents the dividing surface separating the reactant region (left box) and empty space (right box). Finally, the blue curly brace denotes the gap size while the black curly brace marks wall thickness.*

As the dividing surface must be infinitesimally small, the thickness of the wall was varied over 1, 2, 3, 4, and 5-points in order to extrapolate the rate as width of the wall tends to zero. Gap size was similarly varied over the set of 2-, 4-, 6-, 8-, and 10 points.

3.1 The Code

All code for this project was written in Python using NumPy as the external library for calculations. Everything was done in atomic units to simplify calculations. After results

were found, they were converted back to SI units using the relevant conversion factors.

For the algorithms, we use 0_n to denote a vector of all zeros such that $0_n \in \mathbb{R}^n$. Similarly, $0_{n,m}$ is used to represent a matrix of all zeros such that $0_{n,m} \in \mathbb{R}^{n\times m}$. It is important to note that n, $m \in \mathbb{W}$ and n can equal m. This is done for algorithmic clarity and denoting the pre-allocation of arrays for increased code efficacy.

3.1.1 Creation of the Hamiltonian and Slit

Construction of the Hamiltonian matrix started by dividing the space into nx and ny points along the x- and y-axis respectively. The space between each point including the walls was defined such that $dn \equiv \frac{L_{\max - L_{\min}}}{n_n+1}$ where $L_{\max}$ is the largest value along one axis and $L_{\min}$ the minimum along the same axis. Lastly, n_n denotes the number of points along the an axis that is used to discretize the space. This procedure is shown in Algorithm 1 below.

Algorithm 1: Creation of the Hamiltonian Matrix for a 1D P.I.B. System

Procedure *1D_Hamiltonian(n, lmax, lmin, m)*:
- **Input:** Number of points: n; Max. length: *lmax*; Min. length: *lmin*; Mass: m
- **Output:** Hamiltonian matrix H for the associated 1D system.
- **begin**
  - $dn \leftarrow \frac{lmax-lmin}{n+1}$;
  - $dn2 \leftarrow dn^2$;
  - $H \leftarrow 0_{n,n}$; `/* Initialize Hamiltonian, H for system ∈ ℝ */`
  - **for** $i \leq n$ **do**
    - **for** $j \leq n$ **do**
      - **if** $i = j$ **then**
        - $H_{i,j} = \frac{\pi^2}{3} \cdot \frac{\hbar^2}{2\cdot m\cdot dn}$
      - **else**
        - $H_{i,j} = \frac{2}{(i-j)^2} * (-1)^{(i-j+1)}$
      - **end**
    - **end**
  - **end**
- **end**

end

A key part of making the Hamiltonian for the two-dimensional system is the use of the Kronecker sum, defined as

$$A \oplus B \equiv A \otimes I_B + I_A \otimes B\,. \tag{3.1}$$

Here, $A \in \mathbb{C}^{a\times a}$, $B \in \mathbb{C}^{b\times b}$, $\otimes$ denotes a Kronecker product, and $a, b \in \mathbb{R}$. For the following code, however, $A, B \in \mathbb{R}^{a\times a}, \mathbb{R}^{b\times b}$ as the matrices will not contain $i = \sqrt{-1}$. Applying the Kronecker Sum to the one-dimensional Hamiltonian corresponding to the x and y-axis produces a Hamiltonian for a two-dimensional particle in a box system, shown below in Algorithm 2.

Algorithm 2: Creation of the Hamiltonian for a 2D P.I.B. System

Procedure *2D_Hamiltonian* (h_x, h_y)**:**
  Input: 1D x-Hamiltonian: h_x; 1D y-Hamiltonian: h_y
  Output: Hamiltonian matrix H for the associated 2D P.I.B. system.
  begin
    $H = h_x \oplus h_y$; `/* H for P.I.B.` $\in \mathbb{R}^2$ `before adding slit */`
  end
end

The Kronecker sum computationally produces the same result and qualitatively makes sense to apply as it's used to combine two vector spaces and the respective "cross terms" as well.

For the final portion of creating the Hamiltonian, the slit is introduced into our system. Particularly, it is placed symmetrically such that the center of the slit lies at the center of the y-axis. The eigenvalues and eigenfunctions are then found where the latter are normalized in order to fulfill their probabilistic interpretation. The logic for its creation is listed below in Algorithm 3.

Algorithm 3: Creation of the Hamiltonian for 2D Effusion System

Procedure *effusion_slit* (h_x, h_y)**:**
- **Input:** x-points: nx; y-points: ny; gap size: s; wall thickness: w; 2D Hamiltonian: H
- **Output:** Hamiltonian matrix H for the associated 2D Effusion system.
- **begin**
  - i = -1;
  - **if** $k \leq nx$ **then**
    - **if** $j \leq ny$ **then**
      - i = i + 1;
      - $x = dx \cdot k + xmin$;
      - $y = dy \cdot j + ymin$;
      - **if** $\frac{1.0}{a_0} < y < \frac{1.0}{a_0} + w \cdot dy$ **then**
        - **if** $x < \left(\frac{xmax}{2} - \frac{s \cdot dx}{2}\right)$ *or* $x > \left(\frac{xmax}{2} + \frac{s \cdot dx}{2}\right)$ **then**
          - $H_i = 0_{nx,ny}$;
          - $H_{j,i} = 0_{nx,ny}$;
          - $H_{i,i} = -1$
        - **end**
      - **end**
    - **end**
  - **end**
- **end**
- Diagonalize H ; `/* Recall the eigenvecs for H =` ψ_i `*/`
- **forall** i **do**
  - z = $|\psi_i|^2$;
  - norm = $\sqrt{\text{sum } z}$;
  - $\psi_i = \frac{\psi_i}{\text{norm}}$
- **end**

end

An example of a wavefunction produced through this scheme is shown below in Figure 3.2.

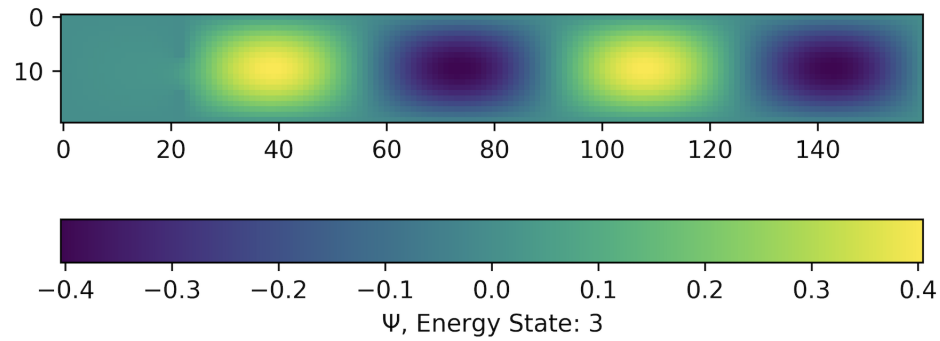

Figure 3.2: *An example of a normalized wavefunction in the 3rd excited state found at a 2 point wall thickness and 6 point gap size where nx = 20, ny = 160.*

3.1.2 Flux and Correlation Functions

The derivative over the dividing surface is calculated using centered finite difference approximations of the fourth order. The process can be found below in Algorithm 4.

Algorithm 4: Calculation of Flux.

Procedure *flux_calculator (x, y, nx, ny, d, s)*:
  Input: # of x-points: nx; # of y-points: ny; wall thickness: d; gap size: s
  Output: Flux Squared Matrix.
  begin
    $cd \leftarrow 0_d$;
    $F \leftarrow 0_{i,j}$;
    $fx2 \leftarrow F^2$;
  end
  for $k \leq d$ **do**
    zz = 1.0
    for $j \leq d$ **do**
      if $j \neq k$ **then**
        $zz = \frac{(j+1)^2}{(j+1)^2-(k+1)^2}$
      else
        Continue
      end
    end
    $cd_k = \frac{zz}{2\cdot(k+1)}$
  end
end
forall i **do**
  forall *Points on Div. Surf. ;* `/* First deriv. along div. surf. */` **do**
    $\frac{\partial\psi_i(x)}{\partial y} = \frac{\psi_{i,(x,y-2)} - 8\,\psi_{i,(x,y-1)} + 8\,\psi_{i,(x,y+1)} - \psi_{i,(x,y+2)}}{12dy}$
  end
  forall i, j **do**
    $F_{i,j} = \sum_{x\in \text{ div. surf}} -\frac{\hbar}{2m}\left\{\frac{\partial\psi_i(x)}{\partial x}\psi_j(x) - \frac{\partial\psi_j(x)}{\partial x}\psi_i(x)\right\}$
  end
  $fx2 = F_{i,j}^2$
end

Finally, the correlation functions were calculated as described in Eqn. 2.22 and Eqn. 2.25. An algorithm for this procedure can be found below in 5.

Algorithm 5: Calculation of Correlation Functions.

Procedure *corr_calculator(T, t, fx2)*:
  Input: Temperature range: T; time array: t; Flux Squared Matrix: *fx2*
  Output: $C_{fs}(t)$ and $C_{ff}(t)$ functions.
  forall *t that is considered* **do**
    $Cfs \leftarrow 0_t$; `/* Flux-Side Corr */`
    $C_{ff} \leftarrow 0_t$
  end
  forall *i, j* **do**
    $s_{ff} = exp\left[-\frac{\beta(E_i+E_j}{2}\right]\cos\left(\frac{t(E_i-E_j)}{\hbar}\right)\cdot fx2$
    if $i = j$ **then**
      $s_{f,s} = exp\left[-\frac{\beta(E_i+E_j}{2}\right]\cdot t\cdot fx2$
    else
      $s_{f,s}exp\left[-\frac{\beta(E_i+E_j}{2}\right]\frac{\hbar\sin\left(\frac{t(E_i-E_j)}{\hbar}\right)}{(E_i-E_j)}\cdot fx2$
    end
  end
  $C_{ff} = C_{ff} + s_{ff}$;
  $C_{fs} = C_{fs} + s_{fs}$
end

3.2 Finding Rate Constants

Finding the constants is contingent on the flux-side correlation function C_{fs} being well-behaved, resulting in a plateau existing at an intermediate point in time. In order to quantitatively approach this, we implement numerical differentiation at each point to find where the derivative is ~ 0, creating a step function to denote where this plateaus are located. A very simple algorithm for the procedure can be found below in 6.

Algorithm 6: Calculation of Rates.

Procedure *rate_calculator(Cfs, ms)*:
  Input: Correlation Function: C_{fs}; max_slope: ms
  Output: The length of each plateau in the step function.
  $dc \leftarrow 0_t$; /* Array to hold num. deriv. */
  $sf \leftarrow 0_t$; /* Step Func. */
  forall $i \in C_{fs}$ **do**
    $dc_i = \frac{dC_{fs}}{dt}$
  end
  for $i \in dc$ **do**
    if $i \sim 0$ **then**
      $sf_i = 1$
    else
      $sf_i = 0$
    end
  end
end
forall *consecutive values of 1 in sf* **do**
  Return start and stop point of each string of 1's
end

Once the ranges with 1 of the step function are found, the middle of the largest plateau is taken and the value of C_{fs} isolated. An example of the step function relative to C_{fs} at 1000 K is shown below in Figure 3.3.

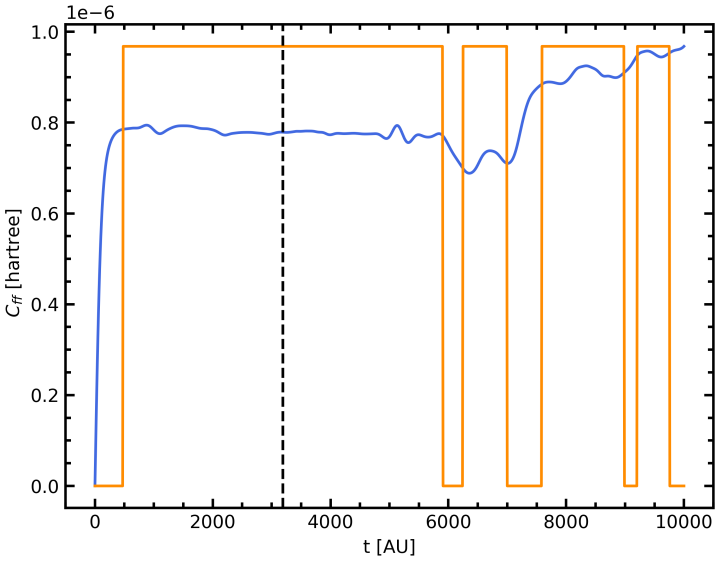

Figure 3.3: *C_{fs} function at 1000K for a system with a 6pt gap and 2 pt wall thickness. The blue plot denotes the flux-side correlation function while the orange plot is a step function whose value is "1" if the numerical derivative is $\sim$ 0 and 0 otherwise.*

The black dashed line denotes the center of the widest plateau where its intersection with C_{fs} is the y-value we extract to calculate the quantum rate constants, k_q. Finally, these results as wall thickness tended to zero were extracted at a fixed temperature and gap size using a quadratic fit. An example of this can be found below in Figure 3.4.

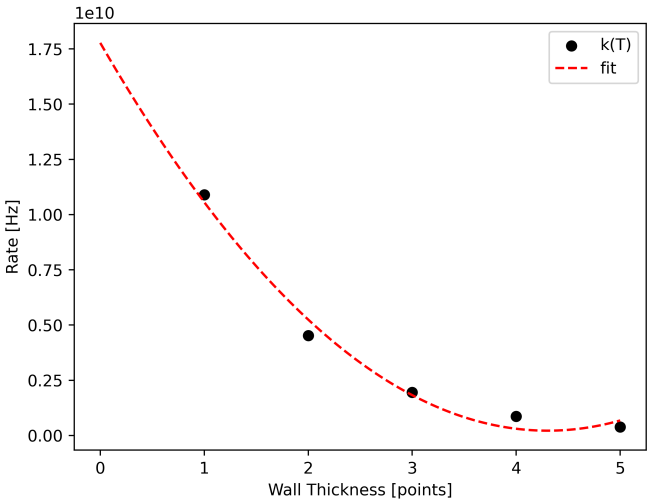

Figure 3.4: *Example of fitting a quadratic function to rate constants at 1250 K with a 6 point gap.*

Chapter 4
Results and Discussion

Rate constants were calculated for the system as described at the beginning of the previous chapter. The key points are the reactant box is 1 Å × 1 Å while "open space" 1 Å × 8 Å, represented by 60 × 60 and 60 × 480 points respectively. Consequently, an n point quantity has a conversion factor where 1 point = $0.01\bar{6}$ Å thus n points = $0.01\bar{6}\,n$ Å. These lengths, nx, and ny were found to be adequate for convergence of rates by a collaborator, eliminating the need to increase computational load by through an increase in number of points.

The mass of the particle was 1 a.m.u. (1.66×10^{-27} kg), and the wall thickness and gap size were varied between 1, 2, 3, 4, 5, and 2, 4, 6, 8, and 10 points respectively. Plots of the log of the quantum and classical rate constants are shown in Figure 4.1 through Figure 4.5. It's important to note that Eqn. 2.13 has no dependence on wall thickness as the analytic solutions expresses results where wall thickness is zero. Consequently, the classical rates in the aforementioned figures will be the same while the quantum rates differ.

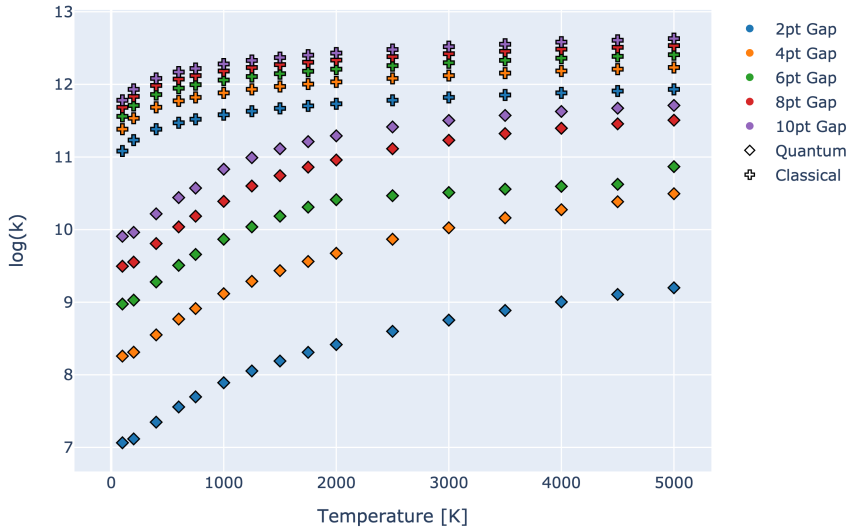

Figure 4.1: *Log of quantum vs. classical rate constants at a 1 point wall thickness. The varied colors denote denote different gap sizes while the shapes signify the formalism. The units of each rate constant were in Hz prior to taking the log.*

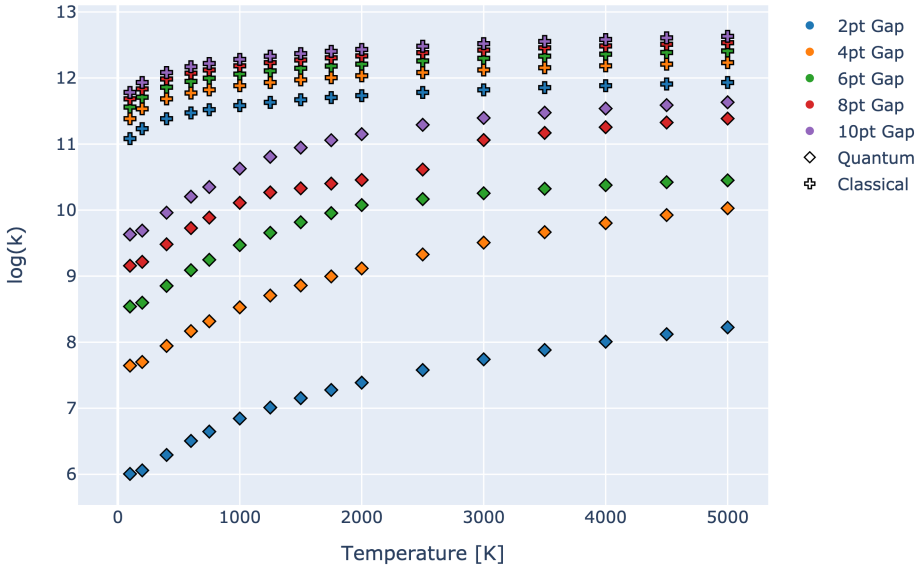

Figure 4.2: *Log of quantum vs. classical rate constants at a 2 point wall thickness. The varied colors denote denote different gap sizes while the shapes signify the formalism. The units of each rate constant were in Hz prior to taking the log.*

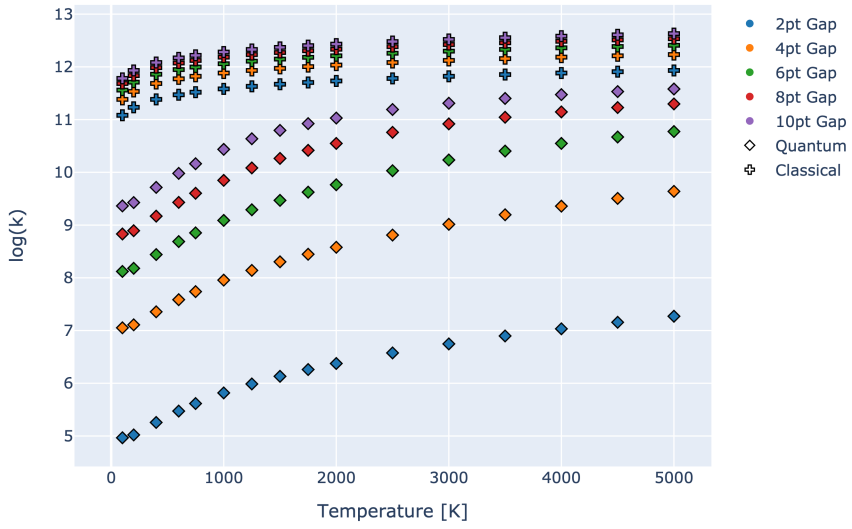

Figure 4.3: *Log of quantum vs. classical rate constants at a 3 point wall thickness. The varied colors denote denote different gap sizes while the shapes signify the formalism. The units of each rate constant were in Hz prior to taking the log.*

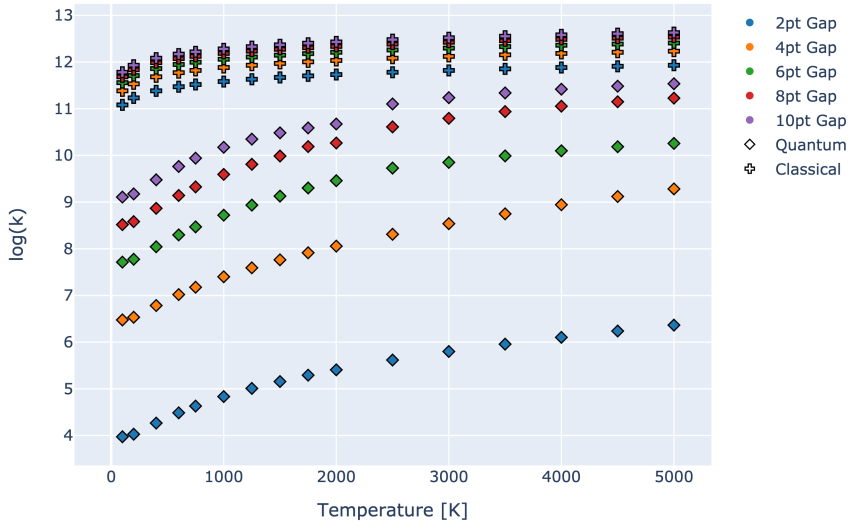

Figure 4.4: *Log of quantum vs. classical rate constants at a 4 point wall thickness. The varied colors denote denote different gap sizes while the shapes signify the formalism. The units of each rate constant were in Hz prior to taking the log.*

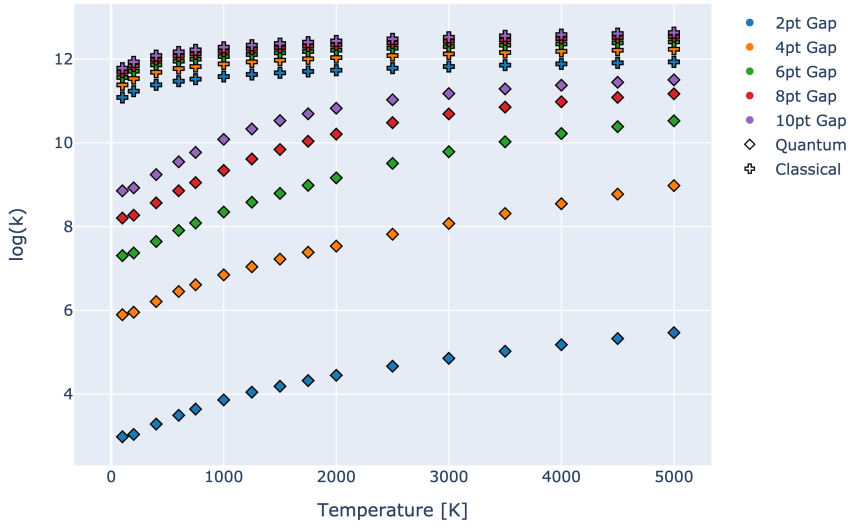

Figure 4.5: *Log of quantum vs. classical rate constants at a 5 point wall thickness. The varied colors denote denote different gap sizes while the shapes signify the formalism. The units of each rate constant were in Hz prior to taking the log.*

The above figures within the chapter show expected deviation between quantum and

classical constants save for the high temperature limit where the correspondence principle appears to hold. Similarly, the intuitive proportionality between gap size and temperature holds, evidenced by larger gap sizes and temperatures producing larger rates. Interestingly, the gap size appears to have a greater effect on the quantum rate constants evidenced by the order(s) of magnitude leaps when comparing points at a constant temperature and different gap sizes to constant gap size and varying temperature. A similar trend can be seen when comparing wall thickness and rates as the thinnest walls yield the greatest quantum values.

One possible rationalization for the gap size vs. temperature contributions to the rate constants can be found by returning to Eqn. 2.13 as we see $k \propto T^{1/2}$. On the other hand, $k \propto A$ denotes that the gap size has a greater contribution to the rate in the classical case and could likely hold true in the quantum regime. It is difficult to make a similar argument for wall thickness and its effects as it is absent in the analytic, classical solution. It nonetheless seems reasonable that wall thickness has a similar effect as gap since they are physically similar parameters.

Where things deviate from expected results come when comparing the relative constants between the classical and quantum formalism. At each wall thickness, there appears to be some "quantum inhibition" where the quantum rates are orders of magnitude lower than the classical ones save for the high temperature, large gap size limit. This could presumably be caused by the discretization scheme, funneling the particle along a certain path rather than giving it free reign to effuse.

Another point of departure from intuition is the lack of rates increasing slightly at low temperatures where quantum mechanical effects such as tunneling and zero-point energy make significant contributions. Instead, we see rate as a monotonically increasing function of temperature and gap size though it's possible sufficiently low temperatures and lengths were not considered. Finally, quantum rate constants extrapolated as wall thickness went to 0 can be found in Table 4.1.

Table 4.1: Extrapolated quantum rate constants k_Q at a wall thickness of zero in units of Hz while temperature and gap size are varied.

Temperature [K]	2 PT. Gap	4 PT. Gap	6 PT. Gap	8 PT. GAP	10 PT. Gap
1.000e+02	2.080e+07	3.155e+08	1.575e+09	4.991e+09	1.236e+10
2.000e+02	2.349e+07	3.572e+08	1.783e+09	5.684e+09	1.398e+10
4.000e+02	3.999e+07	6.182e+08	3.160e+09	1.018e+10	2.480e+10
6.000e+02	6.473e+07	1.019e+09	5.336e+09	1.707e+10	4.067e+10
7.500e+02	8.911e+07	1.422e+09	7.517e+09	2.373e+10	5.379e+10
1.000e+03	1.395e+08	2.275e+09	1.209e+10	3.743e+10	9.867e+10
1.250e+03	2.026e+08	3.365e+09	1.777e+10	6.063e+10	1.410e+11
1.500e+03	2.778e+08	4.711e+09	2.475e+10	8.320e+10	1.860e+11
1.750e+03	3.656e+08	6.294e+09	3.273e+10	1.069e+11	2.321e+11
2.000e+03	4.678e+08	8.145e+09	4.094e+10	1.340e+11	2.783e+11
2.500e+03	7.114e+08	1.267e+10	4.289e+10	1.818e+11	3.030e+11
3.000e+03	1.016e+09	1.812e+10	4.404e+10	2.318e+11	3.965e+11
3.500e+03	1.377e+09	2.463e+10	4.524e+10	2.796e+11	4.560e+11
4.000e+03	1.809e+09	3.170e+10	4.511e+10	3.251e+11	5.099e+11
4.500e+03	2.292e+09	4.056e+10	4.377e+10	3.678e+11	5.589e+11
5.000e+03	2.828e+09	5.214e+10	9.441e+10	4.075e+11	6.038e+11

Similarly to the data that was collected, the proportionality of temperature and gap size to the quantum rates holds in the extrapolated results. Moreover, gap size has a greater impact on the rates than temperature does, evidenced by the rough order in magnitude jump looking across a row opposed to down a column. Figure 4.6 below shows the extrapolated quantum rates plotted against the classical results.

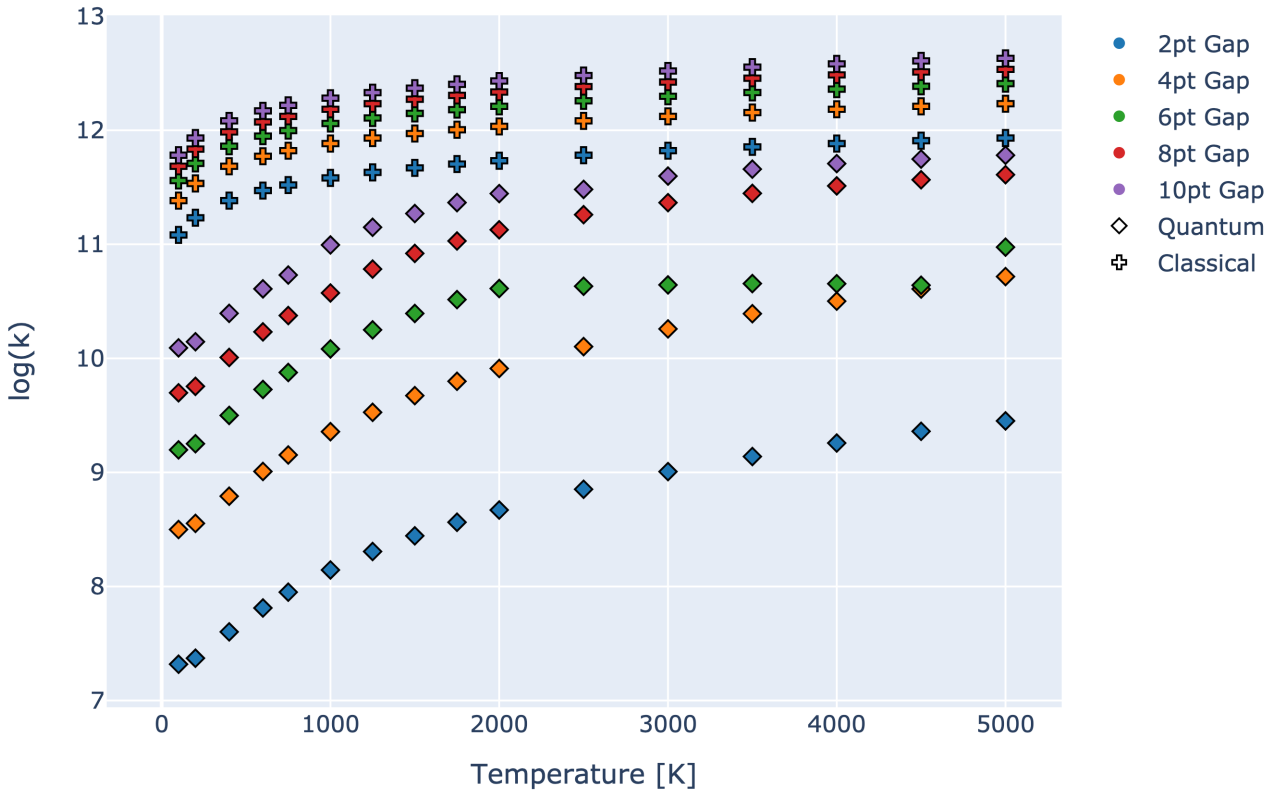

Figure 4.6: *Log of quantum vs. classical rate constants at the extrapolated 0 point wall thickness. The varied colors denote denote different gap sizes while the shapes signify the formalism. The units of each rate constant were in Hz prior to taking the log.*

Even in the case of extrapolated quantum rates, we see $k_q << k_c$ except at the high temperature limit where the correspondence principle is followed. Furthermore, the general trend of rate monotonically increasing with respect to temperature and gap size also persists.

Chapter 5
Conclusion

This study shows that at low temperatures and in small systems, quantum mechanical effusion rate constants k_q greatly deviate from the classical values k_c. Where things differ from expectations are that $k_q << k_c$ in most cases save for the high temperature limit where the correspondence principle is present indicating the potential of the project.

In future avenues of this work, there are some parameters would be worth investigating. One that would be most easily done is the use of a wider range of temperatures and gap sizes, especially at very small values. The reason for this would be to probe whether or not the present system does see rate act as a monotonically increasing function of both temperature and gap size. On the other end of the spectrum, a larger box and heavier particle would be fascinating to consider as the scale of things presented here is only feasible for a hydrogen atom.

Varying the location of the slit is also alluring as disrupting the symmetry of the system would presumably produce fairly different results. Finally, changing the solving methodology from the finite centered difference method to another should be explored given the possible effect of the wavefunction being unable to effuse. Furthermore, it would be computationally infeasible for the number of points in the box to increase proportionally to the size unless the discretization scheme becomes more efficient.

Bibliography

[1] Samantha E. Sloane et al. "Precision Deuteration Using Cu-Catalyzed Transfer Hydrodeuteration to Access Small Molecules Deuterated at the Benzylic Position". In: *JACS Au* 3.6 (2023), pp. 1583–1589. DOI: 10.1021/jacsau.3c00053.

[2] Hnery Eyring. "The Activated Complex in Chemical Reactions". In: *J. Chem. Phys.* 3 (Feb. 1935), pp. 107–115. DOI: https://doi.org/10.1063/1.1749604.

[3] A. B. Balantekin and N. Takigawa. "Quantum tunneling in nuclear fusion". In: *Rev. Mod. Phys.* 70 (1 Jan. 1998), pp. 77–100. DOI: 10.1103/RevModPhys.70.77. URL: https://link.aps.org/doi/10.1103/RevModPhys.70.77.

[4] Matthias Koch et al. "Direct Observation of Double Hydrogen Transfer via Quantum Tunneling in a Single Porphycene Molecule on a Ag(110) Surface". In: *Journal of the American Chemical Society* 139.36 (2017). PMID: 28826219, pp. 12681–12687. DOI: 10.1021/jacs.7b06905. eprint: https://doi.org/10.1021/jacs.7b06905. URL: https://doi.org/10.1021/jacs.7b06905.

[5] Ramamurti Shankar. *Principles of quantum mechanics*. New York, NY: Plenum, 1980. URL: https://cds.cern.ch/record/102017.

[6] David Chandler. "Statistical mechanics of isomerization dynamics in liquids and the transition state approximatio". In: *J. Chem. Phys* 68.6 (1978), pp. 2959–2970. DOI: https://doi.org/10.1063/1.436049.

[7] William H. Miller Xiong Sun Haobin Wang. "Semiclassical theory of electronically nonadiabatic dynamics: Results of a linearized approximation to the initial value representation". In: *J. Chem. Phys* 109.17 (1998), pp. 7064–7074. DOI: https://doi.org/10.1063/1.477389.

[8] William H. Miller, Steven D. Schwartz, and John W. Tromp. "Quantum mechanical rate constants for bimolecular reactions". In: *Chem Phys* 79 (Nov. 1983), pp. 4889–4898. DOI: https://doi.org/10.1063/1.445581. URL: https://pubs.aip.org/aip/jcp/article/79/10/4889/446769/Quantum-mechanical-rate-constants-for-bimolecular.

[9] David E. Manolopoulos. *Chemical Reaction Dynamics*. 2008. URL: http://manolopoulos.chem.ox.ac.uk/.

[10] "Chapter 1: Brief Summary of Finite Difference Methods". In: *A Primer on Radial Basis Functions with Applications to the Geosciences*, pp. 1–17. DOI: 10.1137/1.9781611974041.ch1.

Chapter A
Code

The code to produce the Hamiltonian, flux squared matrix, and calculate correlation functions are found below.

```
#/* ----------------------------------------------------------------------------
#   Egorov Group
#   University of Virginia
#   Mohan Shankar
#
#   2d_slit.py
#   "This file calculates eigenfunctions of P.I.B. in the presence of a slit"
#---------------------------------------------------------------------------- */
# DEPENDENCIES
import numpy as np
import matplotlib.pyplot as plt
#---------------------------------------------------------------------------- */
# INPUTS
me = 5.485799e-4 # Electron mass in daltons
m_au = 1.0 / me

lx = 1.0 # Angstroms
ly = 8.0 # Angstroms

a0 = 0.529177258 # Bohr radius

lx_au = lx/a0
ly_au = ly/a0

nnx = 60 # number of x points
nny = 480 # number of y points

xmin = 0
xmax = lx_au

ymin = 0
```

```
ymax = ly_au

dx = (xmax-xmin)/(nnx+1)
dy = (ymax-ymin)/(nny+1)

pi = np.pi

nd = 2 # length of stencil for derivative across dividing surface
wall_thickness = 5
cd = np.zeros(nd)
nn = 10000 # times to consider

gap_size = 4.0
#------------------------------------------------------------------------------ */
# FUNCTION DEFINITIONS

def kron_sum(A1, A2):
    '''
    Assumes A1, A2 are nxn, mxm matrices where n can be equal to m
    '''
    i1 = np.identity(len(A1[0]))
    i2 = np.identity(len(A2[0]))
    return np.kron(A1,i2) + np.kron(i1, A2)


def PIB_one(points, lmax, lmin, mass, hbar=1):
    '''
    function to create Hamiltonian for 1D PIB
    '''
    dn = (lmax-lmin)/(points+1)
    dn2 = dn**2  # second derivative
    H = np.zeros((points, points)) # initialize
    z = -pi**2/3.0  # weight for diagonal
    for i in range(points):
        for j in range(points):
            if i == j:
                H[i][j] = z # weight for diagonals of matrix
            else:
                H[i][j] = (2/(i-j)**2)*((-1)**(i-j+1))  # weights for non-diagonals
    H *= (-1/(2*mass*dn2)) # hbar = 1 hence 1/(2 * mass * dn2)
    return H
#------------------------------------------------------------------------------ */
# CREATE MATRICES
h1_x = PIB_one(nnx, xmax, xmin, m_au) # 1-D Hamiltonian from x points
h1_y = PIB_one(nny, ymax, ymin, m_au) # 1-D Hamiltonian from y points
```

```
H = kron_sum(h1_x, h1_y) # 2-D Hamiltonian
# STENCIL FOR FLUX ALONG DIVIDING SURFACE
for k in range(nd):
    zz = 1.0
    for j in range(nd):
        if j != k:
            zz *= ((j+1)**2) / ((j+1)**2 - (k+1)**2)
        else:
            continue
    cd[k] = zz/(2*(k+1))

print("Matrices created!")
#------------------------------------------------------------------------------ */
# MAKE EFFUSION SLIT
i = -1

for k in range(nnx):
    for j in range(nny):
        i = i+1
        x = dx * k + xmin
        y = dy * j + ymin
        if (1.0 / a0) < y < (1.0 / a0 + wall_thickness * dy):
            if x <(0.5 * xmax - (gap_size/2.0) * dx) or x >(0.5 * xmax + (gap_size/2.0)
                H[i] = np.zeros(nnx * nny)
                H[:, i] = np.zeros(nnx * nny)
                H[i, i] = -1
print("Slit created!")
#------------------------------------------------------------------------------ */
# CALCULATE EIGENVECTORS AND VALUES
eigvals, eigvecs = np.linalg.eigh(H) # find eigenvalues and eigenvectors
print("Eigs found!")
#------------------------------------------------------------------------------ */
# CLEANUP OF DATA
psi = np.transpose(eigvecs) #  transpose for easier indexing

# as vector are returned in column form

psi = psi[np.argsort(eigvals)]
energies = eigvals[np.argsort(eigvals).real]

cut = np.where(energies > 0)
print(cut)
energies = energies[cut]
psi = psi[cut]
```

```
for i in range(len(psi)):
    normalization = np.sqrt(np.sum(psi[i]**2 * dy * dx))
    psi[i] = psi[i]/normalization

np.savez("eigs_file", psi = psi, eigvals = eigvals)
print("Eigs saved!")
#------------------------------------------------------------------------------ */
# FIRST DERIVATIVE OVER DIVIDING SURFACE

dpsi = np.zeros((nn, nnx))

y = (1/a0) + 1.0 * dy
i0 = int((y-ymin)/dy)

for j in range(nn):
    for i2 in range(nnx):
        ix = nny * i2 + i0
        zz = 0.0
        for k in range(nd):
            kk = k + 1
            zz += cd[k] * (psi[j][ix + kk] - psi[j][ix-kk])
        zz = zz/dy
        dpsi[j, i2] = zz

print("First derivative over dividing surface found!")
#------------------------------------------------------------------------------ */
# CALCULATION OF FLUX SQUARED

fx2 = np.zeros((nn, nn))

for j1 in range(nn):
    for j2 in range(nn):
        zz = 0.0
        for i2 in range(nnx):
            ix = nny * i2 + i0
            zz0 = dpsi[j1, i2] * psi[j2, ix] - psi[j1, ix] * dpsi[j2, i2]
            zz0 = zz0 * dy / (2.0 * m_au)
            zz += zz0
        zz = zz**2
        fx2[j1, j2] = zz

with open("Energy.npz", "wb") as f:
    np.savez(f, energies = energies, fx2 = fx2, )
```

```
print("Job was successfully completed!")
#------------------------------------------------------------------------------ */

#/* ----------------------------------------------------------------------------
#   Egorov Group
#   University of Virginia
#   Mohan Shankar
#
#   correlation.py
#   "This code calculates correlation functions for a given system in parallel"
#------------------------------------------------------------------------------ */
# DEPENDENCIES
import numpy as np
from multiprocessing import Pool # relevant package for parallel processing
from datetime import datetime
#------------------------------------------------------------------------------ */
# INPUTS & LOAD IN DATA (PART 1)

start_time = datetime.now()
print("Starting!", start_time)
kb = 3.166830e-6 # Boltzmann constant in hartree
har = 2.194746e5 # hartree in cm^{-1}

Trange = np.array([100, 200, 400, 600, 750, 1000, 1250, 1500, 1750,
                   2000, 2500, 3000, 3500, 4000, 4500, 5000])

tmin = 0
tmax = 10000
nt = 1000
dt = (tmax - tmin)/nt

hbar = 1.0

with np.load('Energy.npz') as data:
    E = np.array([data['energies']])
    fx2 = data['fx2']

nn = np.shape(fx2)[0]
# print("Energies", E)
epsilon= np.log(1e-16)
zero = E[0]
dE = E - E.transpose()
sE = E + E.transpose()

dEm = dE.copy()
```

```
np.fill_diagonal(dEm, 1)

ttot = np.arange(tmin, tmax+1, dt)
CF = np.zeros((2, nt+1))
CS = np.zeros((2, nt+1))
CF[0] = ttot
CS[0] = ttot

end_time =  datetime.now()
print('Duration of Part One: {}'.format(end_time - start_time))
#------------------------------------------------------------------------------ */
# Correlation Functions (PART 2)

start_time = datetime.now()

def f(T):
    beta = 1/(kb*T)
    cut = 2 * epsilon/beta
    over = np.where(E > zero - cut)
    if np.shape(over) == 0:
        print(f'Not converged at {T}')
        Q = nn
    else:
        print(f'At {T}K, {over[0]} functions are significant')
        if over[0] <= nn:
            Q = over[0]
        else:
            Q = nn

    for time in range(nt + 1):
        t = tmin + time * dt
        zz = np.exp(-beta * 0.5 * sE[0:Q, 0:Q]) * fx2[0:Q, 0:Q]
        CF1 = zz * np.cos(dE[0:Q, 0:Q] * t)
        zzd = np.diagonal(zz)
        CS1 = zz * np.sin(dE[0:Q, 0:Q] * t) / dEm[0:Q, 0:Q]
        CSd = zzd * t
        np.fill_diagonal(CS1, CSd)

        CF[1, time] = np.sum(CF1)
        CS[1, time] = np.sum(CS1)

    return CS, CF

if __name__ == '__main__':
```

```
    pool = Pool(processes=10) # allocate number of cores for parallel processing
    fargs=zip(Trange)

    result = pool.starmap(f, fargs) # result is a list of all values for CS, CF
    # at a given T

    # result is a 4-D object; result[a][b][c][d] --> a denotes which temp;
    # b = 0 --> CS; b = 1 --> CF
    # [c][d] are then indices for a 2D matrix where the first column
    # corresponds to time and the second the relevant correlation function

    for i, val in enumerate(Trange):
        np.savetxt('side-flux'+str(val)+'K.txt',
                   np.transpose((result[i][0][0], result[i][0][1])),
                   delimiter = ',', header="Time , Side-Flux", fmt='%1.4e')

        # result[i][a][b] means Trange[i]; [a] = 0 means Flux-Side (CS);
        # [b] = 0 means time column; b = 1 means correlation function;
        # free index [d] corresponds to a particular element in column so it's unused

    for i, val in enumerate(Trange):
        np.savetxt('flux-flux'+str(val)+'K.txt',
                   np.transpose((result[i][1][0], result[i][1][1])),
                   delimiter = ',', header="Time , Flux-Flux", fmt='%1.4e')

        # result[i][a][b] means Trange[i]; [a] = 1 means Flux-Flux (CF) [b] = 0 means
        # time column while b = 1 means correlation function;
        # [c][d] are then indices for a 2D matrix where the first column
        # corresponds to time and the second the relevant correlation function

    pool.close()
    pool.join()


end_time =  datetime.now()
print('Duration of Part Two: {}'.format(end_time - start_time))
#------------------------------------------------------------------------------ */
```

Chapter B
Example Calculation for Rate Constants

Returning to Fig. 3.3, the y-value associated with the point where the black dashed line meets the correlation function is $C_{fs}(3190) = 7.7884 \times 10^{-7}$ hartree. Recall $k_r(T) = C_{fs}/Q_r(T)$ thus we must find $Q_r(T)$. The partition function was numerically calculated using the following code:

```
#/* ----------------------------------------------------------------------------
#   Egorov Group
#   University of Virginia
#   Mohan Shankar
#
# partition_function.py
#   "This file calculates numerical partition functions for a PIB system"
#---------------------------------------------------------------------------- */
# DEPENDENCIES
import numpy as np
import matplotlib.pyplot as plt
#---------------------------------------------------------------------------- */
Trange = np.array([100, 200, 400, 600, 750, 1000, 1250, 1500, 1750,
                   2000, 2500, 3000, 3500, 4000, 4500, 5000])

T = 1000 # arbitrary temperature

temp_index = np.where(Trange == T) # get index of Trange array where T match
#---------------------------------------------------------------------------- */
# CALCULATE PARTITION FUNCTIONS FOR DIFFERENT TEMPERATURES
hbar = 1.054571817e-34 # hbar in SI Units [J*s]

m = 1.67262192e-27 # mass of proton [kg]

L = 1e-10 # 1 Angstrom so 1e-10 [m]

kb = 1.380649e-23 # Boltzmann const. in [J/K]
```

```
n = np.arange(1, 21, 1)
Q = np.empty_like(Trange)# init. empty array w/ same dim as Trange
E = (np.pi ** 2 * hbar ** 2) / (2 * m * L ** 2)

for i, temp in enumerate(Trange):
    T = temp
    beta = 1 / (kb * T)
    zz = np.exp(-beta * E * n** 2)
    final = np.sum(zz)
    Q[i] = final**2
#------------------------------------------------------------------------------ */
print(Q[temp_index]) # find Q for a given temperature
```

The numeric value of $Q_r(1000) = 1.7369$, so we find

$$k_r(1000) = \frac{7.7884 \times 10^{-7} \text{ hartree}}{1.7369} = 4.4841 \times 10^{-7} \text{ hartree}.$$

Finally, we convert from hartree to inverse seconds by using 1 hartree is 6.57966×10^{15} Hz:

$$k_r(1000) = 4.4841 \times 10^{-7} \text{ hartree} \cdot \frac{6.57966 \times 10^{15} \text{ Hz}}{1 \text{ hartree}} = \mathbf{2.95 \times 10^{9} \text{ Hz}.}$$