Advanced Quantum Mechanics

Chem 572a: Lecture Notes

Prof. Victor S. Batista

Room: Sterling Chemistry Laboratories (SCL) 19
Tuesdays and Thursdays 9:00 – 10:15 am
Yale University - Department of Chemistry

Contents

1 Syllabus

The goal of this course is to introduce fundamental concepts of *Quantum Mechanics* with emphasis on Quantum Dynamics and its applications to the description of molecular systems and their interactions with electromagnetic radiation. Quantum Mechanics involves a *mathematical formulation* and a *physical interpretation*, establishing the correspondence between the mathematical elements of the theory (e.g., functions and operators) and the elements of reality (e.g., the observable properties of real systems). The presentation of the theory will be mostly based on the so-called *Orthodox Interpretation*, developed in Copenhagen during the first three decades of the 20th century. However, other interpretations will be discussed, including the 'pilot-wave' theory first suggested by Pierre De Broglie in 1927 and independently rediscovered by David Bohm in the early 1950's.

Textbooks: The official textbook for this class is:
R1:"Introduction to Quantum Mechanics: A Time-Dependent Perspective" by David J. Tannor (University Science Books).
However, the lectures will be heavily complemented with material from other textbooks including:
R2: "Quantum Theory" by David Bohm (Dover),
R3: "Quantum Physics" by Stephen Gasiorowicz (Wiley),
R4: "Quantum Mechanics" by Claude Cohen-Tannoudji (Wiley Interscience),
R5: "Quantum Mechanics" by E. Merzbacher (Wiley),
R6: "Modern Quantum Mechanics" by J. J. Sakurai (Addison Wesley),
All these references are 'on-reserve' at the Kline science library.
References to specific pages of the textbooks listed above are indicated in the notes as follows: R1(190) indicates "for more information see Reference 1, Page 190".
Furthermore, a useful mathematical reference is R. Shankar, Basic Training in Mathematics. A Fitness Program for Science Students, Plenum Press, New York 1995.
Useful search engines for mathematical and physical concepts can be found at
http://scienceworld.wolfram.com/physics/ and http://mathworld.wolfram.com/
The lecture notes are posted online at: (http://ursula.chem.yale.edu/∼batista/classes/v572/v572.pdf)

Grading: There will be no final exam for this class.
The final grading evaluation is the same for both undergraduate and graduate students:
homework (50%), and three mid-terms (50%) on 10/02/08, 11/06/08 and 12/02/08.
Homework includes exercises and computational assignments with due dates indicated in the lecture notes.

Contact Information and Office Hours: Office hours will be held at SCL 239 as requested by the students via email to victor.batista@yale.edu, or phone at (203) 432-6672.

2   The Fundamental Postulates of Quantum Mechanics

Quantum Mechanics can be formulated in terms of a few *postulates* (*i.e.*, theoretical principles based on experimental observations). The goal of this section is to introduce such principles, together with some mathematical concepts that are necessary for that purpose. To keep the notation as simple as possible, expressions are written for a 1-dimensional system. The generalization to many dimensions is usually straightforward.

$\boxed{Postulate\ 1}$: *Any system in pure state can be described by a function $\psi(t, x)$, where t is a parameter representing the time and x represents the coordinates of the system. Function $\psi(t, x)$ must be continuous, single valued and square integrable.*
Note 1: As a consequence of Postulate 4, we will see that $P(t, x) = \psi^*(t, x)\psi(t, x)\,dx$ represents the probability of finding the system between x and $x + dx$ at time t.

$\boxed{Postulate\ 2}$: *Any observable (i.e., any measurable property of the system) can be described by an operator. The operator must be linear and hermitian.*
What is an *operator* ? What is a *linear operator* ? What is a *hermitian operator*?

Definition 1: An operator $\hat{O}$ is a mathematical entity that transforms a function $f(x)$ into another function $g(x)$ as follows, **R4(96)**
$$\hat{O}f(x) = g(x),$$

where f and g are functions of x.

Definition 2: An operator $\hat{O}$ that represents an observable O is obtained by first writing the classical expression of such observable in Cartesian coordinates (e.g., $O = O(x, p)$) and then substituting the coordinate x in such expression by the *coordinate operator* $\hat{x}$ as well as the momentum p by the *momentum operator* $\hat{p} = -i\hbar\partial/\partial x$.

Such a definition of the momentum operator is equivalent to the eigenvalue equation:

$$\hat{p}\delta(p - p_j) = p_j\delta(p - p_j)$$

since according to the Fourier expansion $\delta(p - p_j) = \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} dx\, e^{\frac{i}{\hbar}x(p-p_j)}$, we obtain:

$$
\begin{aligned}
\hat{p}\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} dx\, e^{\frac{i}{\hbar}x(p-p_j)} &= p_j\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} dx\, e^{\frac{i}{\hbar}x(p-p_j)} \\
\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} dx\, e^{-\frac{i}{\hbar}xp_j}\hat{p}\,e^{\frac{i}{\hbar}xp} &= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} dx\, e^{\frac{i}{\hbar}xp}i\hbar\frac{\partial}{\partial x}e^{-\frac{i}{\hbar}xp_j} \\
&= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} dx\, e^{\frac{i}{\hbar}xp}i\hbar\frac{\partial}{\partial x}e^{-\frac{i}{\hbar}xp_j} \\
&= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} dx\, e^{-\frac{i}{\hbar}xp_j}(-i\hbar)\frac{\partial}{\partial x}e^{\frac{i}{\hbar}xp}
\end{aligned}
\tag{1}
$$

where in the last row of Eq. (1) we have integrated by parts.

Definition 3: An operator $\hat{O}$ is *linear* if and only if (iff),

$$\hat{O}(af(x) + bg(x)) = a\hat{O}f(x) + b\hat{O}g(x),$$

where a and b are constants.

Definition 4: An operator $\hat{O}$ is *hermitian* iff,

$$\int dx \phi_n^*(x)\hat{O}\psi_m(x) = \left[\int dx \psi_m^*(x)\hat{O}\phi_n(x)\right]^*,$$

where the asterisk represents the complex conjugate of the expression embraced by brackets.

Definition 5: A function $\phi_n(x)$ is an *eigenfunction* of $\hat{O}$ iff,

$$\hat{O}\phi_n(x) = O_n\phi_n(x),$$

where O_n is a number called *eigenvalue*.

Property 1: The eigenvalues of a hermitian operator are real.

Proof: Using Definition 4, we obtain

$$\int dx \phi_n^*(x)\hat{O}\phi_n(x) - \left[\int dx \phi_n^*(x)\hat{O}\phi_n(x)\right]^* = 0,$$

therefore,

$$[O_n - O_n^*] \int dx \phi_n(x)^* \phi_n(x) = 0.$$

Since $\phi_n(x)$ are square integrable functions, then,

$$O_n = O_n^*.$$

Property 2: Different eigenfunctions of a hermitian operator (i.e., eigenfunctions with different eigenvalues) are orthogonal (i.e., the *scalar product* of two different eigenfunctions is equal to zero). Mathematically, if $\hat{O}\phi_n = O_n\phi_n$, and $\hat{O}\phi_m = O_m\phi_m$, with $O_n \neq O_m$, then $\int dx \phi_n^* \phi_m = 0$.

Proof:

$$\int dx \phi_m^* \hat{O}\phi_n - \left[\int dx \phi_n^* \hat{O}\phi_m\right]^* = 0,$$

and

$$[O_n - O_m] \int dx \phi_m^* \phi_n = 0.$$

Since $O_n \neq O_m$, then $\int dx \phi_m^* \phi_n = 0$.

$\boxed{Postulate\ 3}$: *The only possible experimental results of a measurement of an observable are the eigenvalues of the operator that corresponds to such observable.*

$\boxed{Postulate\ 4}$: *The average value of many measurements of an observable O, when the system is described by function $\psi(x)$, is equal to the expectation value $\bar{O}$, which is defined as follows,*

$$\bar{O} = \frac{\int dx\psi(x)^*\hat{O}\psi(x)}{\int dx\psi(x)^*\psi(x)}.$$

$\boxed{Postulate\ 5}$: The evolution of $\psi(x,t)$ in time is described by the time-dependent Schrödinger equation:

$$i\hbar\frac{\partial\psi(x,t)}{\partial t} = \hat{H}\psi(x,t), \tag{2}$$

where $\hat{H} = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + \hat{V}(x)$, is the operator associated with the total energy of the system, $E = \frac{p^2}{2m} + V(x)$.

$\boxed{Expansion\ Postulate}$: **R5(15), R4(97)**

The eigenfunctions of a linear and hermitian operator form a complete basis set. Therefore, any function $\psi(x)$ that is continuous, single valued, and square integrable can be expanded as a linear combination of eigenfunctions $\phi_n(x)$ of a linear and hermitian operator $\hat{A}$ as follows,

$$\psi(x) = \sum_j C_j\phi_j(x),$$

where C_j are numbers (e.g., complex numbers) called *expansion coefficients*.
Note that $\bar{A} = \sum_j C_j C_j^* a_j$, when $\psi(x) = \sum_j C_j\phi_j(x)$,

$$\hat{A}\phi_j(x) = a_j\phi_j(x), \quad \text{and} \quad \int dx\phi_j(x)^*\phi_k(x) = \delta_{jk}.$$

This is because the eigenvalues a_j are the only possible experimental results of measurements of $\hat{A}$ (according to Postulate 3), and the expectation value $\bar{A}$ is the average value of many measurements of $\hat{A}$ when the system is described by the expansion $\psi(x) = \sum_j C_j\phi_j(x)$ (Postulate 4). Therefore, the product $C_j C_j^*$ can be interpreted as the probability weight associated with eigenvalue a_j (i.e., the probability that the outcome of an observation of $\hat{A}$ will be a_j).

Hilbert-Space

According to the Expansion Postulate (together with Postulate 1), the state of a system described by the *function* $\Psi(x)$ can be expanded as a linear combination of eigenfunctions $\phi_j(x)$ of a linear and hermitian operator (e.g., $\Psi(x) = C_1\phi_1(x) + C_2\phi_2(x) + \ldots$). Usually, the space defined by these eigenfunctions (i.e., functions that are continuous, single valued and square integrable) has an infinite number of dimensions. Such space is called *Hilbert-Space* in honor to the mathematician Hilbert who did pioneer work in spaces of infinite dimensionality.**R4(94)**

A representation of $\Psi(x)$ in such space of functions corresponds to a vector-function,

11
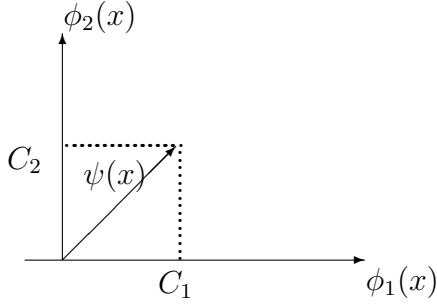
where C_1 and C_2 are the projections of $\Psi(x)$ along $\phi_1(x)$ and $\phi_2(x)$, respectively. All other components are omitted from the representation because they are orthogonal to the "plane" defined by $\phi_1(x)$ and $\phi_2(x)$.

3 Continuous Representations

Certain operators have a continuous spectrum of eigenvalues. For example, the coordinate operator is one such operator since it satisfies the equation $\hat{x}\,\delta(x_0 - x) = x_0\,\delta(x_0 - x)$, where the eigenvalues x_0 define a *continuum*. Delta functions $\delta(x_0 - x)$ thus define a continuous representation (the so-called 'coordinate representation') for which

$$\psi(x) = \int dx_0 C_{x_0}\delta(x_0 - x),$$

where $C_{x_0} = \psi(x_0)$, since

$$\int dx\delta(x - \beta)\psi(x) = \int dx \int d\alpha C_\alpha \delta(x - \beta)\delta(\alpha - x) = \psi(\beta).$$

When combined with postulates 3 and 4, the definition of the expansion coefficients $C_{x_0} = \psi(x_0)$ implies that the probability of observing the system with coordinate eigenvalues between x_0 and $x_0 + dx_0$ is $P(x_0) = C_{x_0}C_{x_0}^* dx_0 = \psi(x_0)\psi(x_0)^* dx_0$ (see Note 1).

In general, eigenstates $\phi(\alpha, x)$ with a continuum spectrum of eigenvalues α define continuous representations,

$$\psi(x) = \int d\alpha C_\alpha \phi(\alpha, x),$$

with $C_\alpha = \int dx\phi(\alpha, x)^*\psi(x)$. Delta functions and the plane waves are simply two particular examples of basis sets with continuum spectra.

Note 2: According to the Expansion Postulate, a function $\psi(x)$ is uniquely and completely defined by the coefficients C_j, associated with its expansion in a complete set of eigenfunctions $\phi_j(x)$. However, the coefficients of such expansion would be different if the same basis functions ϕ_j depended on different coordinates (e.g., $\phi_j(x')$ with $x' \neq x$). In order to eliminate such ambiguity in the description it is necessary to introduce the concept of *vector-ket* space.**R4(108)**

4 Vector Space

Vector-Ket Space ε: The vector-ket space is introduced to represent states in a convenient space of *vectors* $|\phi_j>$, instead of working in the space of *functions* $\phi_j(x)$. The main difference is that the coordinate dependence does not need to be specified when working in the vector-ket space. According to such representation, function $\psi(x)$ is the *component* of vector $|\psi>$ associated with index x (*vide infra*). Therefore, for any function $\psi(x) = \sum_j C_j \phi_j(x)$, we can define a ket-vector $|\psi>$ such that,

$$|\psi> = \sum_j C_j |\phi_j>.$$

The representation of $|\psi>$ in space ε is,
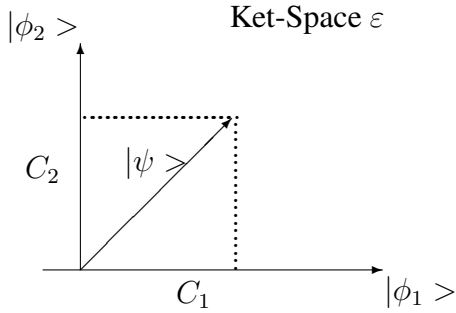
Note that the expansion coefficients C_j depend only on the kets $|\psi_j>$ and not on any specific vector component. Therefore, the ambiguity mentioned above is removed.

In order to learn how to operate with kets we need to introduce the *bra space* and the concept of *linear functional*. After doing so, this section will be concluded with the description of *Postulate 5*, and the *Continuity Equation*.

Linear functionals

A functional χ is a mathematical operation that transforms a function $\psi(x)$ into a number. This concept is extended to the vector-ket space ε, as an operation that transforms a vector-ket into a number as follows,

$$\chi(\psi(x)) = n, \quad \text{or} \quad \chi(|\psi>) = n,$$

where n is a number. A *linear* functional satisfies the following equation,

$$\chi(a\psi(x) + bf(x)) = a\chi(\psi(x)) + b\chi(f(x)),$$

where a and b are constants.
Example: The scalar product,**R4(110)**

$$n = \int dx \psi^*(x)\phi(x),$$

is an example of a linear functional, since such an operation transforms a function $\phi(x)$ into a number n. In order to introduce the scalar product of kets, we need to introduce the *bra-space*.

Bra Space ε^*: For every ket $|\psi>$ we define a linear functional $<\psi|$, called *bra-vector*, as follows:

$$<\psi|(|\phi>) = \int dx\, \psi^*(x)\phi(x).$$

Note that functional $<\psi|$ is linear because the scalar product is a linear functional. Therefore,

$$<\psi|(a|\phi> + b|f>) = a<\psi|(|\phi>) + b<\psi|(|f>).$$

Note: For convenience, we will omit parenthesis so that the notation $<\psi|(|\phi>)$ will be equivalent to $<\psi||\phi>$. Furthermore, whenever we find two bars next to each other we can merge them into a single one without changing the meaning of the expression. Therefore,

$$<\psi||\phi> = <\psi|\phi>.$$

The space of bra-vectors is called dual space ε^* simply because given a ket $|\psi> = \sum_j C_j|\phi_j>$, the corresponding bra-vector is $<\psi| = \sum_j C_j^* <\phi_j|$. In analogy to the ket-space, a bra-vector $<\psi|$ is represented in space ε^* according to the following diagram:
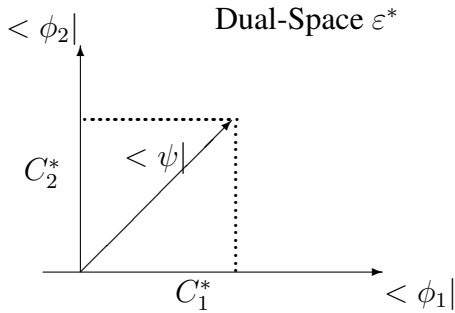
where C_j^* is the projection of $<\psi|$ along $<\phi_j|$.

Projection Operator and Closure Relation
Given a ket $|\psi>$ in a certain basis set $|\phi_j>$,

$$|\psi> = \sum_j C_j|\phi_j>, \tag{3}$$

where $<\phi_k|\phi_j> = \delta_{kj}$,

$$C_j = <\phi_j|\psi>. \tag{4}$$

Substituting Eq. (2) into Eq.(1), we obtain

$$|\psi> = \sum_j |\phi_j><\phi_j|\psi>. \tag{5}$$

From Eq.(3), it is obvious that

$$\sum_j |\phi_j><\phi_j| = \hat{1}, \qquad Closure\ Relation$$

where $\hat{1}$ is the identity operator that transforms any ket, or function, into itself.

Note that $\hat{P}_j = |\phi_j><\phi_j|$ is an operator that transforms any vector $|\psi>$ into a vector pointing in the direction of $|\phi_j>$ with magnitude $<\phi_j|\psi>$. The operator $\hat{P}_j$ is called the *Projection Operator*. It projects $|\phi_j>$ according to,

$$\hat{P}_j|\psi> = <\phi_j|\psi> |\phi_j>.$$

Note that $\hat{P}_j^2 = \hat{P}_j$, where $\hat{P}_j^2 = \hat{P}_j\hat{P}_j$. This is true simply because $<\phi_j|\phi_j> = 1$.

4.1 Exercise 1

Prove that

$$i\hbar\frac{\partial \hat{P}_j}{\partial t} = [\hat{H}, \hat{P}_j],$$

where $[\hat{H}, \hat{P}_j] = \hat{H}\hat{P}_j - \hat{P}_j\hat{H}$.

Continuity Equation

4.2 Exercise 2

Prove that

$$\frac{\partial(\psi^*(x,t)\psi(x,t))}{\partial t} + \frac{\partial}{\partial x}j(x,t) = 0,$$

where

$$j(x,t) = \frac{\hbar}{2mi}\left(\psi^*(x,t)\frac{\partial\psi(x,t)}{\partial x} - \psi(x,t)\frac{\partial\psi^*(x,t)}{\partial x}\right).$$

In general, for higher dimensional problems, the change in time of probability density, $\rho(\mathbf{x}, t) = \psi^*(\mathbf{x}, t)\psi(\mathbf{x}, t)$, is equal to minus the divergence of the probability flux $\mathbf{j}$,

$$\frac{\partial\rho(\mathbf{x}, t)}{\partial t} = -\nabla\cdot\mathbf{j}.$$

This is the so-called *Continuity Equation*.

Note: Remember that given a vector field $\mathbf{j}$, e.g., $\mathbf{j}(x, y, z) = j_1(x, y, z)\hat{i} + j_2(x, y, z)\hat{j} + j_3(x, y, z)\hat{k}$, the divergence of $\mathbf{j}$ is defined as the dot product of the "del" operator $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z})$ and vector $\mathbf{j}$ as follows:

$$\nabla \cdot \mathbf{j} = \frac{\partial j_1}{\partial x} + \frac{\partial j_2}{\partial y} + \frac{\partial j_3}{\partial z}.$$

Note on the Fifth Postulate: We note that when $|\psi\rangle$ is an eigenstate of the Hamiltonian with eigenvalue E (*i.e.*, $|\psi\rangle = |E\rangle$), it is orthogonal to eigenfunctions with different eigenvalues. Therefore,

$$\begin{aligned}
\langle E_0 | E \rangle &= \delta(E_0 - E), \\
&= \frac{1}{2\pi\hbar} \int dt e^{\frac{i}{\hbar}(E_0 - E)t},
\end{aligned} \tag{6}$$

which implies that

$$|E\rangle = \frac{1}{\sqrt{2\pi\hbar}} e^{-\frac{i}{\hbar}Et}. \tag{7}$$

In addition, $|E\rangle$ satisfies the eigenvalue equation,

$$\hat{H}|E\rangle = E|E\rangle,$$

which is equivalent to the time-dependent Schrödinger equation since

$$\begin{aligned}
\hat{H}|E\rangle &= E|E\rangle, \\
&= E \frac{1}{\sqrt{2\pi\hbar}} e^{-\frac{i}{\hbar}Et}, \\
&= i\hbar \frac{\partial}{\partial t} \frac{1}{\sqrt{2\pi\hbar}} e^{-\frac{i}{\hbar}Et}, \\
&= i\hbar \frac{\partial}{\partial t} |E\rangle.
\end{aligned} \tag{8}$$

When $|\psi\rangle$ is not an eigenstate of the Hamiltonian, it can be expanded as a superposition of eigenstates of $\hat{H}$, as follows $|\psi\rangle = \sum_j c_j |E_j\rangle$. Therefore,

$$\begin{aligned}
\hat{H}|\psi\rangle &= \hat{H} \sum_j c_j |E_j\rangle \\
&= \sum_j c_j E_i |E_j\rangle, \\
&= \sum_j c_j E_j \frac{1}{\sqrt{2\pi\hbar}} e^{-\frac{i}{\hbar}E_j t}, \\
&= i\hbar \sum_j c_j \frac{\partial}{\partial t} \frac{1}{\sqrt{2\pi\hbar}} e^{-\frac{i}{\hbar}Et},
\end{aligned} \tag{9}$$

Assuming that c_j are time independent, we obtain:

$$\hat{H}|\psi\rangle = i\hbar\frac{\partial}{\partial t}\sum_j c_j|E_j\rangle,$$

$$= i\hbar\frac{\partial}{\partial t}|\psi\rangle, \tag{10}$$

which is the fifth postulate, $\hat{H} = i\hbar\frac{\partial}{\partial t}$. Therefore, we find that the assumption that the expansion coeficients c_j remain constant is equivalent to the fifth postulate and implies that the probability of observing the system with energy E_j is time invariant, or that the expectation value of $\hat{H}$ is time invariant. Therefore, the fifth postulate implies that quantum dynamics is ruled by energy conservation.

5   Digital Grid-Based Representations

The standard formulation of quantum mechanics, presented in previous sections, relies upon the tools of *calculus* (*e.g.*, derivatives, integrals, etc.) and involves equations and operations with infinitesimal quantities as well as states in Hilbert-space (the infinite dimensional space of functions L^2). The equations, however, seldom can be solved analytically. Therefore, computational solutions are necessary. However, computers can not handle infinite spaces since they have only limited memory. In fact, all they *can* do is to store and manipulate discrete arrays of numbers. Therefore, the question is: how can we represent continuum states and operators in the space of memory of digital computers?

In order to introduce the concept of a *grid-representation*, we consider the state,

$$\Psi_0(x) = \left(\frac{\alpha}{\pi}\right)^{1/4} e^{-\frac{\alpha}{2}(x-x_0)^2 + ip_0(x-x_0)}, \tag{11}$$

which can be expanded in the infinite basis set of delta functions $\delta(x-x')$ as follows,

$$\Psi_0(x) = \int dx' c(x')\delta(x-x'), \tag{12}$$

where $c(x') \equiv \langle x'|\Psi_0\rangle = \Psi_0(x')$. All expressions are written in atomic units, so $\hbar = 1$.

A grid-based representation of $\Psi_0(x)$ can be obtained, in the coordinate range $x = (x_{min}, x_{max})$, by discretizing Eq. (12) as follows,

$$\Psi_0(x) = \Delta\sum_{j=1}^n c_j\delta(x-x_j), \tag{13}$$

where the array of numbers $c_j \equiv \langle x_j|\Psi_0\rangle$ represent the state Ψ_0 on a grid of equally spaced coordinates $x_j = x_{min} + (j-1)\Delta$ with finite resolution $\Delta = (x_{max} - x_{min})/(n-1)$.

Note that the grid-based representation, introduced by Eq. (13), can be trivially generalized to a grid-based representation in the multidimensional space of parameters (*e.g.*, x_j, p_j, γ_j, ... etc.) when expanding the target state $\Psi_0(x)$ as a linear combination of basis functions $\langle x|x_j, p_j, \gamma_j\rangle$, with expansion coefficients as $c_j \equiv \langle x_j, p_j, \gamma_j|\Psi_0\rangle$.

17

5.1 Computational Problem 1

Write a computer program to represent the wave-packet, introduced by Eq. (11) on a grid of equally spaced coordinates $x_j = x_{min} + (j-1)\Delta$ with finite resolution $\Delta = (x_{max} - x_{min})/(n-1)$ and visualize the output. Choose $x_0 = 0$ and $p_0 = 0$, in the range x=(-20,20), with $\alpha = \omega m$, where $m = 1$ and $\omega = 1$.

Next, we consider grid-based representations in momentum space:

$$\Psi_0(p) = \langle p|\Psi_0\rangle. \tag{14}$$

Inserting the closure relation $\hat{1} = \int dx|x\rangle\langle x|$ in Eq. (14), we obtain that

$$\langle p|\Psi_0\rangle = \int dx\langle p|x\rangle\langle x|\Psi_0\rangle = (2\pi)^{-1/2}\int dxe^{-ipx}\langle x|\Psi_0\rangle. \tag{15}$$

is the Fourier transform of the initial state. The second equality in Eq. (15) was obtained by using:

$$\langle x|p\rangle = (2\pi)^{-1/2}e^{ipx}, \tag{16}$$

which is the eigenstate of the momentum operator $\hat{p} = -i\nabla$, with eigenvalue p, since $\hat{p}\langle x|p\rangle = p\langle x|p\rangle$.

The Fourier transform can be computationally implemented in $O(N\log(N))$ steps by using the Fast Fourier Transform (FFT) algorithm [see, Ch. 12 of Numerical Recipes by W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, Cambridge University Press, Cambridge, 1986 (http://www.library.cornell.edu/nr/bookfpdf/f12-2.pdf)] when $\langle x|\Psi_0\rangle$ is represented on a grid with $N = 2^n$ points (where n is an integer). In contrast, the implementation of the Fourier transform by quadrature integration would require $O(N^2)$ steps.

5.2 Computational Problem 2

Write a computer program to represent the initial state, introduced by Eq. (11), in the momentum space by applying the FFT algorithm to the grid-based representation generated in Problem 1 and visualize the output. Represent the wave-packet amplitudes and phases in the range p=(-4,4) and compare your output with the corresponding values obtained from the analytic Fourier transform obtained by using:

$$\int dx \exp(-a_2 x^2 + a_1 x + a_0) = \sqrt{\pi/a_2}\exp(a_0 + a_1^2/(4a_2)).$$

Next, we consider the grid-based representation of operators (*e.g.*, $\hat{x}$, $\hat{p}$, $V(\hat{x})$, and $\hat{T} = \hat{p}^2/(2m)$) and learn how these operators act on states represented on grids in coordinate and momentum spaces. For simplicity, we assume that the potential is Harmonic:

$$V(\hat{x}) = \frac{1}{2}m\omega^2(\hat{x} - \bar{x})^2. \tag{17}$$

18

Consider first applying the potential energy operator to the initial state, as follows,

$$V(\hat{x})\Psi_0(x) = V(x)\Psi_0(x) \equiv \tilde{\Psi}_0(x). \tag{18}$$

Since $\tilde{\Psi}_0(x)$ is just another function, Eq. (18) indicates that $V(\hat{x})$ can be represented on the same grid of coordinates as before (*i.e.*, equally spaced coordinates $x_j = x_{min} + (j-1)\Delta$, with finite resolution $\Delta = (x_{max} - x_{min})/(n-1)$). Since for each x_j, $\tilde{\Psi}_0(x_j) = V(x_j)\Psi(x_j)$, the operator $V(\hat{x})$ can be represented just as an array of numbers $V(x_j)$ associated with the grid-points x_j, and its operation on a state is represented on such a grid as a simple multiplication.

5.3 Computational Problem 3

Write a computer program to compute the expectation values of the position $x(0) = \langle\Psi_0|\hat{x}|\Psi_0\rangle$ and the potential energy $V = \langle\Psi_0|V(\hat{x})|\Psi_0\rangle$, where $V(x)$ is defined according to Eq. (17) for the initial wave-packet, introduced by Eq. (11), with various possible values of x_0 and p_0, with $\alpha = \omega m$, where $m = 1$ and $\omega = 1$.

Now consider applying the momentum operator, $\hat{p} = -i\nabla$, to the initial state $\Psi_0(x)$ as follows,

$$G(x) = \langle x|\hat{p}|\Psi_0\rangle = -i\nabla\Psi_0(x). \tag{19}$$

One simple way of implementing this operation, when $\Psi_0(x)$ is represented on a grid of equally spaced points $x_j = x_{min} + (j-1)\Delta$, is by computing finite-increment derivatives as follows:

$$G(x_j) = -i\frac{\Psi_0(x_{j+1}) - \Psi_0(x_{j-1})}{2\Delta}. \tag{20}$$

However, for a more general operator (*e.g.*, $\hat{T} = \hat{p}^2/(2m)$) this finite increment derivative procedure becomes complicated. In order to avoid computing finite-increment derivatives, one can implement an alternative procedure: represent the initial state in momentum-space (by Fourier transform of the initial state); apply the operator by simple multiplication in momentum space, then transform the resulting product back to the coordinate representation (by inverse-Fourier transform). This method can be derived by inserting the closure relation $\hat{1} = \int dp|p\rangle\langle p|$, in Eq. (19),

$$G(x) = \langle x|\hat{p}|\Psi_0\rangle = \int dp\langle x|\hat{p}|p\rangle\langle p|\Psi_0\rangle = (2\pi)^{-1/2}\int dpe^{ipx}p\langle p|\Psi_0\rangle, \tag{21}$$

since $\langle p|\Psi_0\rangle$ is defined, according to Eq. (15), as the Fourier transform of the initial state. Note that the second equality of Eq. (21) is obtained by introducing the substitution

$$\langle x|p\rangle = (2\pi)^{-1/2}e^{ix\hat{p}}. \tag{22}$$

While Eq. (21) illustrates the method for the specific operator $\hat{p}$, one immediately sees that any operator which is a function of $\hat{p}$ (*e.g.*, $\hat{T} = \hat{p}^2/(2m)$) can be analogously applied according to the Fourier transform procedure.

5.4 Computational Problem 4

Write a computer program to compute the expectation values of the initial momentum $p(0) = \langle\Psi_0|\hat{p}|\Psi_0\rangle$ and the kinetic energy $T = \langle\Psi_0|\hat{p}^2/(2m)|\Psi_0\rangle$ by using the Fourier transform procedure, where Ψ_0 is the initial wave-packet introduced by Eq. (11), with $x_0 = 0$, $p_0 = 0$, and $\alpha = \omega m$, where $m = 1$ and $\omega = 1$. Compute the expectation value of the energy $E = \langle\Psi_0|\hat{H}|\Psi_0\rangle$, where $\hat{H} = \hat{p}^2/(2m) + V(\hat{x})$, with $V(x)$ defined according to Eq. (17) and compare your result with the zero-point energy $E_0 = \omega/2$.

6   SOFT Method

The Split-Operator Fourier Transform (SOFT) method is a numerical approach for solving the time-dependent Schrödinger equation by using grid-based representations of the time-evolving states and operators. It relies on the previously introduced Fourier transform procedure to apply operators that are functions of $\hat{p}$ by simple multiplication of array elements. As an example, we will illustrate the SOFT algorithm as applied to the propagation of the harmonic oscillator, which can also be described analytically as follows:

$$\Psi_t(x) = \int dx' \langle x|e^{-i\hat{H}t}|x'\rangle\langle x'|\Psi_0\rangle, \tag{23}$$

where the Kernel $\langle x|e^{-i\hat{H}t}|x'\rangle$ is the quantum propagator

$$\langle x|e^{-i\hat{H}t}|x'\rangle = \sqrt{\frac{m\omega}{2\pi\sinh(it\omega)}}\exp\left(-\frac{m\omega}{2\sinh(\omega it)}[(x^2 + x'^2)\cosh(\omega it) - 2xx']\right). \tag{24}$$

The essence of the method is to discretize the propagation time on a grid $t_k = (k-1)\tau$, with $k = 1, ..., n$ and time-resolution $\tau = t/(n-1)$, and obtain the wave-packet at the intermediate times t_k by recursively applying Eq. (23) as follows,

$$\Psi_{t_{k+1}}(x) = \int dx' \langle x|e^{-i\hat{H}\tau}|x'\rangle\langle x'|\Psi_{t_k}\rangle. \tag{25}$$

If τ is a sufficiently small time-increment (*i.e.*, n is large), the time-evolution operator can be approximated according to the Trotter expansion to second order accuracy,

$$e^{-i\hat{H}\tau} = e^{-iV(\hat{x})\tau/2}e^{-i\hat{p}^2\tau/(2m)}e^{-iV(\hat{x})\tau/2} + O(\tau^3), \tag{26}$$

which separates the propagator into a product of three operators, each of them depending either on $\hat{x}$, or $\hat{p}$.

6.1   Computational Problem 5

Expand the exponential operators in both sides of Eq. (26) and show that the Trotter expansion is accurate to second order in powers of τ.

Substituting Eq. (26) into Eq. (25) and inserting the closure relation $\hat{1} = \int dp |p\rangle\langle p|$ gives,

$$\Psi_{t_{k+1}}(x) = \int dp \int dx' e^{-iV(\hat{x})\tau/2}\langle x|p\rangle e^{-ip^2\tau/(2m)}\langle p|x'\rangle e^{-iV(x')\tau/2}\Psi_{t_k}(x'). \quad (27)$$

By substituting $\langle p|x'\rangle$ and $\langle x|p\rangle$ according to Eqs. (16) and (22), respectively, we obtain:

$$\Psi_{t_{k+1}}(x) = e^{-iV(\hat{x})\tau/2}\frac{1}{\sqrt{2\pi}}\int dp e^{ixp}e^{-ip^2\tau/(2m)}\frac{1}{\sqrt{2\pi}}\int dx' e^{-ipx'}e^{-iV(x')\tau/2}\Psi_{t_k}(x'). \quad (28)$$

According to Eq. (28), then, the computational task necessary to propagate $\Psi_t(x)$ for a time-increment τ involves the following steps:

1. Represent $\Psi_{t_k}(x')$ and $e^{-iV(x')\tau/2}$ as arrays of numbers $\Psi_{t_k}(x_j)$ and $e^{-iV(x_j)\tau/2}$ associated with a grid of equally spaced coordinates $x_j = x_{min} + (j-1)\Delta$, with finite resolution $\Delta = (x_{max} - x_{min})/(n-1)$.

2. Apply the potential energy part of the Trotter expansion $e^{-iV(x')\tau/2}$ to $\Psi_{t_k}(x')$ by simple multiplication of array elements:

$$\tilde{\Psi}_{t_k}(x_j) = e^{-iV(x_j)\tau/2}\Psi_{t_k}(x_j).$$

3. Fourier transform $\tilde{\Psi}_{t_k}(x_j)$ to obtain $\tilde{\Psi}_{t_k}(p_j)$, and represent the kinetic energy part of the Trotter expansion $e^{-ip^2\tau/(2m)}$ as an array of numbers $e^{-ip_j^2\tau/(2m)}$ associated with a grid of equally spaced momenta $p_j = j/(x_{max} - x_{min})$.

4. Apply the kinetic energy part of the Trotter expansion $e^{-ip^2\tau/(2m)}$ to the Fourier transform $\tilde{\Psi}_{t_k}(p)$ by simple multiplication of array elements:

$$\widetilde{\Psi}_{t_k}(p_j) = e^{-ip_j^2\tau/(2m)}\tilde{\Psi}_{t_k}(p_j).$$

5. Inverse Fourier transform $\widetilde{\Psi}_{t_k}(p_j)$ to obtain $\widetilde{\Psi}_{t_k}(x_j)$ on the grid of equally spaced coordinates x_j.

6. Apply the potential energy part of the Trotter expansion $e^{-iV(x')\tau/2}$ to $\widetilde{\Psi}_{t_k}(x')$ by simple multiplication of array elements,

$$\Psi_{t_{k+1}}(x_j) = e^{-iV(x_j)\tau/2}\widetilde{\Psi}_{t_k}(x_j).$$

6.2 Computational Problem 6

Write a computer program that propagates the initial state $\Psi_0(x)$ for a single time increment ($\tau = 0.1$ a.u.). Use $x_0 = -2.5$, $p_0 = 0$, and $\alpha = \omega m$, where $m = 1$ and $\omega = 1$. Implement the SOFT method for the Hamiltonian $\hat{H} = \hat{p}^2/(2m) + V(\hat{x})$, where $V(x)$ is defined according to Eq. (17). Compare the resulting propagated state with the analytic solution obtained by substituting Eq. (24) into Eq. (23).

6.3 Computational Problem 7

Loop the computer program developed in Problem 5 with $x_0 = -2.5$ and $p_0 = 0$ for 100 steps with $\tau = 0.1$ a.u. For each step compute the expectation values of coordinates $x(t)$ and momenta $p(t)$ as done in Problems 3 and 4, respectively. Compare your calculations with the analytic solutions obtained by substituting Eq. (24) into Eq. (23). Verify that these correspond to the classical trajectories $x(t) = \bar{x} + (x_0 - \bar{x})\cos(\omega t)$ and $p(t) = p_0 - (x_0 - \bar{x})\omega m \sin(\omega t)$, which can be computed according to the Velocity-Verlet algorithm:

$$
\begin{aligned}
p_{j+1} &= p_j + (F(x_j) + F(x_{j+1}))\tau/2 \\
x_{j+1} &= x_j + p_j\tau/m + F(x_j)\tau^2/(2m).
\end{aligned}
\tag{29}
$$

6.4 Computational Problem 8

Change the potential to that of a Morse oscillator $V(\hat{x}) = De(1 - \exp(-a(\hat{x} - x_e)))^2$, with $x_e = 0$, $De = 8$, and $a = \sqrt{k/(2D_e)}$, where $k = m\omega^2$. Recompute the wave-packet propagation with $x_0 = -0.5$ and $p_0 = 0$ for 100 steps with $\tau = 0.1$ a.u., and compare the expectation values $x(t)$ and $p(t)$ with the corresponding classical trajectories obtained by recursively applying the Velocity-Verlet algorithm.

6.5 Computational Problem 9

Simulate the propagation of a wave-packet with $x_0 = -5.5$ and initial momentum $p_0 = 2$ colliding with a barrier potential $V(x) = 3$, if abs$(x) < 0.5$, and $V(x) = 0$, otherwise. Hint: In order to avoid artificial recurrences you might need to add an absorbing imaginary potential $V_a(x) = i(\text{abs}(x) - 10)^4$, if abs$(x) > 10$, and $V_a(x) = 0$, otherwise.

7 SOFT Propagation on Multiple Surfaces

The goal of this section is to generalize the implementation of the SOFT method to the description of quantum dynamics on multiple coupled potential energy surfaces.

To keep the presentation as simple as possible, we consider a molecule with two-coupled electronic states described by the Hamiltonian,

$$
\hat{H} = \hat{p}^2/(2m) + \hat{V},
\tag{30}
$$

where $\hat{V} = \hat{V}_0 + \hat{V}_c$, with $\hat{V}_0 = V_1(\hat{x})|1\rangle\langle 1| + V_2(\hat{x})|2\rangle\langle 2|$ and $\hat{V}_c = V_c(\hat{x})|1\rangle\langle 2| + V_c(\hat{x})|2\rangle\langle 1|$.

The computational task ahead is to implement the SOFT method to compute the time-dependent wave-packet

$$
|\Psi(\mathbf{x}; t)\rangle = \varphi_1(\mathbf{x}; t)|1\rangle + \varphi_2(\mathbf{x}; t)|2\rangle,
\tag{31}
$$

given the initial conditions $\varphi_1(\mathbf{x}; 0)$ and $\varphi_2(\mathbf{x}; 0)$, where $\varphi_1(\mathbf{x}; t)$ and $\varphi_2(\mathbf{x}; t)$ are the time-dependent nuclear wave-packet components associated with the electronic states $|1\rangle$ and $|2\rangle$, respectively.

Note that here the main challenges are that $\hat{V}_0$ and $\hat{V}_c$ do not commute, $|\Psi(\mathbf{x};t)\rangle$ involves two wave-packet components and $\hat{H}$ is a 2×2 matrix in the basis of $|1\rangle$ and $|2\rangle$.

A simple approach for propagating $\varphi_1(\mathbf{x};t)$ and $\varphi_2(\mathbf{x};t)$ involves the embedded form of the Trotter expansion,

$$e^{-i\hat{H}2\tau} \approx e^{-i\frac{\hat{p}^2}{2m}\tau}e^{-iV(\hat{x})2\tau}e^{-i\frac{\hat{p}^2}{2m}\tau} \approx e^{-i\frac{\hat{p}^2}{2m}\tau}e^{-iV_0(\hat{x})\tau}e^{-iV_c(\hat{x})2\tau}e^{-iV_0(\hat{x})\tau}e^{-i\frac{\hat{p}^2}{2m}\tau}, \qquad (32)$$

which can be implemented in the basis of $|1\rangle$ and $|2\rangle$ according to the following steps:

- Step [I]. Apply the kinetic energy part of the Trotter expansion to both wave-packet components $\varphi_1(\mathbf{x};t)$ and $\varphi_2(\mathbf{x};t)$ for time τ, as follows,

$$\begin{pmatrix} \varphi_1'(\mathbf{x};t+\tau) \\ \varphi_2'(\mathbf{x};t+\tau) \end{pmatrix} = \begin{pmatrix} e^{-i\frac{\hat{p}^2}{2m}\tau} & 0 \\ 0 & e^{-i\frac{\hat{p}^2}{2m}\tau} \end{pmatrix} \begin{pmatrix} \varphi_1(\mathbf{x};t) \\ \varphi_2(\mathbf{x};t) \end{pmatrix}. \qquad (33)$$

- Step [II]. Mix the two wave-packet components $\varphi_1'(\mathbf{x};t+\tau)$ and $\varphi_2'(\mathbf{x};t+\tau)$,

$$\begin{pmatrix} \varphi_1''(\mathbf{x};t+\tau) \\ \varphi_2''(\mathbf{x};t+\tau) \end{pmatrix} = \mathbf{M} \begin{pmatrix} \varphi_1'(\mathbf{x};t+\tau) \\ \varphi_2'(\mathbf{x};t+\tau) \end{pmatrix}, \qquad (34)$$

with

$$\mathbf{M} \equiv \mathbf{L}^{-1} \begin{pmatrix} e^{-iE_1(x)\tau} & 0 \\ 0 & e^{-iE_2(x)\tau} \end{pmatrix} \mathbf{L}, \qquad (35)$$

where $E_1(x)$ and $E_2(x)$ are the eigenvalues of the potential energy matrix $V = V_0 + V_c$ and $\mathbf{L}$ the matrix of column eigenvectors in the basis of diabatic states $|1\rangle$ and $|2\rangle$. Eigenvalues and eigenvectors of a symmetric matrix can be obtained by using the subroutines TRED2, TQLI and EIGSRT, as described in Numerical Recipes (Ch. 11, Numerical Recipes).

While this is a general procedure, the specific case of interest involves a 2×2 Hermitian matrix V, for which the matrix $\mathbf{M}$ can be found analytically,

$$\mathbf{M} \equiv \begin{pmatrix} e^{-i\hat{V}_1(\hat{x})2\tau}\cos(2V_c(\hat{x})\tau) & -i\sin(2V_c(\hat{x})\tau)\,e^{-i(\hat{V}_1(\hat{x})+\hat{V}_2(\hat{x}))\tau} \\ -i\sin(2V_c(\hat{x})\tau)\,e^{-i(\hat{V}_1(\hat{x})+\hat{V}_2(\hat{x}))\tau} & \cos(2V_c(\hat{x})\tau)\,e^{-i\hat{V}_2(\hat{x})2\tau} \end{pmatrix}. \qquad (36)$$

- Step [III]. Propagate $\varphi_1''(\mathbf{x};t+\tau)$ and $\varphi_2''(\mathbf{x};t+\tau)$ for time τ, according to the free-particle propagator, by applying the kinetic energy part of the Trotter expansion:

$$\begin{pmatrix} \varphi_1(\mathbf{x};t+2\tau) \\ \varphi_2(\mathbf{x};t+2\tau) \end{pmatrix} = \begin{pmatrix} e^{-i\frac{\hat{p}^2}{2m}\tau} & 0 \\ 0 & e^{-i\frac{\hat{p}^2}{2m}\tau} \end{pmatrix} \begin{pmatrix} \varphi_1''(\mathbf{x};t+\tau) \\ \varphi_2''(\mathbf{x};t+\tau) \end{pmatrix}. \qquad (37)$$

In practice, however, step [III] is combined with step [I] of the next propagation time-slice for all but the last propagation time-increment.

23

7.1 Problem 10

(a) Derive Eq. (36) by considering that,

$$e^{-i\mathbf{V}_c 2\tau} = \mathbf{D}^\dagger \begin{pmatrix} e^{iV_c(\mathbf{x})2\tau} & 0 \\ 0 & e^{-iV_c(\mathbf{x})2\tau} \end{pmatrix} \mathbf{D}, \tag{38}$$

with

$$\mathbf{D} = \mathbf{D}^\dagger \equiv \begin{pmatrix} -1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}, \tag{39}$$

since

$$e^{-i\mathbf{V}_c 2\tau} = \mathbf{1} + (-i\mathbf{V}_c 2\tau) + \frac{1}{2!}(-i\mathbf{V}_c 2\tau)^2 + ..., \tag{40}$$

and

$$\mathbf{V}_c \equiv \begin{pmatrix} 0 & V_c(\mathbf{x}) \\ V_c(\mathbf{x}) & 0 \end{pmatrix} = \mathbf{D}^\dagger \begin{pmatrix} -V_c(\mathbf{x}) & 0 \\ 0 & V_c(\mathbf{x}) \end{pmatrix} \mathbf{D}, \tag{41}$$

with $\mathbf{D}\mathbf{D}^\dagger = 1$.

7.2 Problem 11

Derive Eq. (35) by writing the matrix V in the basis of adiabatic eigenstates

$$\begin{aligned} \phi_1(x) &= L_{11}(x)|1\rangle + L_{21}(x)|2\rangle, \\ \phi_2(x) &= L_{12}(x)|1\rangle + L_{22}(x)|2\rangle, \end{aligned} \tag{42}$$

with eigenvalues $E_1(x)$ and $E_2(x)$, respectively. Then, using the expansion

$$e^{-i\mathbf{V}2\tau} = \mathbf{1} + (-i\mathbf{V}2\tau) + \frac{1}{2!}(-i\mathbf{V}2\tau)^2 + ..., \tag{43}$$

show that in the adiabatic representation

$$e^{-i\mathbf{V}2\tau} = \begin{pmatrix} e^{-iE_1(x)2\tau} & 0 \\ 0 & e^{-iE_2(x)2\tau} \end{pmatrix}. \tag{44}$$

Finally, show that the diagonal matrix introduced by Eq. (44) can be rotated to the representation of diabatic states $|1\rangle$, $|2\rangle$ according to the similarity transformation

$$\mathbf{L}^{-1} \begin{pmatrix} e^{-iE_1(x)\tau} & 0 \\ 0 & e^{-iE_2(x)\tau} \end{pmatrix} \mathbf{L}. \tag{45}$$

7.3 Computational Problem 12

(a) Write a computer program to implement the SOFT approach described in this section, where step [II] is numerically computed according to Eq. (35). Propagate $|\Psi(\mathbf{x};t)\rangle = \varphi_1(\mathbf{x};t)|1\rangle + \varphi_2(\mathbf{x};t)|2\rangle$, where $\varphi_1(\mathbf{x};0) = \varphi_1(\mathbf{x};0) = \Psi_0(x)$ and $\Psi_0(x)$ as defined in Eq. (11). Use $x_0 = -2.2$, $p_0 = 0$, $m = 1$, $\omega = 1$ and two coupled potential energy surfaces described by the potential energy matrix

$$V = \begin{pmatrix} V_1(x) & \delta \\ \delta & V_2(x) \end{pmatrix},\tag{46}$$

where $\delta = 0.3$, $V_1(x) = m\omega^2(x - \bar{x})^2/2$ and $V_2(x) = -x^2/2 + x^4/22$; (b) Propagate $\Psi(\mathbf{x};t)$ according to the potential energy matrix introduced by Eq. (50), with $\delta = 0$ and compare your results with those obtained in item (a).

7.4 Computational Problem ICN

Write a computer program to implement the SOFT approach, described in this section, as applied to the simulation of the ICN photodissociation as reported in JCP 112, 5566 (2000).

Solution in Sec. 61.27.

7.5 SOFT Surface Hopping

The goal of this section is to introduce a numerically exact 'surface hopping' stochastic algorithm for simulations of nonadiabatic quantum dynamics (called Split Operator Fourier Transform/Surface Hopping (SOFT/SH) throughout this section). If the coupling matrix elements are constant, $V_c = \delta$, the embedded form of the Trotter expansion introduced by Eq. (32) becomes:

$$\begin{pmatrix} \varphi_1(\mathbf{x};t+n\tau) \\ \varphi_2(\mathbf{x};t+n\tau) \end{pmatrix} = \begin{bmatrix} \cos(V_c\tau)\hat{U}_1 & -i\sin(V_c\tau)\hat{U}_A \\ -i\sin(V_c\tau)\hat{U}_A & \cos(V_c\tau)\hat{U}_2 \end{bmatrix}^n \begin{pmatrix} \varphi_1(\mathbf{x};t) \\ \varphi_2(\mathbf{x};t) \end{pmatrix},\tag{47}$$

where $\hat{U}_j = e^{-i\left(\frac{\hat{\mathbf{p}}^2}{2m} + V_j(\hat{x})\right)\tau}$ with $V_A(x) = \frac{V_1(x)+V_2(x)}{2}$ the average potential energy surface. Equation (47) indicates that the time-evolved wave-packet components $\varphi_j(\mathbf{x};t+\tau)$ result from the interference between the component $\varphi_j(\mathbf{x};t)$ propagated on the diabatic potential energy surface (e.g., V_j) and the other component propagated on the *average potential*, V_A. The relative weights associated with these two contributions to $\varphi_j(\mathbf{x};t+\tau)$ are given by the pre-exponential factors, $\cos(V_c\tau)$ and $\sin(V_c\tau)$, respectively.

Equation (47) is particularly suitable for the implementation of a stochastic integration algorithm that propagates an ensemble of realizations of the systems, sampled according to the nature of the initial states. Each realization $\varphi_j(\mathbf{x};t)$ will contribute to the time-evolved wavepacket component $\varphi_j(\mathbf{x};t+\tau)$ when propagated on diabatic surface V_j for time τ, with probability proportional to $\cos(V_c\tau)$. Otherwise, that realization is propagated on the average potential $V_A = (V_j+V_k)/2$ and contributes to the time-evolved wave-packet component $\varphi_k(\mathbf{x};t+\tau)$ with probability proportional

to $\sin(V_c\tau)$). Therefore, the partial contribution of each basis function(realization) to the overall ensemble average requires propagation for short-time increments followed by stochastic switches based on the strengths of the couplings between PESs. When the diabatic propagation is based on SOFT or MP/SOFT approaches, the resulting method is a numerically exact surface hopping algorithm. Approximate implementations can also be developed with propagation on diabatic(average) surfaces according to classical or semiclassical propagation schemes.

Another possible implementation of Eq. (47) is based on the hopping probability $P = \sin^2(V_c\tau)$, as follows:

$$\begin{pmatrix} \varphi_1(\mathbf{x}; t+n\tau) \\ \varphi_2(\mathbf{x}; t+n\tau) \end{pmatrix} = \begin{bmatrix} (1-P)w_+\hat{U}_1 & Pw_-\hat{U}_A \\ Pw_-\hat{U}_A & (1-P)w_+\hat{U}_2 \end{bmatrix}^n \begin{pmatrix} \varphi_1(\mathbf{x}; t) \\ \varphi_2(\mathbf{x}; t) \end{pmatrix}, \tag{48}$$

with $w_+ = 2/(e^{iV_c\tau} + e^{-iV_c\tau})$ and $w_- = 2/(e^{iV_c\tau} - e^{-iV_c\tau})$. Equation (48) can be evaluated as follows:

$$\begin{aligned} \varphi_1(\mathbf{x}; t+n\tau) &\approx (1-P)^n \omega_+^n \hat{U}_1^n \varphi_1(\mathbf{x}; t) \\ &+ P^2(1-P)^{n-2}\omega_+^{n-2}\omega_-^2 \sum_{n_1=0}^{n-2}\sum_{n_2=0}^{n-2-n_1} \hat{U}_1^{n-n_1-n_2-2}\hat{U}_A\hat{U}_2^{n_2}\hat{U}_A\hat{U}_1^{n_1}\varphi_1(\mathbf{x}; t) + \cdots \\ &+ P(1-P)^{n-1}\omega_+^{n-1}\omega_- \sum_{n_1=0}^{n-1} \hat{U}_1^{n-n_1-1}\hat{U}_A\hat{U}_2^{n_1}\varphi_2(\mathbf{x}; t) + \cdots, \end{aligned} \tag{49}$$

where the first two rows of Eq. (49) include the flux that started and finished on surface V_1, including diabatic propagation on surface V_1 (first row), or interrupted by transient propagation on V_A and V_2 (second row). The third row involves the non-diabatic flux that started on surface V_2 and transferred to surface V_1.

7.6 Computational Problem 12SH

(a) Write a computer program to implement the SOFT Surface Hopping approach, keeping only the terms with 1 or 2 hops, as explictily written in Eq. (49). Compare your results to those obtain in Problem 12 through propagation of $|\Psi(\mathbf{x}; t)\rangle = \varphi_1(\mathbf{x}; t)|1\rangle + \varphi_2(\mathbf{x}; t)|2\rangle$, where $\varphi_1(\mathbf{x}; 0) = \varphi_1(\mathbf{x}; 0) = \Psi_0(x)$ and $\Psi_0(x)$ as defined in Eq. (11). Use $x_0 = -2.2$, $p_0 = 0$, $m = 1$, $\omega = 1$ and two coupled potential energy surfaces described by the potential energy matrix

$$V = \begin{pmatrix} V_1(x) & \delta \\ \delta & V_2(x) \end{pmatrix}, \tag{50}$$

where $\delta = 0.3$, $V_1(x) = m\omega^2(x-\bar{x})^2/2$ and $V_2(x) = -x^2/2 + x^4/22$; (b) Propagate $\Psi(\mathbf{x}; t)$ according to the potential energy matrix introduced by Eq. (50), with $\delta = 0$ and compare your results with those obtained in item (a).

8   Path Integrals: Thermal Correlation Functions

The goal of this section is to show how to compute thermal correlation functions $C(t)$ for systems where quantum mechanical effects are important. For comparisons, classical thermal correlation functions can be computed by propagating Hamilton's equations according to the Velocity-Verlet algorithm. Coordinates and momenta $q(t)$ and $p(t)$ are propagated for a sufficiently long trajectory and correlation functions are obtained as follows:

$$C(t) = \langle A(0)B(t) \rangle = \frac{1}{\tau} \int_0^\tau dt' A(q(t'), p(t')) B(q(t'+t), p(t'+t)), \tag{51}$$

where $A(0)$ and $B(t)$ represent the quantities of interest at time 0 and t, respectively. [1].

The quantum mechanical expression of $C(t)$ is,

$$C(t) = Tr[\hat{\rho}\hat{A}\hat{B}(t)], \tag{52}$$

where $\hat{\rho} = Z^{-1}exp(-\beta\hat{H})$ is the density operator and the operators $\hat{A}$ and $\hat{B}(t)$ are defined so that $A(0) = \langle \Psi_0 | \hat{A} | \Psi_0 \rangle$ is the expectation value of A at $t = 0$ and

$$B(t) = \langle \Psi_0 | \hat{B}(t) | \Psi_0 \rangle = \langle \Psi_0 | e^{(i/\hbar)\hat{H}t} \hat{B} e^{-(i/\hbar)\hat{H}t} | \Psi_0 \rangle, \tag{53}$$

is the expectation value of B at time t when the system is initially prepared in state $|\Psi_0\rangle$ and evolves according to the Hamiltonian,

$$\hat{H} = \hat{p}^2/(2m) + \hat{V}, \tag{54}$$

as follows: $|\Psi_t\rangle = e^{-(i/\hbar)\hat{H}t}|\Psi_0\rangle$. Note that $\hat{B}(t) = e^{(i/\hbar)\hat{H}t}\hat{B}e^{-(i/\hbar)\hat{H}t}$ is the Heisenberg operator associated with quantity B.

Thermal correlation functions can therefore be expressed as,

$$C(t) = Z^{-1}Tr[e^{-\beta\hat{H}} \hat{A} e^{(i/\hbar)\hat{H}t} \hat{B} e^{-(i/\hbar)\hat{H}t}], \tag{55}$$

an expression that can be re-written in coordinate representation as follows:

$$C(t) = Z^{-1} \int dx \int dx' \int dx'' \int dx''' \int dx'''' \langle x|e^{-\beta\hat{H}}|x'\rangle \langle x'|\hat{A}|x''\rangle \langle x''|e^{(i/\hbar)\hat{H}t}|x'''\rangle$$
$$\langle x'''|\hat{B}|x''''\rangle \langle x''''|e^{-(i/\hbar)\hat{H}t}|x\rangle. \tag{56}$$

Note that in order to compute $C(t)$ it is necessary to obtain expressions for the Boltzmann operator matrix elements $\langle x|e^{-\beta\hat{H}}|x'\rangle$ as well as for the forward and backward time-evolution operator matrix elements $\langle x|e^{-(i/\hbar)\hat{H}t}|x'\rangle$ and $\langle x|e^{(i/\hbar)\hat{H}t}|x'\rangle$, respectively.

In order to obtain an expression of the matrix elements of the Boltzmann operator, we express the exponential operator as a product of a large number n of exponential operators,

$$\langle x_0|e^{-\beta\hat{H}}|x_n\rangle = \langle x_0|e^{-\epsilon\hat{H}}e^{-\epsilon\hat{H}}...e^{-\epsilon\hat{H}}|x_n\rangle, \tag{57}$$

---

[1]Note that calculations of $C(t)$ provide a description of any equilibrium property, $\langle A \rangle$, when $\hat{B} = 1$, or dynamical ensemble average $\langle B(t) \rangle$, when $\hat{A} = 1$, respectively

where $\epsilon \equiv \beta/n << 1$. Inserting the closure relation in between exponential operators we obtain,

$$\langle x_0|e^{-\beta\hat{H}}|x_n\rangle = \int dx_1 ... \int dx_{n-1}\langle x_0|e^{-\epsilon\hat{H}}|x_1\rangle ... \langle x_{n-1}|e^{-\epsilon\hat{H}}|x_n\rangle. \tag{58}$$

The high-temperature Boltzmann operator $e^{-\epsilon\hat{H}}$ can be written in the form of the Trotter expansion,

$$e^{-\epsilon\hat{H}} \approx e^{-\epsilon\hat{V}/2}e^{-\epsilon\hat{p}^2/(2m)}e^{-\epsilon\hat{V}/2}, \tag{59}$$

to second order accuracy.

Therefore matrix elements of the Boltzmann operator at high-temperature can be obtained as follows:

$$\langle x_0|e^{-\epsilon\hat{H}}|x_1\rangle = \int dx \int dp \int dx' \int dp'\langle x_0|e^{-\epsilon\hat{V}/2}|x'\rangle\langle x'|p'\rangle\langle p'|e^{-\epsilon\hat{p}^2/(2m)}|p\rangle\langle p|x\rangle\langle x|e^{-\epsilon\hat{V}/2}|x_1\rangle, \tag{60}$$

where

$$\langle x|p\rangle = \frac{1}{\sqrt{2\pi\hbar}}e^{\frac{i}{\hbar}xp}, \tag{61}$$

since

$$-i\hbar\frac{\partial}{\partial x}\langle x|p\rangle = p\langle x|p\rangle. \tag{62}$$

Furthermore,

$$\langle x|e^{-\epsilon\hat{V}/2}|x'\rangle = e^{-\epsilon V(x)/2}\delta(x - x'). \tag{63}$$

Therefore,

$$\langle x_0|e^{-\epsilon\hat{H}}|x_1\rangle = \frac{1}{2\pi\hbar}\int dx \int dp \int dx' \int dp'e^{-\epsilon V(x')/2}\delta(x' - x_0)e^{\frac{i}{\hbar}x'p'}e^{-\epsilon p^2/(2m)}\delta(p - p')$$
$$e^{-\frac{i}{\hbar}xp}\delta(x - x_1)e^{-\epsilon V(x_1)/2}, \tag{64}$$

which gives,

$$\langle x_0|e^{-\epsilon\hat{H}}|x_1\rangle = \frac{1}{2\pi\hbar}e^{-\frac{\epsilon}{2}[V(x_0)+V(x_1)]}\int dp e^{-\epsilon p^2/(2m)+\frac{i}{\hbar}(x_0-x_1)p}, \tag{65}$$

or,

$$\langle x_0|e^{-\epsilon\hat{H}}|x_1\rangle = \frac{1}{2\pi\hbar}e^{-\frac{\epsilon}{2}(V(x_0)+V(x_1))}\sqrt{\frac{m}{2\pi\epsilon\hbar^2}}e^{-\frac{1}{2}m\left[\frac{(x_1-x_0)}{\hbar\epsilon}\right]^2\epsilon},$$
$$\approx \frac{1}{2\pi\hbar}e^{-\epsilon V(x_1)}\sqrt{\frac{m}{2\pi\epsilon\hbar^2}}e^{-\frac{1}{2}m\left[\frac{(x_1-x_0)}{\hbar\epsilon}\right]^2\epsilon}, \tag{66}$$

Matrix elements of the Boltzmann operator at finite-temperature can be obtained by substituting Eq. (66) into Eq. (58):

$$\langle x_0|e^{-\beta\hat{H}}|x_n\rangle = \int dx_1 ... \int dx_{n-1}\left(\frac{m}{2\pi\epsilon\hbar^2}\right)^{n/2}e^{-\epsilon\sum_{j=1}^{n}V(x_j)+\frac{1}{2}m\omega^2(x_j-x_{j-1})^2}, \tag{67}$$

28

where $\omega = 1/(\hbar\epsilon)$. Note that the r.h.s of Eq. (67) corresponds to the partition function of a chain of n-harmonic oscillators with cordinates x_j under the influence of an external potential $V(x_j)$. Each chain of harmonic oscillators describes a path from x_0 to x_n.

The multidimential integral, introduced by Eq. (67), can be computed by importance sampling Monte Carlo by sampling sets of coordinates $x_1, ..., x_{n-1}$ with sampling functions defined by the Gaussians associated with the linked harmonic oscillators. Such a computational approach for obtaining thermal equilibrium density matrices is called *Path Integral Monte Carlo*.

8.1 Ring Polymer Implementation

An alternative expression for the matrix elements of the Boltzmann operator can be obtained by introducing into Eq. (67) the following substitution:

$$\left(\frac{m}{2\pi\epsilon\hbar^2}\right)^{1/2} = (2\pi\hbar)^{-1} \int dp_j e^{-\epsilon\frac{p_j^2}{2m}}, \tag{68}$$

as follows:

$$\langle x_0|e^{-\beta\hat{H}}|x_n\rangle = (2\pi\hbar)^{-n} \int dp_1... \int dp_n e^{-\epsilon\sum_{j=1}^{n}\frac{p_j^2}{2m}} \tag{69}$$
$$\times \int dx_1... \int dx_{n-1} e^{-\epsilon\sum_{j=1}^{n} V(x_j)+\frac{1}{2}m\omega^2(x_j-x_{j-1})^2},$$

to obtain

$$\langle x_0|e^{-\beta\hat{H}}|x_n\rangle = (2\pi\hbar)^{-n} \int dx_1... \int dx_{n-1} \int dp_1... \int dp_n\, e^{-\epsilon H_n(x_0;\mathbf{x}^n,\mathbf{p}^n)}, \tag{70}$$

where H_n is the n-bead polymer Hamiltonian

$$H_n(x_0; \mathbf{x}^n, \mathbf{p}^n) = \sum_{j=1}^{n} \frac{p_j^2}{2m} + V(x_j) + \frac{1}{2}m\omega^2(x_j - x_{j-1})^2, \tag{71}$$

and

$$Z = \int dx_0 \int dx_n \delta(x_0 - x_n)\langle x_0|e^{-\beta\hat{H}}|x_n\rangle$$
$$= (2\pi\hbar)^{-n} \int d\mathbf{x}^n \int d\mathbf{p}^n\, \delta(x_0 - x_n)e^{-\epsilon H_n(x_0;\mathbf{x}^n,\mathbf{p}^n)}. \tag{72}$$

8.2 Exercise 13

Compute $\langle x_0|e^{-\beta\hat{H}}|x_n\rangle$ for the Harmonic oscillator defined by the Hamiltonian

$$\hat{H} = \frac{\hat{p}^2}{2m} + \frac{1}{2}m\omega_0^2\hat{x}^2, \tag{73}$$

by using the Path Integral Monte Carlo method, with $n = 2, 4, 6, 8$ and 10 and show that for larger values of n the calculation converges to the analytic expression:

$$\langle x|e^{-\beta \hat{H}}|x'\rangle = \sqrt{\frac{m\omega}{2\pi\hbar \sinh(\beta\hbar\omega)}} exp\left(-\frac{m\omega}{2\hbar \sinh(\omega\beta\hbar)}\left[(x^2 + x'^2)cosh(\omega\beta) - 2xx'\right]\right), \quad (74)$$

which in the free particle limit ($\omega \to 0$) becomes

$$\langle x|e^{-\beta \hat{H}}|x'\rangle = \sqrt{\frac{m}{2\pi\beta\hbar^2}} exp\left(-\frac{m}{2\beta\hbar^2}\left[(x - x')^2\right]\right), \quad (75)$$

since $sinh(\beta\hbar\omega) \to \beta\hbar\omega$ and $cosh(\beta\hbar\omega) \to 1$.

Matrix elements of the time-evolution operator $e^{-\frac{i}{\hbar}\hat{H}t}$ can be obtained by following the same methodology implemented for the Boltzmann matrix $e^{-\beta\hat{H}\tau}$. We first introduce the variable substitution $\epsilon \equiv i\tau/\hbar$ in Eq. (66) and then we obtain the short-time propagator as follows:

$$\langle x|e^{-\frac{i}{\hbar}\hat{H}\tau}|x'\rangle = \sqrt{\frac{m}{2\pi\hbar i\tau}}e^{\frac{i}{\hbar}\left(\frac{1}{2}m\left[\frac{(x-x')}{\tau}\right]^2 - \frac{1}{2}[V(x)+V(x')]\right)\tau}. \quad (76)$$

Then, we concatenate the short-time propagators introduced by Eq. (76) and we obtain the finite-time propagator,

$$\langle x_0|e^{-\frac{i}{\hbar}\hat{H}t}|x_n\rangle = \int dx_1... \int dx_{n-1} \left(\frac{m}{2\pi\hbar i\tau}\right)^{n/2} e^{\frac{i}{\hbar}\left(\sum_{j=1}^{n} \frac{1}{2}m\left[\frac{(x_j-x_{j-1})}{\tau}\right]^2 - \frac{1}{2}[V(x_j)+V(x_{j-1})]\right)\tau}, \quad (77)$$

which in the limit when $\tau \to 0$ and $n \to \infty$ with $t = n\tau$ becomes,

$$\langle x_0|e^{-\frac{i}{\hbar}\hat{H}t}|x_n\rangle = \int \mathfrak{D}[x(t)]e^{\frac{i}{\hbar}S_c(t)}, \quad (78)$$

where $S_c(t)$ is the classical action associated with the arbitrary trajectory $x(t)$,

$$S_c(t') \equiv \int_0^{t'} dt \left[\frac{1}{2}m\left(\frac{\partial}{\partial t}x(t)\right)^2 - V(x(t))\right], \quad (79)$$

and $\mathfrak{D}[x(t)]$ is defined as follows,

$$\int \mathfrak{D}[x(t)]f(x(t)) \equiv \int dx_1... \int dx_{n-1} \left(\frac{m}{2\pi\hbar i\tau}\right)^{n/2} f(x(t)), \quad (80)$$

representing the integral over all paths $x(t)$ from x_0 to x_n, with intermediate coordinates $x_1, x_2, ..., x_{n-1}$ at times $\tau, 2\tau, ..., (n-1)\tau$, respectively.

9   SOFT Computations of Thermal Correlation Functions

The goal of this section is to introduce a generalization of the SOFT method for the description of thermal-equilibrium density matrices, finite-temperature time-dependent expectation values and time-correlation functions. Thermal correlation functions $C(t)$ can be obtained according to the following symmetrized form of Eq. (55):

$$C(t) = Z^{-1} \int d\mathbf{x} \int d\mathbf{x}' \int d\mathbf{x}'' \langle \mathbf{x}|e^{-\frac{\beta}{2}\hat{H}_0}|\mathbf{x}'\rangle A(\mathbf{x}')\langle \mathbf{x}'|e^{i\hat{H}_1 t}\hat{B}e^{-i\hat{H}_1 t}|\mathbf{x}''\rangle\langle \mathbf{x}''|e^{-\frac{\beta}{2}\hat{H}_0}|\mathbf{x}\rangle. \quad (81)$$

The computational task necessary to obtain $C(t)$, according to Eq. (81), requires obtaining the matrix elements $A(\mathbf{x}')\langle \mathbf{x}'|e^{-\frac{\beta}{2}\hat{H}_0}|\mathbf{x}\rangle$ and $\langle \mathbf{x}''|e^{-\frac{\beta}{2}\hat{H}_0}|\mathbf{x}\rangle$ and the subsequent real-time propagation for time t, according to $\hat{H}_1$. The matrix elements are computed, as described below by imaginary-time integration of the Bloch equation according to $\hat{H}_0$. The extension of the SOFT method, introduced in this section, involves the numerically exact treatment of both the real- and imaginary-time propagation steps as described below for the imaginary-time propagation. The real-time propagation is analogously performed by simply implementing the variable transformation $\beta \to -it$ from imaginary to real time.

The Boltzmann-operator matrix-elements are obtained by solving the Bloch equation,

$$\{\frac{\partial}{\partial \beta} - \frac{1}{2m}\nabla_{\mathbf{x}}^2 + V_0(\mathbf{x})\}\rho(\mathbf{x}, \mathbf{x}'; \beta) = 0, \quad (82)$$

for $\rho(\mathbf{x}, \mathbf{x}'; \beta) \equiv \langle \mathbf{x}|e^{-\beta\hat{H}_0}|\mathbf{x}'\rangle$ subject to the initial condition given by the high-temperature approximation,

$$\rho(\mathbf{x}, \mathbf{x}'; \epsilon) = \left(\frac{m}{2\pi\epsilon}\right)^{1/2} e^{-\frac{\epsilon}{2}[V_0(\mathbf{x})+V_0(\mathbf{x}')]}e^{-\frac{m}{2\epsilon}(\mathbf{x}-\mathbf{x}')^2}, \quad (83)$$

where ϵ defines a sufficiently high temperature $T = 1/(k_B\epsilon)$.

Equation (82) is formally integrated as follows,

$$\rho(\mathbf{x}, \mathbf{x}'; \beta) = \int d\mathbf{x}''\rho(\mathbf{x}, \mathbf{x}''; \beta - \epsilon)\rho(\mathbf{x}'', \mathbf{x}'; \epsilon), \quad (84)$$

where the propagator $\rho(\mathbf{x}, \mathbf{x}''; \beta - \epsilon) \equiv \langle \mathbf{x}|e^{-(\beta-\epsilon)\hat{H}_0}|\mathbf{x}''\rangle$ is imaginary-time sliced by repeatedly inserting the resolution of identity,

$$\hat{\mathbf{1}} = \int d\mathbf{x}_j|\mathbf{x}_j\rangle\langle \mathbf{x}_j|, \quad (85)$$

yielding,

$$\langle \mathbf{x}|e^{-(\beta-\epsilon)\hat{H}_0}|\mathbf{x}''\rangle = \int d\mathbf{x}_{s-1}... \int d\mathbf{x}_1\langle \mathbf{x}|e^{-i\hat{H}_0\tau}|\mathbf{x}_{s-1}\rangle...\langle \mathbf{x}_1|e^{-i\hat{H}_0\tau}|\mathbf{x}''\rangle, \quad (86)$$

where $\tau \equiv -i(\beta - \epsilon)/s$ is a sufficiently thin imaginary-time slice.

Each finite-time propagator, introduced by Eq. (86), is approximated for sufficiently small imaginary-time slices τ by the Trotter expansion to second-order accuracy,

$$e^{-i\hat{H}_0\tau} \approx e^{-i V_0(\hat{\mathbf{x}})\tau/2}e^{-i \frac{\hat{\mathbf{p}}^2}{2m}\tau} e^{-i V_0(\hat{\mathbf{x}})\tau/2}. \quad (87)$$

9.1 Computational Problem 14

Item (A): Generalize your program developed in Problem 6 to perform 1-dimensional wavepacket propagation of a state $\Psi(x, x'; t)$ that depends parametrically on x'.

Item (B): Make the variable substitution $\beta = it/\hbar$ and use your program to propagate the density matrix of a particle in a harmonic potential from a high-temperature T_i to a final temperature T_f.

Item (C): Compare the density of states $P(x; \beta) = Z^{-1}\rho(x, x; \beta)$, obtained in (B) at $\beta_i = 1/(k_B T_i)$ and $\beta_f = 1/(k_B T_f)$, to the corresponding analytic expressions given by Eq. (74) at T_i and T_f, respectively.

Item (D): Compare the density of states $P(x; \beta) = Z^{-1}\rho(x, x; \beta)$, obtained in (B) at $\beta_i = 1/(k_B T_i)$ and $\beta_f = 1/(k_B T_f)$, to the corresponding classical expression $P_c(x; \beta) = Z^{-1}exp(-\beta V(x))$.

Item (E): Repeat items (B)–(D) for the double-well potential and analyze the importance of quantum effects, such as tunneling, at high and low temperature.

It is important to note that a problem requiring $O(l)$ grid points for an accurate propagation of the state in 1-dimension, requires $O(l^N)$ points for the solution of a similar problem in N-dimensions. Therefore, the applicability of the grid-based SOFT method is limited to systems with very few degrees of freedom since both the storage and manipulation of multidimensional grids is prohibited for other than very small values of l and N. This problem, however, can be partially overcome by using compact coherent state representations as implemented in the MP/SOFT approach [Chen, X.; Wu, Y.; Batista V.S. J. Chem. Phys. 122, 64102 (2005)];[Wu, Y.; Batista V.S. J. Chem. Phys. 121, 1676 (2004)].

10 MP/SOFT Method

The Matching-Pursuit/Split Operator Fourier Transform (MP/SOFT) method is essentially the SOFT approach implemented in coherent-state representations, *i.e.*, where the grid-based representation of $\tilde{\rho}(\mathbf{x}_j, \mathbf{x'}_k; \epsilon)$ is substituted by coherent-state expansions generated according to the Matching-Pursuit algorithm.

The MP/SOFT propagation of the initial state $\rho(\mathbf{x}, \mathbf{x'}; \epsilon)$ entails the following steps:

- Step [1]: Decompose $\tilde{\rho}(\mathbf{x}, \mathbf{x'}; \epsilon) \equiv e^{-iV_0(\mathbf{x})\tau/2}\rho(\mathbf{x}, \mathbf{x'}; \epsilon)$ in a matching-pursuit coherent-state expansion:

$$\tilde{\rho}(\mathbf{x}, \mathbf{x'}; \epsilon) \approx \sum_{j=1}^{n} c_j \phi_j(\mathbf{x})[\phi'_j(\mathbf{x'})]^*, \tag{88}$$

where $\phi_j(\mathbf{x})$ and $\phi'_j(\mathbf{x})$ are N-dimensional coherent-states defined as follows,

$$\phi_j(\mathbf{x}) \equiv \prod_{k=1}^{N} A_{\phi_j}(k)e^{-\gamma_{\phi_j}(k)\left(x(k)-x_{\phi_j}(k)\right)^2/2} \, e^{i\, p_{\phi_j}(k)\left(x(k)-x_{\phi_j}(k)\right)}, \tag{89}$$

with complex-valued coordinates $x_{\phi_j}(k) \equiv r_{\phi_j}(k) + id_{\phi_j}(k)$, momenta $p_{\phi_j}(k) \equiv g_{\phi_j}(k) + if_{\phi_j}(k)$ and scaling parameters $\gamma_{\phi_j}(k) \equiv a_{\phi_j}(k) + ib_{\phi_j}(k)$. The normalization constants are $A_{\phi_j}(k) \equiv (a_{\phi_j}(k)/\pi)^{1/4} e^{-\frac{1}{2}a_{\phi_j}(k)d_{\phi_j}(k)^2 - d_{\phi_j}(k)g_{\phi_j}(k) - \left(b_{\phi_j}(k)d_{\phi_j}(k)+f_{\phi_j}(k)\right)^2/(2a_{\phi_j}(k))}$.

The expansion coefficients, introduced by Eq. (88), are defined as follows:

$$c_j \equiv \begin{cases} I_j, & \text{when } j = 1, \\ I_j - \sum_{k=1}^{j-1} c_k \langle \phi_j | \phi_k \rangle \langle \phi'_k | \phi'_j \rangle, & \text{for } j = 2 - n, \end{cases} \tag{90}$$

where the overlap integral I_j is defined as follows,

$$I_j \equiv \int d\mathbf{x}' d\mathbf{x} \, \phi_j(\mathbf{x}) \tilde{\rho}(\mathbf{x}, \mathbf{x}'; \epsilon) [\phi'_j(\mathbf{x}')]^*. \tag{91}$$

- Step [2]: Analytically Fourier transform the coherent-state expansion to the momentum representation, apply the kinetic energy part of the Trotter expansion and analytically inverse Fourier transform the resulting expression back to the coordinate representation to obtain the imaginary-time evolved Boltzmann-operator matrix elements:

$$\rho(\mathbf{x}, \mathbf{x}'; \epsilon + i\tau) = \sum_{j=1}^{n} c_j e^{-iV_0(\mathbf{x})\tau/2} \widetilde{\phi}_j(\mathbf{x}) [\phi'_j(\mathbf{x}')]^*, \tag{92}$$

where

$$\widetilde{\phi}_j(\mathbf{x}) \equiv \prod_{k=1}^{N} A_{\widetilde{\phi}_j}(k) \sqrt{\frac{m}{m + i\tau\gamma_{\widetilde{\phi}_j}(k)}} \exp\left(\frac{\left(\frac{p_{\widetilde{\phi}_j}(k)}{\gamma_{\widetilde{\phi}_j}(k)} - i(x_{\widetilde{\phi}_j}(k) - x(k)) \right)^2}{\left(\frac{2}{\gamma_{\widetilde{\phi}_j}(k)} + \frac{i2\tau}{m} \right)} - \frac{p_{\widetilde{\phi}_j}(k)^2}{2\gamma_{\widetilde{\phi}_j}(k)} \right). \tag{93}$$

Note that the MP/SOFT approach reduces the computational task necessary for the imaginary or real time propagation of the Boltzmann operator matrix elements $\rho(\mathbf{x}, \mathbf{x}'; \beta)$ to the problem of recursively generating the coherent-state expansions introduced by Eq. (88).

Coherent-state expansions are obtained by combining the matching pursuit algorithm and a gradient-based optimization method as follows:

- Step [1.1]. Evolve the complex-valued parameters, that define the initial trial coherent-states $\phi_j(\mathbf{x})$ and $\phi'_j(\mathbf{x})$, to locally maximize the overlap integral I_j, introduced in Eq. (91). The parameters $x_{\phi_1}(k), p_{\phi_1}(k), \gamma_{\phi_1}(k)$ and $x_{\phi'_1}(k), p_{\phi'_1}(k), \gamma_{\phi'_1}(k)$ of the corresponding local maximum define the first pair of coherent-states ϕ_1 and ϕ'_1 in the expansion introduced by Eq. (88) and the first expansion coefficient c_1, as follows: $\tilde{\rho}(\mathbf{x}, \mathbf{x}'; \epsilon) = c_1\phi_1(\mathbf{x})[\phi'_1(\mathbf{x}')]^* + \varepsilon_1(\mathbf{x}, \mathbf{x}')$, where $c_1 \equiv I_1$, as defined according to Eq. (91). Note that due to the definition of c_1, the residue $\varepsilon_1(\mathbf{x}, \mathbf{x}')$ does not overlap with the product state $\phi_1(\mathbf{x})[\phi'_1(\mathbf{x}')]^*$. Therefore, the norm of the remaining residue $\varepsilon_1(\mathbf{x}, \mathbf{x}')$ is smaller than the norm of the initial target state $\tilde{\rho}(\mathbf{x}, \mathbf{x}'; \epsilon)$ —i.e., $\| \varepsilon_1 \| < \| \tilde{\rho} \|$.

- Step [1.2]. Goto [1.1], replacing $\tilde{\rho}(\mathbf{x}, \mathbf{x}'; \epsilon)$ by $\varepsilon_1(\mathbf{x}, \mathbf{x}')$ —*i.e.*, sub-decompose the residue by its projection along the direction of its locally optimum match as follows: $\varepsilon_1(\mathbf{x}, \mathbf{x}') = c_2 \phi_2(\mathbf{x})[\phi_2'(\mathbf{x}')]^* + \varepsilon_2(\mathbf{x}, \mathbf{x}')$, where

$$c_2 \equiv \int d\mathbf{x}' d\mathbf{x} \; \phi_2(\mathbf{x}) \varepsilon_1(\mathbf{x}, \mathbf{x}')[\phi_2'(\mathbf{x}')]^*. \tag{94}$$

Note that $\| \varepsilon_2 \| < \| \varepsilon_1 \|$, since $\varepsilon_2(\mathbf{x}, \mathbf{x}')$ is orthogonal to the product state $\phi_2(\mathbf{x})[\phi_2'(\mathbf{x}')]^*$.

Step [1.2] is repeated each time on the resulting residue. After n successive projections, the norm of the residue ε_n is smaller than a desired precision ϵ —*i.e.*, $\| \varepsilon_n \| = (1 - \sum_{j=1}^n |c_j|^2)^{1/2} < \epsilon$, and the resulting expansion is given by Eq. (88). Note that norm conservation of $\hat{\tilde{\rho}}_\epsilon$ is maintained within a desired precision, just as in a linear orthogonal decomposition, although the coherent-states in the expansion are non-orthogonal basis-functions.

It is important to mention that the computational bottleneck of the MP/SOFT method involves the calculation of overlap matrix elements $\langle \phi_j \mid e^{-iV_j(\hat{\mathbf{x}})\tau/2} \mid \widetilde{\phi}_k \rangle$ and $\langle \phi_j \mid e^{-iV_j(\hat{\mathbf{x}})\tau/2} \mid \phi_k \rangle$, where $|\phi_k\rangle$ and $|\widetilde{\phi}_k\rangle$ are localized Gaussians introduced by Eqs. (89) and (93), respectively. The underlying computational task is however trivially parallelized.

The overlap integrals are most efficiently computed in applications to reaction surface Hamiltonians where a large number of harmonic modes can be *arbitrarily* coupled to a few reaction (tunneling) coordinates. For such systems, the Gaussian integrals over harmonic coordinates can be analytically computed and the remaining integrals over reaction coordinates are efficiently obtained according to numerical quadrature techniques. For more general Hamiltonians, the overlap matrix elements can be approximated by analytic Gaussian integrals when the choice of width parameters $\gamma_j(k)$ allows for a local expansion of $V_j(\hat{\mathbf{x}})$ to second order accuracy. Otherwise, the quadratic approximation is useful for numerically computing the corresponding full-dimensional integrals according to variance-reduction Monte Carlo techniques.

11   RP-MP/SOFT Method

The Ring Polymer-Matching Pursuit/Split Operator Fourier Transform (RP-MP/SOFT) method is essentially the MP/SOFT method for real time propagation of finite temperature densitiy matrix elements in coherent-state representations but with initial conditions obtained from Monte Carlo sampling of open polymer configurations.

Substituting Eq. (70) into Eq. (81), we obtain:

$$C(t) = \int dx_0 \int dx_n \; P(x_0, x_n) A(x_n) \langle x_n | e^{i\hat{H}_1 t} \hat{B} e^{-i\hat{H}_1 t} | x_0 \rangle, \tag{95}$$

where

$$P(x_0, x_n) = \frac{\int dx_1 ... \int dx_{n-1} \int d\mathbf{p}^n \; e^{-\epsilon H_n(x_0; \mathbf{x}^n, \mathbf{p}^n)}}{\int d\mathbf{x}^n \int d\mathbf{p}^n \; e^{-\epsilon H_n(x_n; \mathbf{x}^n, \mathbf{p}^n)}} \tag{96}$$

12   Orthogonalized Matching Pursuit Decomposition

Matching Pursuit is an algorithm originally developed for compressing representations of signals (*e.g.*, audio files, images, etc.) in terms of expansions of wavelet functions. Following the strategy characteristic of greedy algorithms, matching pursuit selects basis functions by making the choice that seems to be best at each decomposition stage and then addresses the decomposition of the residue at later stages. The method thus iteratively makes one greedy choice after another, reducing the initial problem (*i.e.*, the decomposition of a target state) into a smaller and smaller problems (*i.e.*, the decomposition of a target residues of smaller and smaller amplitude).

We adapt the matching pursuit algorithm for obtaining efficient representations of multidimensional wavepackets by using coherent states as a particular form of wavelets. The multidimensional coherent states of interest are defined as products of one-dimensional coherent states,

$$\alpha_j(x) = \left(\frac{\gamma_j}{\pi}\right)^{1/4} e^{-\gamma_j(x-u_j)^2/2 + ip_j(x-u_j)}, \tag{97}$$

where γ_j, u_j and p_j define the amplitude, position and frequency modulation parameters. The continuum spectrum of possible values for these parameters define overcomplete basis sets, providing enough flexibility to represent arbitrary states in Hilbert space.

Coherent state representations of multidimensional wavepackets are generated by successively finding at each decomposition stage the coherent state α_i that is the best approximation to the target function $\phi(x)$ (i.e., the coherent state with maximum overlap with the target state). This concept is quantitatively formulated according to the variational principle,

$$\delta \tilde{I}_1 = 0, \tag{98}$$

with

$$\tilde{I}_1(\alpha_1) = \|\varepsilon_1\|^2 = \int_{-\infty}^{\infty} dx \, (\phi(x) - c_1\alpha_1(x))(\phi(x) - c_1\alpha_1(x))^*. \tag{99}$$

Here, the target function ϕ is assumed to be expanded to first order with coherent state α_1 and its corresponding residue ε_1 as follows,

$$\phi(x) = c_1\alpha_1(x) + \varepsilon_1(x). \tag{100}$$

The coherent state α_1 giving the best approximation to the target function is the one that minimizes the norm of the residue, as defined by $|\varepsilon_1\rangle$. Since the functional $\tilde{I}_1$ is dependent on the parameters that define α_1, the minimization of the functional with respect to those coherent state parameters determines the optimal coherent state and its expansion coefficient $c_1 \equiv \langle\alpha_1|\phi\rangle$. Note that the residue $\varepsilon_1(x)$ is orthogonal to $\alpha_1(x)$ by virtue of the definition of c_1. Therefore, $\|\phi\| < \|\varepsilon_1\|$.

The next step involves the sub-decomposition of the residual vector $\varepsilon_1(x)$ by projecting it along the direction of its best match $\alpha_2(x)$ as follows:

$$\varepsilon_1(x) = c_2\alpha_2(x) + \varepsilon_2(x), \tag{101}$$

where $c_2 \equiv \langle\alpha_2|\varepsilon_1\rangle$. Note that, since $\varepsilon_2(x)$ is orthogonal to $\alpha_2(x)$, the norm of $\varepsilon_2(x)$ is smaller than the norm of $\varepsilon_1(x)$. This procedure is repeated each time on the resulting residue.

After n successive orthogonal projections, the norm of the residual vector $|\varepsilon_n\rangle$ is smaller than a desired precision ϵ, maintaining norm conservation within a desired precision,

$$\parallel \varepsilon_n \parallel = \sqrt{1 - \sum_{j=1}^{n} |c_j|^2} < \epsilon, \tag{102}$$

just as in a linear orthogonal decomposition. The resulting expansion is

$$\phi(x) \approx \sum_{j=1}^{n} c_j \alpha_j(x), \tag{103}$$

where the coefficients c_j are recursively defined as follows:

$$c_j = \langle j|\phi\rangle - \sum_{k=1}^{j-1} c_k \langle j|k\rangle. \tag{104}$$

The variational formula, introduced by Eq. (98), gives the most general definition of the optimum coherent state at a specific decomposition stage. At the same time, it also gives the numerical recipe for matching pursuit because $\mathrm{Min}(\tilde{I}(\alpha))$ is equivalent to $\mathrm{Max}(I(\alpha))$ with

$$I(\alpha) \equiv \parallel \int_{-\infty}^{\infty} dx\ \phi(x)\alpha(x)^* \parallel^2. \tag{105}$$

A proof of the above equivalence is shown in the Appendix.

The procedure for maximizing the square of the norm of the overlap integral, $c_j = \int_{-\infty}^{\infty} dx\ \phi(x)\alpha_j(x)^*$, is defined as follows:

1. Starting with the initial trial coherent, locally maximizing the norm of the overlap integral $\langle \alpha_1|\phi\rangle$ in the parameter space $\{\gamma_1, u_1, p_1\}$ of coherent states will generate the optimal parameters of the currently-pursued coherent state.

2. The parameters $\{\gamma_1, u_1, p_1\}$ of the local maximum define the first coherent state in the expansion, as well as the first expansion coefficient c_1, as follows: $|\phi\rangle = c_1|\alpha_1\rangle + |\epsilon_1\rangle$, where, $c_1 \equiv \langle \alpha_1|\phi\rangle$.

3. Go to Step. 1, replacing $|\phi\rangle$ with $|\epsilon_1\rangle$, sub-decompose the residue by its projection along the direction of its locally optimum match as follows: $|\epsilon_1| = c_2|\alpha_2\rangle + |\epsilon_2\rangle$ where, $c_2 \equiv \langle \alpha_2|\epsilon_1\rangle$.

The actual computational optimization of coherent state parameters can be performed according to a gradient-based optimization technique. A parallel implementation under the Message Passing Interface (MPI) environment can speed up the search for a satisfactory local minimum. The optimization starts with an initial trial coherent state $|\chi_j\rangle$ and optimizes the parameters $x_j(k)$, $p_j(k)$ and $\gamma_j(k)$ to maximize locally the overlap with the target state. Initial guess parameters $\gamma_j(k)$, $x_j(k)$ and $p_j(k)$ can be chosen as defined by the basis elements of the previous wave-packet representation (or initial state). An alternative to the optimization technique is the a fixed-point iterative procedure described below. With either technique, the main computational effort involves the evaluation of the overlap integrals, as described for specific applications in the next section and the following chapters.

13 MP/SOFT Propagation

In order to implement the SOFT approach in terms of coherent state representations generated according to the matching pursuit algorithm, we consider the propagation of state $\psi(x,t)$ for a short time increment τ as follows:

$$\psi(x, t+\tau) = e^{-i\tau\hat{H}}\psi(x,t), \tag{106}$$

and we substitute the time evolution operator by using the following Trotter expansion to second order accuracy:

$$e^{-i\tau\hat{H}} \approx e^{-i\frac{\tau}{2}\frac{\hat{p}^2}{2m}} e^{-i\tau\hat{V}(x)} e^{-i\frac{\tau}{2}\frac{\hat{p}^2}{2m}}. \tag{107}$$

Considering that $\psi(x,t)$ is given as a linear combination of coherent state,

$$\psi(x,t) = \sum_{j=1}^{n_i} c_j \alpha_j(x), \tag{108}$$

the underlying computational task necessary for selecting the first coherent state $\alpha_1^N(x)$ of the matching pursuit expansion,

$$\psi(x, t+\tau) = \sum_{j=1}^{n_f} c_j^N \alpha_j^N(x), \tag{109}$$

involves maximizing the norm of the overlap integral,

$$\begin{aligned} c_1^N(\alpha_1^N) &= \langle \alpha_1^N(x)|e^{-i\frac{\tau}{2}\hat{p}^2/2m} e^{-i\tau\hat{V}(x)} e^{-i\frac{\tau}{2}\hat{p}^2/2m}|(\sum_{j=1}^{n_i} c_j\alpha_j(x))\rangle \\ &= \sum_{j=1}^{n_i} c_j \langle \tilde{\alpha}_1^N(x)|e^{-i\tau V(x)}|\tilde{\alpha}_j(x)\rangle, \end{aligned} \tag{110}$$

with

$$|\tilde{\alpha}_j(x)\rangle = \left(\frac{\gamma_j}{\pi}\right)^{1/4} \sqrt{\frac{m}{m+i\tau\gamma_j/2}} \times \exp\left(\frac{\left(\frac{p_j}{\gamma_j} - i(x-u_j)\right)^2}{\left(\frac{2}{\gamma_j} + \frac{i\tau}{m}\right)} - \frac{p_j}{2\gamma_j}\right), \tag{111}$$

and

$$\langle\tilde{\alpha}_1^N(x)| = \left(\frac{\gamma_1^N}{\pi}\right)^{1/4} \sqrt{\frac{m}{m-i\tau\gamma_1^N/2}} \times \exp\left(\frac{\left(\frac{p_1^N}{\gamma_1^N} - i(x-u_1^N)\right)^2}{\left(\frac{2}{\gamma_k^N} - \frac{i\tau}{m}\right)} - \frac{p_1^N}{2\gamma_1^N}\right). \tag{112}$$

Once the maximum of the norm of the overlap integral is found by optimization of the parameters γ_1^N, p_1^N and u_1^N, the coherent state α_1^N defines the first term of the expansion with expansion coefficient c_1^N.

The first-order residue,

$$\epsilon_1(x) \equiv \psi(x, t + \tau) - c_1^N \alpha_1^N(x), \tag{113}$$

defines the target state for finding the second coherent state $\alpha_2^N(x)$ in the expansion by maximization of the norm of c_2^N) where,

$$c_k^N(\alpha_k^N) = \sum_{j=1}^{n_i} c_j \langle \tilde{\alpha}_k^N(x) | e^{-i\tau V(x)} | \tilde{\alpha}_j(x) \rangle - \sum_{l=1}^{n_k-1} c_l^N \langle \alpha_k^N(x) | \alpha_l{}^N(x) \rangle, \tag{114}$$

with $k = 1$. Analogously, subsequent coherent states α_k^N, with $k > 1$, are chosen by maximization of the norm of c_k^N with respect to the parameters that define the k-th coherent state α_k in the expansion.

Note that the implementation described herein is slightly different from the original implementation of the MP/SOFT method, since here we implicitly consider the two free-particle propagators of the Trotter expansion in the functional form of the coherent states $\tilde{\alpha}^N$. The resulting implementation thus merges the multiple elementary steps of the MP/SOFT algorithm into a single computational step, simplifying the method without any additional overhead.

The computational bottleneck is the calculation of matrix elements $\langle \tilde{\alpha}_k^N(x) | e^{-iV(x)\tau} | \tilde{\alpha}_j(x) \rangle$ where $|\tilde{\alpha}_j\rangle$ and $\langle \tilde{\alpha}_k^N |$ are localized Gaussians introduced by Eqs. 111 and 112, respectively. However, the underlying computational effort can be trivially parallelized, according to a portable Single-Program-Multiple-Data (SPMD) stream code that runs under the Message-Passing-Interface (MPI) environment, by distributing the n_i terms of the integrals $I_k(\alpha_k^N)$ among multiple processing elements. These overlap integrals are most efficiently computed in applications to reaction surface Hamiltonians where a large number of harmonic modes can be *arbitrarily* coupled to a few reaction (tunneling) coordinates. For such systems, the Gaussian integrals over harmonic coordinates can be analytically computed and the remaining integrals over reaction coordinates are efficiently obtained according to numerical quadrature techniques. For more general Hamiltonians, the overlap matrix elements can be approximated by analytic Gaussian integrals when the choice of width parameters $\gamma_j(k)$ allows for a local expansion of $V_j(\hat{x})$ to second order accuracy. Otherwise, the quadratic approximation is useful for numerically computing the corresponding full-dimensional integrals according to variance-reduction Monte Carlo techniques.

13.1 Variational Formulation

The goal of this section is to derive the MP/SOFT method from the variational principle equation $\delta \tilde{I} = 0$ with

$$\tilde{I}(\alpha_1) = \int_{-\infty}^{\infty} dx \, (\phi(x) - c_1 \alpha_1(x; u_1, p_1, \gamma_1))^\dagger (\phi(x) - c_1 \alpha_1(x; u_1, p_1, \gamma_1)), \tag{115}$$

defined as the norm of the residue between the target state ϕ and the selected basis function α_1. We show that minimizing the norm of the residue at a given decomposition stage is equivalent to maximizing the overlap overlap between the target state and the basis function. Finally, we

present the derivation of the nonlinear equations that defined the optimum parameters u_1, p_1, γ_1, as a formulation of the variational principle for coherent state expansions.

Writing Eq. 115 in terms of Paul Dirac's bra and ket notation, we obtain:

$$
\begin{aligned}
\tilde{I}(\alpha_1) &= (\langle \phi | - c_1^\dagger \langle \alpha_1 |)(| \phi \rangle - c_1 | \alpha_1 \rangle) \\
&= \langle \phi | \phi \rangle + c_1^\dagger c_1 \langle \alpha_1 | \alpha_1 \rangle) - c_1 \langle \phi | \alpha_1 \rangle - c_1^\dagger \langle \alpha_1 | \phi \rangle \\
&= 1 + c_1^\dagger c_1 - c_1 \langle \phi | \alpha_1 \rangle - c_1^\dagger \langle \alpha_1 | \phi \rangle.
\end{aligned}
$$

Essentially, $\tilde{I}$ is a multi-dimensional function of the parameters u_1, p_1, and γ_1 of the coherent state α_1, and the minimization of $\tilde{I}(\alpha_1)$ with respect to the coherent-state parameters is equivalent to solving the following set of non-linear equations:

$$
\frac{\partial \tilde{I}(\alpha_1)}{\partial c_1} = 0, \tag{116}
$$

$$
\frac{\partial \tilde{I}(\alpha_1)}{\partial u_1} = 0, \tag{117}
$$

$$
\frac{\partial \tilde{I}(\alpha_1)}{\partial p_1} = 0, \tag{118}
$$

$$
\frac{\partial \tilde{I}(\alpha_1)}{\partial \gamma_1} = 0. \tag{119}
$$

Minimization of the integrals $\tilde{I}(\alpha_i(x; u_i, p_i, \gamma_i))$, by differentiation with respect to the coherent state parameters u_i, p_i and γ_i that defined the trial coherent state α_i, leads to a set of nonlinear transcendental equations defining the optimum values of the coherent state parameters at the current decomposition stage, including the optimum position:

$$
u_i = Re\left[\frac{\langle \alpha_i | \hat{x} | \phi(x) \rangle}{\langle \alpha_i | \phi(x) \rangle} \right], \tag{120}
$$

frequency modulation:

$$
p_i = Re\left[\frac{\langle \alpha_i | \hat{p} | \phi(x) \rangle}{\langle \alpha_i | \phi(x) \rangle} \right], \tag{121}
$$

and amplitude:

$$
\gamma_i = 2Re\left[\frac{\langle \alpha_i | (\hat{x} - u_i)^2 | \phi(x) \rangle}{\langle \alpha_i | \phi(x) \rangle} \right]. \tag{122}
$$

13.2 MP/SOFT Expansion Coefficients

The expression of the expansion coefficients is obtained by minimizing $\tilde{I}$ with respect to c_1. Since c_1 is a complex-valued coefficient, Eq. 116 requires that,

$$
\frac{\partial \tilde{I}(\alpha_1)}{\partial c_{1r}} = 0; \quad \frac{\partial \tilde{I}(\alpha_1)}{\partial c_{1i}} = 0, \tag{123}
$$

where

c_{1r} and c_{1i} are the real and imaginary parts of the overlap $c_1 \equiv c_{1r} + i \, c_{1i} = \langle \alpha_1 | \phi \rangle$. After expanding the derivatives in Eq. 123, we obtain:

$$\frac{\partial \tilde{I}(\alpha_1)}{\partial c_{1r}} = 2c_{1r} - \langle \phi | \alpha_1 \rangle - \langle \alpha_1 | \phi \rangle, \tag{124}$$

and

$$\frac{\partial \tilde{I}(\alpha_1)}{\partial c_{1i}} = 2c_{1i} - i \, \langle \phi | \alpha_1 \rangle + i \, \langle \alpha_1 | \phi \rangle. \tag{125}$$

Simplifying the r.h.s's of the above two equations, the definition of c_1 is found:

$$c_1 = \langle \alpha_1 | \phi(x) \rangle \tag{126}$$

as a function of u_1, p_1, and γ_1.

Substituting Eq. (126) into Eq. (115), we find a new expression of $\tilde{I}(\alpha_1)$ depending on $\{ u_1, p_1, \gamma_1 \}$ can be found,

$$\tilde{I}(\alpha_1) = 1 - \|\langle \alpha_1 | \phi(x) \rangle\|^2. \tag{127}$$

This equation indicates that minimizing $\tilde{I}(\alpha_1)$ is equivalent to maximizing the square of the norm of the overlap integral. Therefore,

$$\mathrm{Min}[I(\tilde{\alpha}_1)] = \mathrm{Max}[c_1]. \tag{128}$$

13.3   Coherent State Parameters

The nonlinear equations of the parameters u_1, p_1, and γ_1 of the coherent state α_1 can be obtained from the variational equation $\tilde{I} = 0$, using the new formulae for the functional, $\tilde{I}(\alpha_1) = 1 - \|\langle \alpha_1 | \phi(x) \rangle\|^2$. Before starting with such a derivation, we need to define the coherent states in the position representation,

$$|\alpha_1(x)\rangle = (\frac{1}{\pi \gamma_1})^{1/4} \exp(-\frac{(x - u_1)^2}{2\gamma_1} + ip_1(x - u_1)), \tag{129}$$

and in the momentum representation,

$$|\alpha_1(p)\rangle = (\frac{\gamma_1}{\pi})^{1/4} \exp(-\frac{\gamma_1(p - p_1)^2}{2} - iu_1 p), \tag{130}$$

along with the following partial derivatives:

$$\frac{\partial|\alpha_1(x)\rangle}{\partial u_1} = (-ip_1 + \frac{(x-u)}{\gamma})\ |\alpha_1(x)\rangle, \tag{131}$$

$$\frac{\partial\langle\alpha_1(x)|}{\partial u_1} = (ip_1 + \frac{(x-u)}{\gamma})\ \langle\alpha_1(x)|, \tag{132}$$

$$\frac{\partial|\alpha_1(p)\rangle}{\partial p_1} = (-iu_1 + p_1\gamma_1 - \gamma_1 p)\ |\alpha_1(p)\rangle, \tag{133}$$

$$\frac{\partial\langle\alpha_1(p)|}{\partial p_1} = (iu_1 + p_1\gamma_1 - \gamma_1 p)\ \langle\alpha_1(p)|, \tag{134}$$

$$\frac{\partial|\alpha_1(x)\rangle}{\partial p_1} = (-\frac{1}{4\gamma} + \frac{(x-u_1)^2}{2\gamma^2})\ |\alpha_1(x)\rangle, \tag{135}$$

$$\frac{\partial\langle\alpha_1(x)|}{\partial p_1} = (-\frac{1}{4\gamma} + \frac{(x-u_1)^2}{2\gamma^2})\ \langle\alpha_1(x)|. \tag{136}$$

By substituting Eq. 127 into Eqs. 117, 118 and 119, we obtain the following equations:

$$\frac{\partial\langle\alpha_1(x)|\phi(x)\rangle}{\partial u_1}\langle\phi(x)|\alpha_1(x)\rangle + \langle\alpha_1(x)|\phi(x)\rangle\frac{\partial\langle\phi(x)|\alpha_1(x)\rangle}{\partial u_1} = 0, \tag{137}$$

$$\frac{\partial\langle\alpha_1(p)|\phi(p)\rangle}{\partial p_1}\langle\phi(p)|\alpha_1(p)\rangle + \langle\alpha_1(p)|\phi(p)\rangle\frac{\partial\langle\phi(p)|\alpha_1(p)\rangle}{\partial p_1} = 0, \tag{138}$$

$$\frac{\partial\langle\alpha_1(x)|\phi(x)\rangle}{\partial\gamma_1}\langle\phi(x)|\alpha_1(x)\rangle + \langle\alpha_1(x)|\phi(x)\rangle\frac{\partial\langle\phi(x)|\alpha_1(x)\rangle}{\partial\gamma_1} = 0. \tag{139}$$

Note that in order to derive the equation for p_1, the derivative of Eq. 118 is computed in the momentum representation. Substituting Eqs. 131 - 136 into the above three equations and collecting the corresponding terms, we obtain the three nonlinear equations of the three parameters, u_1, p_1, and γ_1. Let us take u_1 as an example. Using Eq. 131 and 132 into Eq. 137, we obtain

$$\begin{aligned}&\frac{\partial\langle\alpha_1(x)|\phi(x)\rangle}{\partial u_1}\langle\phi(x)|\alpha_1(x)\rangle + \langle\alpha_1(x)|\phi(x)\rangle\frac{\partial\langle\phi(x)|\alpha_1(x)\rangle}{\partial u_1}\\ &=\langle\alpha_1(x)|ip_1 + \frac{(x-u_1)}{\gamma}|\phi(x)\rangle\langle\phi(x)|\alpha_1(x)\rangle+\\ &\langle\alpha_1(x)|\phi(x)\rangle\langle\phi(x)| - ip_1 + \frac{(x-u_1)}{\gamma}|\alpha_1(x)\rangle\\ &=\langle\alpha_1(x)|\frac{(x-u_1)}{\gamma}|\phi(x)\rangle\langle\phi(x)|\alpha_1(x)\rangle + \langle\alpha_1(x)|\phi(x)\rangle\langle\phi(x)|\frac{(x-u_1)}{\gamma}|\alpha_1(x)\rangle = 0.\end{aligned} \tag{140}$$

Collecting terms for x and u and reorganizing the formulae, we obtain: have,

$$u_1 = \frac{1}{2}\frac{\langle\alpha_1(x)|x|\phi(x)\rangle}{\langle\alpha_1(x)|\phi(x)\rangle} + \frac{1}{2}\frac{\langle\phi(x)|x|\alpha_1(x)}{\langle\phi(x)|\alpha_1(x)}, \tag{141}$$

or

$$u_1 = Re\left[\frac{\langle\alpha_1(x)|x|\phi(x)\rangle}{\langle\alpha_1(x)|\phi(x)\rangle}\right]. \tag{142}$$

Following an analogous procedure we obtain:

$$p_1 = Re\left[\frac{\langle\alpha_1(p)|p|\phi(p)\rangle}{\langle\alpha_1(p)|\phi(p)\rangle}\right], \tag{143}$$

and

$$\gamma_1 = 2Re\left[\frac{\langle\alpha_1(x)|(x-u)^2|\phi(x)\rangle}{\langle\alpha_1(x)|\phi(x)\rangle}\right]. \tag{144}$$

Finally, Fourier transforming the coherent states in Eq. 143 to the position representation, we obtain: can have

$$p_1 = Re\left[\frac{\langle\alpha_1(x)|\hat{p}|\phi(x)\rangle}{\langle\alpha_1(x)|\phi(x)\rangle}\right]. \tag{145}$$

13.4   Orthogonalization

The goal of this section is to introduce a simple orthogonalization scheme to discount the contribution that new terms in the matching pursuit expansion introduce along the directions of previously selected terms.

We consider the matching pursuit decomposition of the target state $|\psi\rangle$,

$$|\psi\rangle = \sum_{j=1}^{n} c_j|j\rangle + |\epsilon_n\rangle, \tag{146}$$

with

$$c_j = \langle j|\psi\rangle - \sum_{k=1}^{j-1} c_k\langle j|k\rangle. \tag{147}$$

The resulting expansion can be formally rewritten as follows

$$|\psi\rangle = \sum_{j=1}^{n} c_j|j_o\rangle + \sum_{j=1}^{n} c_j \sum_{k=1}^{j-1} |k\rangle\langle k|j\rangle + |\epsilon_n\rangle, \tag{148}$$

where $|j_o\rangle$ is the component of $|j\rangle$ orthogonal to the previous terms. Therefore, according to Eq. (147), we obtain:

$$|\psi\rangle = \sum_{j=1}^{n}\left(c_j|j_o\rangle + \sum_{k=1}^{j-1}\left(\langle k|j\rangle\langle j|\psi\rangle - \sum_{k'=1}^{j-1} c_{k'}\langle k|j\rangle\langle j|k'\rangle\right)|k\rangle\right) + |\epsilon_n\rangle. \tag{149}$$

Equation 149 shows that the basis function $|j\rangle$ introduces not only a component $|j_o\rangle$ orthogonal to previously the selected terms (*i.e.*, first term in Eq. (149)) but also contributions along the previously

selected basis functions $|k\rangle$. The latter contributions effectively modify the expansion coefficients of the previous terms, as follows:

$$c_k \rightarrow c_k + \left(\langle j|\psi\rangle - \sum_{k'=1}^{j-1} c_{k'} \langle j|k'\rangle \right) \langle k|j\rangle,$$

$$\rightarrow c_k + c_j \langle k|j\rangle.$$

(150)

Therefore, a simple way of introducing an additional term that does not introduce any component along the direction of the previously selected basis functions is to decrement their coefficients in the amount introduced by the new term, as follows: $c_k = c_k - c_j \langle k|j\rangle$.

14 Semiclassical Thawed Gaussian (Gaussian Beam) Approach

This section describes the integration of the time-dependent Schrödinger equation,

$$G(x) = i\hbar\frac{\partial\psi}{\partial t} + \frac{\hbar^2}{2m}\frac{\partial^2\psi}{\partial x^2} - V(x,t)\psi = 0, \tag{151}$$

according to the *thawed Gaussian* or *Gaussian beam* approach, introduced by Heller [J. Chem. Phys. 62, 1544 (1975)] and others, including Popov 4 (1982), 85-97; Cerveny, Popov and Psencik [Geophys. J. R. Astr. Soc., 70 (1982), 109-128]; Ralston Studies in PDEs," MAA. Stud. Math., 23, Math. Assoc. America, Washington, DC, (1982), 206-248; Hill [Geophys., 55 (1990), 1416-1428], Coalson and Karplus [J. Chem. Phys. 93, 3919 (1990)].

We propose the Gaussian-beam ansatz

$$\psi(x,t) = e^{iS(t)/\hbar}\varphi(x,t), \tag{152}$$

with

$$\varphi(x,t) = \pi^{-1/4}\hbar^{-1/4}|Q|^{-1/2}e^{-\gamma(x-q)^2/(2\hbar)+\frac{i}{\hbar}p(x-q)}, \tag{153}$$

with $\gamma = PQ^{-1}$, which should make $G(x)$ vanish near q to some order (*e.g.*, second order).

The normalization constant $A = \pi^{-1/4}\hbar^{-1/4}|Q|^{-1/2}$, introduced by Eq. (153), is obtained as follows:

$$
\begin{aligned}
1 &= A^2\int dx e^{-\frac{\bar{P}}{\bar{Q}}(x-q)^2/(2\hbar)-\frac{i}{\hbar}p(x-q)}e^{-\frac{P}{Q}(x-q)^2/(2\hbar)+\frac{i}{\hbar}p(x-q)}, \\
&= A^2\int dx e^{-\frac{1}{2\hbar}\left(\frac{\bar{P}}{\bar{Q}}+\frac{P}{Q}\right)(x-q)^2}, \\
&= A^2\int dx e^{-\frac{1}{2\hbar}\left(\frac{\bar{P}Q+\bar{Q}P}{\bar{Q}Q}\right)(x-q)^2}, \\
&= A^2\int dx e^{-\frac{1}{\hbar\bar{Q}Q}(x-q)^2}, \\
&= A^2\sqrt{\pi\hbar|Q|^2}m
\end{aligned}
\tag{154}
$$

where the bars indicate conjugate values, and $\bar{P}Q + \bar{Q}P = 2$. Therefore, $A^2 = \pi^{-1/2}\hbar^{-1/2}|Q|$.

A Taylor expansion gives,

$$G(x) = G(q) + G'(q)(x-q) + \frac{1}{2}G''(q)(x-q)^2 + ... \tag{155}$$

and making $G(q) = G'(q) = G''(q) = 0$, we obtain a solution to third order accuracy (*i.e.*, $G = O(|x-q|^3)$).

Considering that

$$i\hbar\frac{\partial\psi}{\partial t} = \left(-\dot{S} - \frac{i\hbar}{2}\frac{\dot{Q}}{Q} - i\left(\dot{P}Q^{-1} - PQ^{-2}\dot{Q}\right)(x-q)^2/2 + iPQ^{-1}(x-q)\dot{q} - \dot{p}(x-q) + p\dot{q}\right)\psi, \tag{156}$$

and

$$\frac{\hbar^2}{2m}\frac{\partial^2\psi}{\partial x^2} = \frac{\hbar^2}{2m}\left([-PQ^{-1}(x-q)/\hbar + \frac{i}{\hbar}p]^2 - PQ^{-1}/\hbar\right)\psi, \tag{157}$$

we obtain

$$G(x) = \left(-\dot{S} - \frac{i\hbar}{2}\frac{\dot{Q}}{Q} - i\left(\dot{P}Q^{-1} - PQ^{-2}\dot{Q}\right)(x-q)^2/2 + iPQ^{-1}(x-q)\dot{q} - \dot{p}(x-q) + p\dot{q}\right.$$
$$\left. + \frac{\hbar^2}{2m}\left[-PQ^{-1}(x-q)/\hbar + \frac{i}{\hbar}p\right]^2 - \frac{\hbar}{2m}PQ^{-1} - V(x)\right)\psi, \tag{158}$$

with

$$G'(x) = G(x)\frac{\psi'}{\psi} + \left(-i\left(\dot{P}Q^{-1} - PQ^{-2}\dot{Q}\right)(x-q) + iPQ^{-1}\dot{q} - \dot{p}\right.$$
$$\left. - \frac{\hbar}{m}\left[-PQ^{-1}(x-q)/\hbar + \frac{i}{\hbar}p\right]PQ^{-1} - V'(x)\right)\psi. \tag{159}$$

and

$$G''(x) = G(x)\frac{\psi''}{\psi} + G'(x)\frac{\psi'}{\psi} - G(x)\frac{\psi'^2}{\psi^2} + \left[G'(x)\frac{1}{\psi} - G(x)\frac{\psi'}{\psi^2}\right]\psi'$$
$$+ \left(-i\left(\dot{P}Q^{-1} - PQ^{-2}\dot{Q}\right) + \frac{1}{m}\left[PQ^{-1}\right]^2 - V''(x)\right)\psi, \tag{160}$$

Therefore, making $G(q) = G'(q) = 0$, we obtain:

$$G'(q) = \left(iPQ^{-1}\dot{q} - \dot{p} - i\frac{p}{m}PQ^{-1} - V'(q)\right)\psi(q) = 0. \tag{161}$$

This equation must be satisfied even when $\gamma = PQ^{-1}$ is real. Therefore, since the real and imaginary parts of the bracket must be zero,

$$\dot{q} = \frac{p}{m}, \tag{162}$$
$$\dot{p} = -V'(q).$$

In addition,

$$G(q) = \left(-\dot{S} - \frac{i\hbar}{2}\frac{\dot{Q}}{Q} - PQ^{-1}\frac{\hbar}{2m} - V(q) - \frac{p^2}{2m} + p\dot{q}\right)\psi(q) = 0, \tag{163}$$

which must hold true even when $\gamma = PQ^{-1}$ is imaginary. Therefore,

$$\dot{S} = p\dot{q} - \left(V(q) + \frac{p^2}{2m}\right), \tag{164}$$
$$\dot{Q} = i\frac{P}{m}.$$

45

Finally,

$$G''(q) = \left(-i \left(\dot{P}Q^{-1} - PQ^{-2}\dot{Q} \right) + \frac{1}{m} \left[PQ^{-1} \right]^2 - V''(q) \right) \psi(q) = 0, \tag{165}$$

which is verified when

$$\dot{P} = iV''(q)Q. \tag{166}$$

15 Semiclassical Dynamics in the Gaussian-Hermite Basis

Consider the time-dependent Schrödinger equation,

$$i\hbar \frac{\partial \psi}{\partial t} = \frac{\hat{p}^2}{2m}\psi + V(x,t)\psi, \tag{167}$$

for the time-dependent harmonic potential with $k(t) = m\omega(t)^2$,

$$V(x,t) = V_0(t) + \frac{1}{2}k(t)(x - x_e(t))^2.$$

An exact solution of Eq. (167) can be written, as follows:

$$\psi_0(x,t) = e^{iS(t)/\hbar}\varphi_0(x,t), \tag{168}$$

with φ_0 the first element of the Gaussian-Hermite basis-set,

$$\varphi_\nu(x,t) = H_\nu(\hbar^{-1/2}|Q|^{-1}(x-q))A(Q,\nu)e^{-\gamma(x-q)^2/(2\hbar) + \frac{i}{\hbar}p(x-q)}, \tag{169}$$

where $\nu = 0, 1, 2, \ldots$ and the parameters $\gamma = PQ^{-1}$, P, Q, q and p are time-dependent. Further, H_ν are Hermite polynomials, and $A(Q,\nu)$ are normalization constants,

$$A(Q,\nu) = 2^{-\nu/2}(\nu!)^{-1/2}\pi^{-1/4}\hbar^{-1/4}Q^{-(\nu+1)/2}\bar{Q}^{\nu/2}, \tag{170}$$

with $\bar{Q}$ the conjugate of Q. Therefore, since $H_0 = 1$,

$$\varphi_0(x,t) = \pi^{-1/4}\hbar^{-1/4}|Q|^{-1/2}e^{-\gamma(x-q)^2/(2\hbar) + \frac{i}{\hbar}p(x-q)}. \tag{171}$$

The action $S(t)$ and conjugate variables q and p evolve classically:

$$\begin{aligned}
\dot{S} &= \frac{p^2}{2m} - V(q,t), \\
\dot{q} &= \frac{p}{m}, \\
\dot{p} &= -\frac{\partial V}{\partial q}.
\end{aligned} \tag{172}$$

Finally, the equations of motion for Q and P are obtained by substituting $\psi_0(x,t)$ into Eq. (167):

$$\dot{Q} = i\frac{P}{m},$$
$$\dot{P} = i\frac{\partial^2 V}{\partial q^2}Q, \tag{173}$$

or,

$$\ddot{Q} = -\frac{1}{m}\frac{\partial^2 V}{\partial q^2}Q$$
$$= -\frac{k(t)}{m}Q, \tag{174}$$
$$= -\omega(t)^2 Q,$$

with Q and P defined as linear combinations of the partial derivatives of $q(t)$ and $p(t)$ with respect to their initial values, as follows:

$$Q(t) = \frac{\partial q(t)}{\partial q(0)}Q(0) + i\frac{\partial q(t)}{\partial p(0)}P(0),$$
$$P(t) = \frac{\partial p(t)}{\partial p(0)}P(0) - i\frac{\partial p(t)}{\partial q(0)}Q(0), \tag{175}$$

since Q and P, defined according to Eqs. (175), satisfy Eqs. (173). According to Eqs. (175), Q and P satisfy the following relation:

$$\bar{Q}P + \bar{P}Q = 2, \tag{176}$$

which is equivalent to $\mathrm{Re}[\gamma] = |Q|^{-2}$ and determine the position and momentum uncertainties of $\varphi_\nu(x,t)$, as follows:

$$\Delta x = \sqrt{\hbar(\nu + 1/2)}|Q|,$$

and

$$\Delta p = \sqrt{\hbar(\nu + 1/2)}|P|.$$

Therefore, $\gamma(t) = |Q(t)|^{-2} + i\mathrm{Im}[P(t)/Q(t)]$. When $\gamma(0)$ is real, $Q(0) = \gamma^{-1/2}$ and $P(0) = \gamma^{1/2}$ since $\gamma(0) = P(0)/Q(0)$.

Note that, according to Eqs. (173), $Q(t)$ and $P(t)$ remain real and purely imaginary, respectively, when choosing initial conditions with $Q(0) = \mathrm{Re}[Q(0)]$ real and $P(0) = i\mathrm{Im}[P(0)]$. With such initial conditions, $\gamma(t) = Q(t)^{-2} + P(t)/Q(t)$. In particular, when choosing $P(0) = i/Q(0)$, with real $Q(0)$, $\mathrm{Re}[\gamma(0)] = \mathrm{Im}[\gamma(0)] = Q(0)^{-2}$, or $\gamma(0) = (1+i)/Q(0)^2$.

Other possible solutions can be obtained by defining the raising and lowering ladder operators Ł$_+$ and Ł$_-$, respectively, as follows [Hagedorn, G. A.: Raising and lowering operators for semiclassical wave packets. *Ann. Phys.* **269**, 77-104 (1998)] and [Computing Semiclassical Quantum Dynamics with Hagedorn Wavepackets. *SIAM J. Sci. Comput.* **31**, 3027-3041 (2008)]:

$$Ł_+ = (2\hbar)^{-1/2}(\bar{P}(x-q) - i\bar{Q}(\hat{p}-p)),$$
$$Ł_- = (2\hbar)^{-1/2}(P(x-q) + iQ(\hat{p}-p)), \tag{177}$$

and noting that

$$[Ł_-, Ł_+] = 1,$$

$$\frac{1}{2}(Ł_- Ł_+ + Ł_+ Ł_-)\psi_\nu = \left(\nu + \frac{1}{2}\right)\psi_\nu, \tag{178}$$

we see that

$$Ł_+\psi_\nu = \sqrt{\nu + 1}\psi_{\nu+1}. \tag{179}$$

Therefore,

$$\psi_\nu(x, t) = Ł_+^\nu \psi_0(x; t) = \sqrt{\nu!} e^{iS(t)/\hbar} \varphi_\nu(x, t), \tag{180}$$

with $\nu = 0, 1, 2, \ldots$ are also solutions of Eq. (167).

15.1   Multidimentional Semiclassical Dynamics

The 2-dimensional generalization of Eq. (171) is:

$$\varphi_0(\mathbf{x}, t) = \pi^{-n/4}\hbar^{-n/4}[\det(\mathbf{Q})]^{-1/2}e^{-(\mathbf{x}-\mathbf{q})\cdot\mathbf{P}\mathbf{Q}^{-1}\cdot(\mathbf{x}-\mathbf{q})/(2\hbar)+\frac{i}{\hbar}\mathbf{p}\cdot(\mathbf{x}-\mathbf{q})}, \tag{181}$$

with $n = 2$. Here, $\mathbf{P}$ and $\mathbf{Q}$ are $n \times n$ matrices that evolve in time, as follows:

$$\dot{\mathbf{Q}} = i\frac{\mathbf{P}}{m},$$
$$\dot{\mathbf{P}} = i\mathbf{k}\mathbf{Q}, \tag{182}$$

or,

$$\ddot{\mathbf{Q}} = -\frac{1}{m}\mathbf{k}\mathbf{Q}, \tag{183}$$

where $\mathbf{k}$ is the hessian matrix of second derivatives $k(i, j) = \partial^2 V(\mathbf{x})/\partial x_i \partial x_j$.

To calculate the $\det(\mathbf{Q})$ for Eq. (181), we use the log-derivative substitution:

$$\mathbf{R} = \frac{\partial}{\partial t}\log[\mathbf{Q}],$$
$$= \dot{\mathbf{Q}}\mathbf{Q}^{-1}, \tag{184}$$

giving,

$$\dot{\mathbf{Q}} = \mathbf{R}\mathbf{Q}. \tag{185}$$

Integrating Eq. (185), we obtain:

$$\mathbf{Q} = \exp\int_0^t dt'\mathbf{R}(t'),$$
$$= \prod_{k=1}^n e^{\Delta t \mathbf{R}_k}. \tag{186}$$

48

Therefore,

$$
\begin{aligned}
\det(\mathbf{Q}) &= \prod_{k=1}^{n} \det\left[e^{\Delta t \mathbf{R}_k}\right], \\
&= \prod_{k=1}^{n} e^{\Delta t \ \det[\mathbf{R}_k]}, \\
&= e^{\Delta t \sum_{k=1}^{n} \mathrm{Tr}[\mathbf{R}_k]}, \\
&= e^{\Delta t \int_0^t dt' \mathrm{Tr}[\mathbf{R}(t')]}.
\end{aligned}
\tag{187}
$$

The equation of motion for $\mathbf{R}$ is obtained from Eq. (184), as follows:

$$
\dot{\mathbf{R}} = \ddot{\mathbf{Q}}\mathbf{Q}^{-1} - \left(\dot{\mathbf{Q}}\mathbf{Q}^{-1}\right)^2.
\tag{188}
$$

Substituting Eqs. (184) and (174) into Eq. (188), we obtain:

$$
\dot{\mathbf{R}} = -\omega(t)^2 - \mathbf{R}^2.
\tag{189}
$$

15.2   Log Derivative Propagation

For constant ω, the propagation of $\mathbf{R} = \dot{\mathbf{Q}}\mathbf{Q}^{-1}$ can be based on Eq. (183):

$$
\ddot{\mathbf{Q}} = -\frac{\mathbf{k}}{m}\mathbf{Q},
\tag{190}
$$

with

$$
\begin{aligned}
\mathbf{Q}(t) &= \mathbf{Q}(0)\cos\left(\sqrt{\frac{\mathbf{k}}{m}}t\right) + \frac{\dot{\mathbf{Q}}(0)}{\sqrt{\frac{\mathbf{k}}{m}}}\sin\left(\sqrt{\frac{\mathbf{k}}{m}}t\right), \\
\dot{\mathbf{Q}}(t) &= \dot{\mathbf{Q}}(0)\cos\left(\sqrt{\frac{\mathbf{k}}{m}}t\right) - \mathbf{Q}(0)\sqrt{\frac{\mathbf{k}}{m}}\sin\left(\sqrt{\frac{\mathbf{k}}{m}}t\right).
\end{aligned}
\tag{191}
$$

Therefore, defining the matrix $\omega = \sqrt{\frac{\mathbf{k}}{m}}$, we obtain:

$$
\mathbf{R}(t) = \left(\mathbf{R}(0)\cos[\omega t] - \omega\sin[\omega t]\right)\left(\cos[\omega t] + \mathbf{R}(0)\sin[\omega t]/\omega\right)^{-1}.
\tag{192}
$$

When ω changes slowly with time,

$$
\mathbf{R}(t+\delta) = \left(\mathbf{R}(t)\cos[\omega(t)\delta] - \omega(t)\sin[\omega(t)\delta]\right)\left(\cos[\omega(t)\delta] + \mathbf{R}(t)\sin[\omega(t)\delta]/\omega(t)\right)^{-1}.
\tag{193}
$$

When the instantaneous normal modes (*i.e.*, eigenvectors of $\mathbf{K}$) are approximately constant, we can solve Eq. (190) by transforming it into a problem of n 1-dimensional equations, as follows [[*J. Chem. Phys.* (1999) **110**:9922-9936]]:

$$
\begin{aligned}
\dot{\tilde{\mathbf{Q}}}(j,j) &= i\frac{\tilde{\mathbf{P}}(j,j)}{m}, \\
\dot{\tilde{\mathbf{P}}}(j,j) &= i\tilde{k}(j)\tilde{\mathbf{Q}}(j,j),
\end{aligned}
\tag{194}
$$

where $j = 1, \cdots, n$ and $\omega(j)$ are the time-dependent frequencies obtained, as follows:

$$
\mathbf{L}^\dagger \cdot \mathbf{K}(t) \cdot \mathbf{L} = m\tilde{\omega}(t)^2.
\tag{195}
$$

where $\tilde{\omega}$ is a diagonal matrix. The new variables $\tilde{\mathbf{Q}}$ and $\tilde{\mathbf{P}}$, introduced by Eq. (194), are defined according to the analogous transformations,

$$
\begin{aligned}
\tilde{\mathbf{Q}}(t) &= \mathbf{L}^\dagger \cdot \mathbf{Q}(t) \cdot \mathbf{L}, \\
\tilde{\mathbf{P}}(t) &= \mathbf{L}^\dagger \cdot \mathbf{P}(t) \cdot \mathbf{L}.
\end{aligned}
\tag{196}
$$

Assuming that $\mathbf{L}$ are approximately constant, we obtain Eq. (194) by computing the time-derivative of Eqs. (196) and substituting $\dot{\mathbf{Q}}$ and $\dot{\mathbf{P}}$ according to Eqs. (182). Note that $\det(\mathbf{Q}) = \det(\tilde{\mathbf{Q}})$ since the $\det(\mathbf{L}) = \det(\mathbf{L}^\dagger) = 1$.

An approximate solution of Eqs. (194) could also be obtained by solving them according to the WKB approximation, as follows. Combining the Eqs. (194), we obtain:

$$
\ddot{\tilde{\mathbf{Q}}}(j,j) = -\tilde{\omega}(j)^2\tilde{\mathbf{Q}}(j,j),
\tag{197}
$$

The possible WKB solutions,

$$
\tilde{\mathbf{Q}}_a(j,j;t) = \tilde{\mathbf{Q}}_a(j,j;0)\exp\left(\pm i\int_0^t \tilde{\omega}(j;t')dt'\right),
\tag{198}
$$

satisfy Eq. (197), as follows:

$$
\ddot{\tilde{\mathbf{Q}}}_a(j,j;t) = -\tilde{\omega}(j;t)^2\tilde{\mathbf{Q}}_a(j,j;t) + \Delta,
\tag{199}
$$

when $|\dot{\tilde{\omega}}(j)| << \tilde{\omega}(j)^2$ and therefore $\Delta = i\dot{\tilde{\omega}}(j;t)\tilde{\mathbf{Q}}_a(j,j;t)$ can be neglected (*i.e.*, WKB approximation).

To satisfy the appropriate boundary conditions, we have the following linear combinations:

$$
\begin{aligned}
\tilde{\mathbf{Q}}_a(j,j;t) &= \tilde{\mathbf{Q}}_a(j,j;0)\cos\left(\int_0^t \tilde{\omega}(j;t')dt'\right) + \frac{\dot{\tilde{\mathbf{Q}}}(0)}{\tilde{\omega}(j;t)}\sin\left(\int_0^t \tilde{\omega}(j;t')dt'\right), \\
\dot{\tilde{\mathbf{Q}}}_a(j,j;t) &= \dot{\tilde{\mathbf{Q}}}_a(j,j;0)\cos\left(\int_0^t \tilde{\omega}(j;t')dt'\right) - \tilde{\mathbf{Q}}_a(j,j;0)\tilde{\omega}(j;t')\sin\left(\int_0^t \tilde{\omega}(j;t')dt'\right).
\end{aligned}
\tag{200}
$$

50

15.3 Normalization of Multidimensional Gaussians

The goal of this section is show that the multidimensional Gaussian, introduced by Eq. (181), is normalized as follows:

$$I_2 = \int dx_1 dx_2 \, e^{-c_{11}x_1^2 - c_{12}x_1x_2 - c_{21}x_2x_1 - c_{22}x_2^2},$$

$$= \int dx_1 dx_2 \, e^{-(x_1 x_2) \begin{pmatrix} c_{11} c_{12} \\ c_{21} c_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}},$$

$$= \int d\mathbf{x} \, e^{-\mathbf{x}^T \cdot \mathbf{c} \cdot \mathbf{x}},$$

$$= \sqrt{\frac{\pi^2}{det(\mathbf{c})}}. \tag{201}$$

First, we introduce the orthogonal transformation $\mathbf{x} = \Gamma \xi$,

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \Gamma_{11} \Gamma_{12} \\ \Gamma_{21} \Gamma_{22} \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix}, \tag{202}$$

where the columns of the matrix Γ are the eigenvectors of $\mathbf{c}$, so that $\Gamma^{-1} \mathbf{c} \Gamma = \lambda$, or

$$\begin{pmatrix} c_{11} c_{12} \\ c_{21} c_{22} \end{pmatrix} \begin{pmatrix} \Gamma_{11} \Gamma_{12} \\ \Gamma_{21} \Gamma_{22} \end{pmatrix} = \begin{pmatrix} \Gamma_{11} \Gamma_{12} \\ \Gamma_{21} \Gamma_{22} \end{pmatrix} \begin{pmatrix} \lambda_1 0 \\ 0 \lambda_2 \end{pmatrix}. \tag{203}$$

Substituting Eq. (202) into Eq. (201), we obtain:

$$I_2 = \int d\xi_1 d\xi_2 \left| \frac{\partial(x_1, x_2)}{\partial(\xi_1, \xi_2)} \right| e^{-(\xi_1 \xi_2) \begin{pmatrix} \lambda_1 0 \\ 0 \lambda_2 \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix}},$$

$$= \int d\xi_1 d\xi_2 \, e^{-\lambda_1 \xi_1^2 - \lambda_2 \xi_2^2}, \tag{204}$$

$$= \sqrt{\frac{\pi^2}{\lambda_1 \lambda_s}} = \sqrt{\frac{\pi^2}{det(\lambda)}}.$$

Furthermore, according to Eq. (203),

$$det(\lambda) = det(\Gamma)^{-1} det(\mathbf{c}) det(\Gamma)$$
$$= det(\mathbf{c}), \tag{205}$$

Therefore, substituting Eq. (205) into Eq. (204), we obtain Eq. (201).

Note: The Jacobian of the orthogonal transformation, introduced by Eq. (202), is equal to 1 since

$$\left| \frac{\partial(x_1, x_2)}{\partial(\xi_1, \xi_2)} \right| = \begin{vmatrix} \dfrac{\partial x_1}{\partial \xi_1} & \dfrac{\partial x_1}{\partial \xi_2} \\ \dfrac{\partial x_2}{\partial \xi_1} & \dfrac{\partial x_2}{\partial \xi_2} \end{vmatrix}$$

$$= det(\Gamma)$$

$$= 1.$$
\tag{206}

To show that $det(\Gamma) = 1$, we use that $det(AB) = det(A)det(B)$ and we obtain:

$$1 = det(\Gamma^{-1}\Gamma),$$
$$= det(\Gamma^{-1})det(\Gamma).$$
\tag{207}

In addition, $\Gamma^{-1} = \Gamma^T$ for orthogonal transformations (*i.e.*, transformations that preserve the norm and orthogonality of vectors), and since always $det(\Gamma^T) = det(\Gamma)$,

$$det(\Gamma) = \frac{1}{det(\Gamma)},$$
\tag{208}

that can only be satisfied by $det(\Gamma) = 1$.

16   Gaussian Beam TDSCF Dynamics

In this section, we implement the Gaussian Beam methodology, in conjunction with the TDSCF approach [Gerber, R.; Buch, V.; Ratner, M. *J. Chem. Phys.* (1982) **77**, 3022; Kotler, Z.; Neria, E.; Nitzan, A. *Computer Physics Communications* (1991) **63** 243-258], to integrate the Schrödinger equation

$$i\hbar \frac{\partial \Phi}{\partial t} = \hat{H}\Phi, \tag{209}$$

for a system described by a wavefunction,

$$\Phi(r, x; t) = \psi(x; t)\chi(r; t),$$
$$= \psi(x; t) \sum_{k=1}^{N} \chi^k(r; t)|k\rangle, \tag{210}$$

where r are nuclear reaction coordinates, x are bath coordinates, and k labels the electronic states. We limit the presentation to a 1-dimensional coordinates x and r. The generalization to many coordinates is straightforward. The system evolves according to the Hamiltonian, defined in the basis of N electronic states, as follows:

$$\hat{H} = \hat{H}_M + \hat{H}_B,$$
$$= \sum_{n=1}^{N} \sum_{m=1}^{N} \left[\hat{H}_M^{nm} + \hat{H}_B^{nm} \right] |n\rangle\langle m|, \tag{211}$$

with

$$\hat{H}_M^{nm} = \hat{T}^M \langle n|m\rangle + V_{nm}^M(r),$$
$$\hat{H}_B^{nm} = \hat{T}^B \langle n|m\rangle + V_{nm}^B(x), \tag{212}$$

where the kinetic energy operators for the reaction coordinates r and the bath coordinates x are:

$$\hat{T}^M = -\frac{\hbar^2}{2m_j} \nabla_r^2,$$
$$\hat{T}^B = -\frac{\hbar^2}{2M_k} \nabla_x^2, \tag{213}$$

Substituting the wavefunction, introduced by Eq. (918), into the time-dependent Schrödinger equation given by Eq. (209), we obtain

$$i\hbar \left[\psi(x; t) \frac{\partial \chi(r; t)}{\partial t} + \frac{\partial \psi(x; t)}{\partial t} \chi(r; t) \right] = \hat{H}_M \psi(x; t)\chi(r; t) + \hat{H}_B \psi(x; t)\chi(r; t). \tag{214}$$

Multiplying by $\psi^*(x)$ and integrating with respect to x, we obtain:

$$i\hbar \left[\sum_{k=1}^{N} \frac{\partial \chi^k(r; t)}{\partial t} |k\rangle + \langle \psi(t)| \frac{\partial \psi(t)}{\partial t} \rangle \sum_{k=1}^{N} \chi^k(r; t)|k\rangle \right] = \sum_{k=1}^{N} \sum_{n=1}^{N} \left[\hat{H}_M^{nk} + \langle \psi(t)|\hat{H}_B^{nk}|\psi(t)\rangle \right] |n\rangle \chi^k(r; t). \tag{215}$$

Furthermore, multiplying by $\langle j|$, we obtain:

$$i\hbar\frac{\partial\chi^j(r;t)}{\partial t} + i\hbar\langle\psi(t)|\frac{\partial\psi(t)}{\partial t}\rangle\chi^j(r;t) = \sum_{k=1}^{N}\left[\hat{H}_M^{jk} + \langle\psi(t)|\hat{H}_B^{jk}|\psi(t)\rangle\right]\chi^k(r;t). \tag{216}$$

Therefore,

$$\begin{aligned}
i\hbar\frac{\partial\chi^j(r;t)}{\partial t} &= \sum_{k=1}^{N}\left[\hat{H}_M^{jk} + \langle\psi(t)|\hat{H}_B^{jk}|\psi(t)\rangle - i\hbar\delta_{jk}\langle\psi(t)|\frac{\partial\psi(t)}{\partial t}\rangle\right]\chi^k(r;t), \\
&= \sum_{k=1}^{N}\left[\langle\psi(t)|\hat{H}_{jk}|\psi(t)\rangle - i\hbar\delta_{jk}\langle\psi(t)|\frac{\partial\psi(t)}{\partial t}\rangle\right]\chi^k(r;t).
\end{aligned} \tag{217}$$

and at fixed $\psi(t)$, we obtain:

$$i\hbar\frac{\partial\chi^j(r;t)}{\partial t}\bigg|_{\psi(t)} = \hat{T}^M\chi^j(r;t) + \sum_{k=1}^{N}\left[V_{jk}^M(r) + \langle\psi(t)|\hat{H}_B^{jk}|\psi(t)\rangle\right]\chi^k(r;t). \tag{218}$$

In addition, multiplying Eq. (214) by $\langle\chi|$, and integrating over r, we obtain:

$$i\hbar\frac{\partial\psi(x;t)}{\partial t} = \sum_{k=1}^{N}\left[-i\hbar\langle\chi^k|\frac{\partial\chi^k(t)}{\partial t}\rangle + \sum_{j=1}^{N}\langle\chi^j(t)|\left[\hat{H}_B^{jk} + \hat{H}_M^{jk}\right]|\chi^k(t)\rangle\right]\psi(x;t), \tag{219}$$

Therefore, for fixed states $\chi_1(t)...\chi_N(t)$, we obtain:

$$\begin{aligned}
i\hbar\frac{\partial\psi(x;t)}{\partial t}\bigg|_{\chi_1(t)...\chi_N(t)} &= \hat{T}^B\psi(x;t) + \sum_{k=1}^{N}\sum_{j=1}^{N}\left[\langle\chi^j(t)|\chi^k(t)\rangle V_{jk}^B(x) + \langle\chi^j(t)|\hat{H}_M^{jk}|\chi^k(t)\rangle\right]\psi(x;t), \\
&= \langle\chi(t)|\hat{H}|\chi(t)\rangle\psi(x;t), \\
&= \hat{T}^B\psi(x;t) + \bar{V}(x;t)\psi(x;t),
\end{aligned} \tag{220}$$

where

$$\bar{V}(x;t) = \sum_{k=1}^{N}\sum_{j=1}^{N}\langle\chi^j(t)|\chi^k(t)\rangle V_{jk}^B(x) + \langle\chi^j(t)|\hat{H}_M^{jk}|\chi^k(t)\rangle. \tag{221}$$

Section 14 showed that the time-dependent Schrödinger equation,

$$i\hbar\frac{\partial\psi(x;t)}{\partial t} = -\frac{\hbar^2}{2m}\frac{\partial^2\psi(x;t)}{\partial x^2} + \bar{V}(x;t)\psi(x;t), \tag{222}$$

can be integrated approximately by using the Gaussian-beam ansatz

$$\psi(x;t) = e^{iS(t)/\hbar}\varphi(x,t), \tag{223}$$

54

with

$$\varphi(x;t) = \pi^{-1/4}\hbar^{-1/4}Q^{-1/2}e^{-\gamma(x-q)^2/(2\hbar)+\frac{i}{\hbar}p(x-q)}, \tag{224}$$

with $\gamma = PQ^{-1}$, by propagating the parameters that define the position and frequency modulation of the Gaussian ansatz, as follows:

$$\begin{aligned}\dot{q} &= \frac{p}{m}, \\ \dot{p} &= -\bar{V}'(q;t).\end{aligned} \tag{225}$$

In addition, the parameters that define the Gaussian width evolve, as follows:

$$\begin{aligned}\dot{Q} &= i\frac{P}{m}. \\ \dot{P} &= i\bar{V}''(q;t)Q.\end{aligned} \tag{226}$$

and the classical action is integrated along the trajectory, according to the usual equation of motion:

$$\dot{S} = p\dot{q} - \left(\bar{V}(q;t) + \frac{p^2}{2m}\right). \tag{227}$$

17   Gaussian Decomposition

The goal of this section is to show that any function $f(x)$ can be expanded as a sum of Gaussians, as follows:

$$\begin{aligned}f(x) &= \sum_{j,k} c_{j,k}\phi_{j,k}(x - x_k), \\ &= \sum_{j,k} \left[\frac{\sqrt{\Delta x}}{\sqrt{2\pi\hbar}}\int dp\ e^{+\frac{i}{\hbar}px_k}h_j(p-p_j)f(p)\right]\frac{\sqrt{\Delta x}}{2\pi\hbar}\sqrt{\pi}\sigma e^{-\frac{\sigma^2}{4\hbar^2}(x-x_k)^2+\frac{i}{\hbar}p_j(x-x_k)}, \\ &= \sum_{j,k} c_{j,k}\frac{\sqrt{\Delta x}}{2\pi\hbar}\sqrt{\pi}\sigma e^{-\frac{1}{2}\frac{P}{Q}(x-x_k)^2+\frac{i}{\hbar}p_j(x-x_k)}, \\ &= \sum_{j,k} c_{j,k}\frac{\sqrt{\Delta x}}{\sqrt{2\pi}}\frac{\sigma}{\hbar\sqrt{2}}e^{-\frac{1}{2}\frac{P}{Q}(x-x_k)^2+\frac{i}{\hbar}p_j(x-x_k)}, \\ &= \sum_{j,k} c_{j,k}\frac{\sqrt{\Delta x}}{\sqrt{2\pi}}\sqrt{\frac{P}{Q}}e^{-\frac{1}{2}\frac{P}{Q}(x-x_k)^2+\frac{i}{\hbar}p_j(x-x_k)},\end{aligned} \tag{228}$$

where we introduced the substitution $P = \frac{\sigma}{\hbar\sqrt{2}}$, and $Q = P^{-1}$, with

$$\phi_{j,k}(x - x_k) = \frac{\sqrt{\Delta x}}{2\pi\hbar}\int dp\ e^{\frac{i}{\hbar}p(x-x_k)}g_j(p-p_j), \tag{229}$$

with

$$g_j(p - p_j) = e^{-(p-p_j)^2/\sigma^2}, \tag{230}$$

and

$$h_j(p - p_j) \equiv \frac{g_j(p - p_j)}{\sum_k (g_k(p - p_k))^2}, \tag{231}$$

so

$$\sum_j g_j(p - p_j) h_j(p - p_j) = \sum_j g_j(p - p_j) \frac{g_j(p - p_j)}{\sum_k (g_k(p - p_k))^2},$$
$$= 1, \tag{232}$$

giving

$$\phi_{j,k}(x - x_k) = \frac{\sqrt{\Delta x}}{2\pi\hbar} \sqrt{\pi}\sigma e^{-\frac{\sigma^2}{4\hbar^2}(x-x_k)^2 + \frac{i}{\hbar}p_j(x-x_k)}. \tag{233}$$

Typically, we choose the width of the Gaussians as $\sigma = \zeta\Delta p$, with $\zeta \geq 2$. The spacing between Gaussians in p-space $\Delta p = p_{j+1} - p_j = 2\pi\hbar/[\Delta x N]$ and $\Delta x = (x_{max} - x_{min})/N$, so that $\Delta x \Delta p = 2\pi\hbar/N$.

We note that the Gaussian expansion, introduced by Eq. (228) by-passes the usual numerical difficulties of the more general Gaussian expansion, based on the identity operator:

$$\hat{1} = \sum_j \sum_k |k\rangle S_{kj}^{-1} \langle j|, \tag{234}$$

which is verified by the identity

$$\langle j'|\hat{1}|k'\rangle = \sum_j \sum_k \langle j'|k\rangle S_{kj}^{-1} \langle j|k'\rangle, \tag{235}$$

or

$$S_{j'k'} = \sum_j \sum_k S_{j'k} S_{kj}^{-1} S_{jk'},$$
$$= \sum_j \sum_k S_{j'k} (S^{-1})_{kj} S_{jk'}, \tag{236}$$
$$= (S S^{-1} S)_{j'k'},$$

giving

$$f(x) = \sum_j \sum_k \langle x|k\rangle S_{kj}^{-1} \langle j|f\rangle,$$
$$= \sum_k c_k \langle x|k\rangle, \tag{237}$$

56

with

$$c_k = \sum_j S_{kj}^{-1} \langle j | f \rangle. \tag{238}$$

In contrast, the Gaussian decomposition introduced in this section, effectively represents the identity operator as

$$\hat{1} = \sum_j |j\rangle\langle j| \hat{S}^{-1}. \tag{239}$$

In the rest of this section, we show that the coefficients $c_{j,k}$ can be defined, as introduced by Eq. (228), as follows:

$$
\begin{aligned}
c_{j,k} &= \int dx' \psi_{j,k}^*(x') f(x'), \\
&= \int dp \ \psi_{j,k}^*(p) f(p), \\
&= \frac{\sqrt{\Delta x}}{\sqrt{2\pi\hbar}} \int dp \ e^{+\frac{i}{\hbar}p x_k} h_j(p - p_j) f(p),
\end{aligned}
\tag{240}
$$

with

$$f(p) = \frac{1}{\sqrt{2\pi\hbar}} \int dx e^{-\frac{i}{\hbar}px} f(x), \tag{241}$$

and

$$\psi_{j,k}(x) \equiv \frac{\sqrt{\Delta x}}{2\pi\hbar} \int dp \ e^{\frac{i}{\hbar}p(x - x_k)} h_j(p - p_j), \tag{242}$$

where

$$
\begin{aligned}
\psi_{j,k}(p) &= \frac{1}{\sqrt{2\pi\hbar}} \int dx e^{-\frac{i}{\hbar}px} \psi_{j,k}(x), \\
&= \frac{1}{\sqrt{2\pi\hbar}} \int dx e^{-\frac{i}{\hbar}px} \frac{\sqrt{\Delta x}}{2\pi\hbar} \int dp' \ e^{\frac{i}{\hbar}p'(x - x_k)} h_j(p' - p_j), \\
&= \frac{\sqrt{\Delta x}}{\sqrt{2\pi\hbar}} \int dp' \frac{1}{2\pi\hbar} \int dx e^{-\frac{i}{\hbar}(p - p')x} e^{-\frac{i}{\hbar}p' x_k} h_j(p' - p_j), \\
&= \frac{\sqrt{\Delta x}}{\sqrt{2\pi\hbar}} \int dp' \ \delta(p - p') e^{-\frac{i}{\hbar}p' x_k} h_j(p' - p_j), \\
&= \frac{\sqrt{\Delta x}}{\sqrt{2\pi\hbar}} e^{-\frac{i}{\hbar}p x_k} h_j(p - p_j).
\end{aligned}
\tag{243}
$$

To demonstrate Eq. (228), we write:

$$
\begin{aligned}
f(x) &= \int dx' f(x') \delta(x - x'), \\
&= \int dx' f(x') \frac{1}{2\pi\hbar} \int dp e^{\frac{i}{\hbar}p(x - x')},
\end{aligned}
\tag{244}
$$

57

Inserting unity, as defined according to Eq. (232), we obtain:

$$
f(x) = \frac{1}{2\pi\hbar} \int dp \left[\sum_j h_j(p-p_j)g_j(p-p_j) \right] \int dx' f(x') e^{\frac{i}{\hbar}p(x-x')},
$$

$$
= \frac{1}{2\pi\hbar} \int dp \left[\sum_j h_j(p-p_j)g_j(p-p_j) \right] \int dx' f(x') e^{\frac{i}{\hbar}p(x-x')} \int dp' \delta(p-p'), \quad (245)
$$

$$
= \frac{1}{2\pi\hbar} \int dp' \int dp \sum_j h_j(p'-p_j)g_j(p-p_j) \int dx' f(x') e^{\frac{i}{\hbar}(px-p'x')} \delta(p-p').
$$

Replacing $\delta(p-p')$ by its discrete Fourier expansion, we obtain:

$$
f(x) = \frac{1}{2\pi\hbar} \int dp' \int dp \sum_j h_j(p'-p_j)g_j(p-p_j) \int dx' f(x') e^{\frac{i}{\hbar}(px-p'x')} \frac{1}{2\pi\hbar} \sum_k e^{-\frac{i}{\hbar}x_k(p-p')} \Delta x,
$$

$$
= \sum_k \frac{\Delta x}{(2\pi\hbar)^2} \int dp' \int dp \sum_j h_j(p'-p_j)g_j(p-p_j) \int dx' f(x') e^{-\frac{i}{\hbar}p'(x'-x_k)} e^{\frac{i}{\hbar}p(x-x_k)},
$$

$$
= \sum_{k,j} \int dx' \left[\frac{\sqrt{\Delta x}}{2\pi\hbar} \int dp' h_j(p'-p_j) e^{-\frac{i}{\hbar}p'(x'-x_k)} \right] f(x') \frac{\sqrt{\Delta x}}{2\pi\hbar} \int dp\, g_j(p-p_j) e^{\frac{i}{\hbar}p(x-x_k)},
$$

$$
= \sum_{k,j} \int dx' \psi^*_{j,k}(x') f(x') \frac{\sqrt{\Delta x}}{2\pi\hbar} \int dp\, g_j(p-p_j) e^{\frac{i}{\hbar}p(x-x_k)},
$$

$$
= \sum_{k,j} c_{j,k} \frac{\sqrt{\Delta x}}{2\pi\hbar} \int dp\, g_j(p-p_j) e^{\frac{i}{\hbar}p(x-x_k)},
$$

$$
= \sum_{k,j} c_{j,k} \phi_{j,k}(x-x_k).
$$

$$(246)$$

17.1 Wide Gaussian Basis Approximation

This subsection discusses the limiting case when $\sigma >> \Delta p$, making the denominator of $h_j(p-p_j)$ approximately constant for all p within the support of $g_j(p-p_j)$. In that limit, $h_j(p-p_j)$ becomes proportional to $g_j(p-p_j)$ (*i.e.*, $h_j(p-p_j) \approx \xi g_j(p-p_j)$) with $\xi = 1/\sum_k (g_k(p_j-p_k))^2$, and the expansion coefficients can be computed analytically.

Considering that the initial state is a Gaussian:

$$
f(x) = \left(\frac{\alpha}{\pi} \right)^{1/4} e^{-\frac{\alpha}{2}(x-x_0)^2 + \frac{i}{\hbar}p_0(x-x_0)}, \quad (247)
$$

then

$$
\begin{aligned}
f(p) &= \frac{1}{\sqrt{2\pi\hbar}} \left(\frac{\alpha}{\pi}\right)^{1/4} \int dx e^{-\frac{i}{\hbar}px} e^{-\frac{\alpha}{2}(x-x_0)^2 + \frac{i}{\hbar}p_0(x-x_0)} \\
&= \frac{1}{\sqrt{2\pi\hbar}} \left(\frac{\alpha}{\pi}\right)^{1/4} e^{-\frac{i}{\hbar}px_0} \int dx e^{-\frac{i}{\hbar}(p-p_0)(x-x_0)} e^{-\frac{\alpha}{2}(x-x_0)^2} \\
&= \frac{1}{\sqrt{2\pi\hbar}} \left(\frac{\alpha}{\pi}\right)^{1/4} e^{-\frac{i}{\hbar}px_0} \sqrt{\frac{2\pi}{\alpha}} e^{-\frac{(p-p_0)^2}{\hbar^2 2\alpha}} \\
&= \frac{1}{(\hbar^2\alpha\pi)^{1/4}} e^{-\frac{1}{2\alpha\hbar^2}(p-p_0)^2 - \frac{i}{\hbar}px_0}.
\end{aligned}
\tag{248}
$$

Substituting into Eq. (240), with $h_j(p-p_j) = \xi g_j(p-p_j)$, we obtain:

$$
\begin{aligned}
c_{j,k} &= \frac{\sqrt{\Delta x}}{\sqrt{2\pi\hbar}} \int dp \; e^{+\frac{i}{\hbar}px_k} h_j(p-p_j) f(p), \\
&= \xi \frac{\sqrt{\Delta x}}{\sqrt{2\pi\hbar}} \int dp \; e^{+\frac{i}{\hbar}px_k} g_j(p-p_j) f(p), \\
&= \xi \frac{\sqrt{\Delta x}}{\sqrt{2\pi\hbar}} \int dp \; e^{+\frac{i}{\hbar}px_k} e^{-(p-p_j)^2/\sigma^2} f(p), \\
&= \frac{\xi}{(\alpha\pi)^{1/4}} \frac{\sqrt{\Delta x}}{\hbar\sqrt{2\pi}} \int dp \; e^{-(p-p_j)^2/\sigma^2} e^{-\frac{1}{2\alpha\hbar^2}(p-p_0)^2 - \frac{i}{\hbar}p(x_0-x_k)}, \\
&= \xi \frac{\sqrt{\Delta x}}{(\alpha\pi)^{1/4}} \frac{1}{\hbar\sqrt{2\pi}} \int dp \; e^{-(p-p_j)^2/\sigma^2} e^{-\frac{1}{2\alpha\hbar^2}(p-p_0)^2 - \frac{i}{\hbar}p(x_0-x_k)}, \\
&= \xi \frac{\sqrt{\Delta x}}{(\alpha\pi)^{1/4}} \frac{1}{\hbar\sqrt{2\pi}} \int dp \; e^{-p^2(\frac{1}{\sigma^2}+\frac{1}{2\alpha\hbar^2})} e^{-(p_j^2-2pp_j)/\sigma^2} e^{-\frac{1}{2\alpha\hbar^2}(p_0^2-2pp_0)-\frac{i}{\hbar}p(x_0-x_k)}, \\
&= \xi \frac{\sqrt{\Delta x}}{(\alpha\pi)^{1/4}} \frac{1}{\hbar\sqrt{2\pi}} \int dp \; e^{-p^2(\frac{1}{\sigma^2}+\frac{1}{2\alpha\hbar^2})+2p\left[(\frac{p_j}{\sigma^2}+\frac{p_0}{2\alpha\hbar^2})-\frac{i}{2\hbar}(x_0-x_k)\right]} e^{-\frac{p_j^2}{\sigma^2}-\frac{p_0^2}{2\alpha\hbar^2}}, \\
&= \xi \frac{\sqrt{\Delta x}}{(\alpha\pi)^{1/4}} \sqrt{\frac{\sigma^2\alpha}{2\alpha\hbar^2+\sigma^2}} e^{-p_j^2/\sigma^2-p_0^2/(2\alpha\hbar^2)} e^{\frac{\sigma^2 2\alpha\hbar^2(2p_j/\sigma^2+2p_0/(2\alpha\hbar^2)-\frac{i}{\hbar}(x_0-x_k))^2}{4(2\alpha\hbar^2+\sigma^2)}} \\
&= \xi \frac{\sqrt{\Delta x}}{(\gamma_0\pi)^{1/4}} \frac{1}{\hbar} \sqrt{\frac{\gamma\gamma_0}{\gamma_0+\gamma}} e^{-\frac{1}{2\gamma}p_j^2-\frac{1}{2\gamma_0}p_0^2} e^{\frac{4\gamma\gamma_0(\frac{1}{2\gamma}p_j+\frac{1}{2\gamma_0}p_0-\frac{i}{2\hbar}(x_0-x_k))^2}{2(\gamma_0+\gamma)}}
\end{aligned}
\tag{249}
$$

or

$$
\begin{aligned}
c_{j,k} &= \xi \frac{\sqrt{\Delta x}}{(\gamma_0\pi)^{1/4}} \frac{1}{\hbar} \sqrt{\frac{\gamma\gamma_0}{\gamma_0+\gamma}} e^{-\frac{1}{2\gamma}p_j^2-\frac{1}{2\gamma_0}p_0^2+2\frac{\gamma\gamma_0}{(\gamma_0+\gamma)}\left(\frac{1}{2\gamma}p_j+\frac{1}{2\gamma_0}p_0-\frac{i}{2\hbar}(x_0-x_k)\right)^2} \\
&= \xi \frac{\sqrt{\Delta x}}{(\gamma_0\pi)^{1/4}} \frac{1}{\hbar} \sqrt{\frac{\gamma\gamma_0}{\gamma_0+\gamma}} e^{-\frac{1}{2(\gamma_0+\gamma)}\left[(p_j-p_0)^2+\frac{\gamma_0\gamma}{\hbar^2}(x_0-x_k)^2\right]-\frac{i}{\hbar}(x_0-x_k)\frac{(\gamma_0 p_j+\gamma p_0)}{(\gamma_0+\gamma)}}
\end{aligned}
\tag{250}
$$

where we introduced the substitutions $\gamma = \sigma^2/2$ and $\gamma_0 = \hbar^2\alpha$.

59

Therefore, the initial state introduced by Eq. (228) can be expanded by using the coefficients introduced in Eq. (249), as follows:

$$f(x) = \sum_{j,k} c_{j,k} \sqrt{\frac{\Delta x}{2\pi}} \sqrt{\frac{P}{Q}} e^{-\frac{1}{2}PQ^{-1}(x-x_k)^2 + \frac{i}{\hbar}p_j(x-x_k)}. \tag{251}$$

while the short-time approximation for the time-evolved initial state is

$$\begin{aligned}
f(x;\tau) &= \sum_{j,k} c_{j,k} \sqrt{\frac{\Delta x}{2\pi}} \sqrt{\frac{P}{Q_\tau^{j,k}}} e^{-\frac{1}{2}\frac{P_\tau^{j,k}}{Q_\tau^{j,k}}(x-x_k(\tau;j))^2 + \frac{i}{\hbar}p_j(\tau;k)(x-x_k(\tau;j))+iS_\tau^{j,k}}, \\
&= \sum_{j,k} c_{j,k} \phi_{j,k}(x - x_k(\tau;j))
\end{aligned} \tag{252}$$

Analogously to the expansion of the initial state $f(x)$, we can expand each of the Gaussians in Eq. (252), as follows:

$$\phi_{j,k}(x - x_k(\tau;j)) = \sum_{j',k'} c_{j',k'}^{j,k}(\tau) \phi_{j',k'}(x - x_{k'}), \tag{253}$$

giving

$$\begin{aligned}
f(x;\tau) &= \sum_{j,k} c_{j,k} \phi_{j,k}(x - x_k(\tau;j)) \\
&= \sum_{j',k'} \left[\sum_{j,k} c_{j,k} c_{j',k'}^{j,k}(\tau) \right] \phi_{j',k'}(x - x_{k'}) \\
&= \sum_{j',k'} c_{j',k'}(\tau) \phi_{j',k'}(x - x_{k'}),
\end{aligned} \tag{254}$$

where

$$c_{j',k'}(\tau) = \sum_{j,k} c_{j,k} c_{j',k'}^{j,k}(\tau). \tag{255}$$

The expansion coefficients $c_{j',k'}^{j,k}(\tau)$ are obtained by first computing the Fourier transform of the Gaussians, as follows:

$$\begin{aligned}
\phi_{j,k}(p - p_j(\tau;k)) &= \frac{1}{\sqrt{2\pi\hbar}} \sqrt{\frac{2Q_\tau^{j,k}\pi}{P_\tau^{j,k}}} e^{iS_\tau^{j,k}} \sqrt{\frac{\Delta x}{2\pi}} \sqrt{\frac{P}{Q_\tau^{j,k}}} e^{-\frac{Q_\tau^{j,k}}{2\hbar^2 P_\tau^{j,k}}(p-p_j(\tau;k))^2 - \frac{i}{\hbar}px_k(\tau;j)} \\
&= \frac{\sqrt{\Delta x}}{\sqrt{2\pi\hbar}} \sqrt{\frac{P}{P_\tau^{j,k}}} e^{-\frac{Q_\tau^{j,k}}{2P_\tau^{j,k}}(p-p_j(\tau;k))^2 - \frac{i}{\hbar}px_k(\tau;j)+iS_\tau^{j,k}}
\end{aligned} \tag{256}$$

and then substituting into Eq. (240) as before, with $h_j(p - p_j) = \xi g_j(p - p_j)$, to obtain:

$$
\begin{aligned}
c_{j',k'}^{j,k}(\tau) &= \xi \frac{\sqrt{\Delta x}}{\sqrt{2\pi\hbar}} \int dp \ e^{+\frac{i}{\hbar}px_{k'}} e^{-(p-p_{j'})^2/\sigma^2} \phi_{j,k}(p - p_j(\tau;k)), \\
&= \xi \frac{\Delta x}{2\pi\hbar} \sqrt{\frac{P}{P_\tau^{j,k}}} \int dp \ e^{-(p-p_{j'})^2/\sigma^2} e^{-\frac{Q_\tau^{j,k}}{2P_\tau^{j,k}}(p-p_j(\tau;k))^2 - \frac{i}{\hbar}p(x_k(\tau;j)-x_{k'})+iS_\tau^{j,k}}, \\
&= \xi \frac{\Delta x}{2\pi\hbar} \sqrt{\frac{P}{P_\tau^{j,k}}} \int dp \ e^{-(p^2+p_{j'}^2-2pp_{j'})/\sigma^2} e^{-\frac{Q_\tau^{j,k}}{2P_\tau^{j,k}}(p^2+p_j(\tau;k)^2-2pp_j(\tau;k)) - \frac{i}{\hbar}p(x_k(\tau;j)-x_{k'})+iS_\tau^{j,k}}, \\
&= \xi \frac{\Delta x}{2\pi\hbar} \sqrt{\frac{P}{P_\tau^{j,k}}} \int dp \ e^{-p^2\left(\frac{1}{\sigma^2}+\frac{Q_\tau^{j,k}}{2P_\tau^{j,k}}\right)+p\left(2p_{j'}/\sigma^2+\frac{Q_\tau^{j,k}}{P_\tau^{j,k}}p_j(\tau;k)-\frac{i}{\hbar}(x_k(\tau;j)-x_{k'})\right)} \\
&\quad \times e^{-p_{j'}^2/\sigma^2 - \frac{Q_\tau^{j,k}}{2P_\tau^{j,k}}p_j(\tau;k)^2+iS_\tau^{j,k}},
\end{aligned}
\tag{257}
$$

which can be simplified, as follows:

$$
\begin{aligned}
c_{j',k'}^{j,k}(\tau) &= \xi \frac{\Delta x}{2\pi\hbar} \sqrt{\frac{P}{P_\tau^{j,k}} \frac{\pi\sigma^2 2P_\tau^{j,k}}{\left(2P_\tau^{j,k}+\sigma^2 Q_\tau^{j,k}\right)}} exp\left[-p_{j'}^2/\sigma^2 - \frac{1}{2}p_j(\tau;k)^2 Q_\tau^{j,k}/P_\tau^{j,k} + iS_\tau^{j,k}\right] \\
&\quad \times exp\left[\frac{\sigma^2 2\frac{P_\tau^{j,k}}{Q_\tau^{j,k}}\left(2p_{j'}/\sigma^2+p_j(\tau;k)Q_\tau^{j,k}/P_\tau^{j,k}-\frac{i}{\hbar}(x_k(\tau;j)-x_{k'})\right)^2}{4\left(2\frac{P_\tau^{j,k}}{Q_\tau^{j,k}}+\sigma^2\right)}\right], \\
&= \frac{\xi\Delta x}{\sqrt{2\pi\hbar}} \sqrt{\frac{P}{\hbar} \frac{\sigma^2}{\left(2P_\tau^{j,k}+\sigma^2 Q_\tau^{j,k}\right)}} exp\left[-p_{j'}^2/\sigma^2 - \frac{1}{2}p_j(\tau;k)^2 Q_\tau^{j,k}/P_\tau^{j,k} + \frac{i}{\hbar}S_\tau^{j,k}\right] \\
&\quad \times exp\left[\frac{\sigma^2 2\frac{P_\tau^{j,k}}{Q_\tau^{j,k}}\left(2p_{j'}/\sigma^2+p_j(\tau;k)Q_\tau^{j,k}/P_\tau^{j,k}-\frac{i}{\hbar}(x_k(\tau;j)-x_{k'})\right)^2}{4\left(2\frac{P_\tau^{j,k}}{Q_\tau^{j,k}}+\sigma^2\right)}\right]. \\
&= \frac{\xi\Delta x}{\sqrt{2\pi\hbar}} \sqrt{\frac{P}{\hbar Q_\tau^{j,k}} \frac{\gamma}{(\gamma_\tau+\gamma)}} exp\left[-p_{j'}^2/\sigma^2 - \frac{1}{2}p_j(\tau;k)^2\frac{Q_\tau^{j,k}}{P_\tau^{j,k}} + \frac{i}{\hbar}S_\tau^{j,k}\right] \\
&\quad \times exp\left[\frac{\left(\frac{2}{\sigma^2}p_{j'}+p_j(\tau;k)\frac{Q_\tau^{j,k}}{P_\tau^{j,k}}-\frac{i}{\hbar}(x_k(\tau;j)-x_{k'})\right)^2}{2\left(\frac{2}{\sigma^2}+\frac{Q_\tau^{j,k}}{P_\tau^{j,k}}\right)}\right].
\end{aligned}
\tag{258}
$$

Expanding the exponent [], we obtain:

$$
[] = -\frac{1}{2\gamma}p_{j'}^2 - \frac{1}{2\gamma_\tau}p_j(\tau;k)^2 + \frac{i}{\hbar}S_\tau^{j,k}
$$
$$
+ 2\frac{\gamma_\tau\gamma}{(\gamma_\tau+\gamma)}\left(\left(\frac{1}{4\gamma^2}p_{j'}^2 + \frac{1}{4\gamma_\tau^2}p_j(\tau;k)^2 + 2\frac{1}{2\gamma}p_{j'}\frac{1}{2\gamma_\tau}p_j(\tau;k)\right)\right.
$$
$$
\left. -\frac{1}{4\hbar^2}(x_k(\tau;j)-x_{k'})^2 - 2\frac{i}{2\hbar}(x_k(\tau;j)-x_{k'})\left(\frac{1}{2\gamma}p_{j'}+\frac{1}{2\gamma_\tau}p_j(\tau;k)\right)\right)
$$

(259)

$$
= -\frac{1}{2}\frac{1}{(\gamma_\tau+\gamma)}(p_{j'}-p_j(\tau;k))^2 - \frac{1}{2}\frac{\gamma_\tau\gamma}{(\gamma_\tau+\gamma)}\frac{1}{\hbar^2}(x_k(\tau;j)-x_{k'})^2
$$
$$
+ \frac{i}{\hbar}\left[S_\tau^{j,k} - \frac{1}{(\gamma_\tau+\gamma)}(x_k(\tau;j)-x_{k'})(\gamma_\tau p_{j'}+\gamma p_j(\tau;k))\right]
$$

(260)

Therefore,

$$
c_{j',k'}^{j,k}(\tau) = \frac{\xi\Delta x}{\hbar\sqrt{2\pi}}\sqrt{\frac{P}{Q_\tau^{j,k}}\frac{\gamma}{(\gamma_\tau+\gamma)}}e^{-\frac{1}{2(\gamma_\tau+\gamma)}\left[(p_{j'}-p_j(\tau;k))^2+\frac{\gamma_\tau\gamma}{\hbar^2}(x_k(\tau;j)-x_{k'})^2\right]}
$$
$$
\times e^{\frac{i}{\hbar}\left(S_\tau^{j,k}-(x_k(\tau;j)-x_{k'})\frac{(\gamma_\tau p_{j'}+\gamma p_j(\tau;k))}{(\gamma_\tau+\gamma)}\right)}
$$

(261)

17.2   Problem: WGB

Write a program to propagate a wavepacket, initialized according to Eq. (251) with coefficients defined according to (250), with $\alpha = 1$, $p_0 = 0$, $x_0 = -2\eta$, with $\eta = 1.3544$, and evolving in a quartic double-well potential $V(x) = x^4/(16\eta) - x^2/2$. Implement the WGB propagation method to evolve the expansion coefficients, according to Eqs. (255) and (261).Compare your results to the corresponding proagation based on the SOFT method described in Sec. 6.

Solution in Sec. 61.23.

17.3   Concatenation

Concatenating short-time propagators, according to Eq. (255), we obtain:

$$
c_{j',k'}(\tau) = \sum_{j,k}\xi\frac{\sqrt{\Delta x}}{(\gamma_0\pi)^{1/4}}\frac{1}{\hbar}\sqrt{\frac{\gamma\gamma_0}{\gamma_0+\gamma}}\frac{\xi\Delta x}{\hbar\sqrt{2\pi}}\sqrt{\frac{P}{Q_\tau^{j,k}}\frac{\gamma}{(\gamma_\tau+\gamma)}}
$$
$$
e^{-\frac{1}{2(\gamma_0+\gamma)}\left[(p_j-p_0)^2+\frac{\gamma_0\gamma}{\hbar^2}(x_0-x_k)^2\right]-\frac{i}{\hbar}(x_0-x_k)\frac{(\gamma_0 p_j+\gamma p_0)}{(\gamma_0+\gamma)}}
$$
$$
e^{-\frac{1}{2(\gamma_\tau+\gamma)}\left[(p_{j'}-p_j(\tau;k))^2+\frac{\gamma_\tau\gamma}{\hbar^2}(x_k(\tau;j)-x_{k'})^2\right]+\frac{i}{\hbar}\left[S_\tau^{j,k}-(x_k(\tau;j)-x_{k'})\frac{(\gamma_\tau p_{j'}+\gamma p_j(\tau;k))}{(\gamma_\tau+\gamma)}\right]}
$$

(262)

and

$$c_{j'',k''}(2\tau) = \sum_{j,k,j',k'} \xi \frac{\sqrt{\Delta x}}{(\gamma_0\pi)^{1/4}} \frac{1}{\hbar} \sqrt{\frac{\gamma\gamma_0}{\gamma_0+\gamma}} \frac{\xi\Delta x}{\hbar\sqrt{2\pi}} \sqrt{\frac{P}{Q_\tau^{j,k}} \frac{\gamma}{(\gamma_\tau+\gamma)}} \frac{\xi\Delta x}{\hbar\sqrt{2\pi}} \sqrt{\frac{P}{Q_\tau^{j',k'}} \frac{\gamma}{(\gamma_\tau'+\gamma)}}$$

$$e^{-\frac{1}{2(\gamma_0+\gamma)}\left[(p_j-p_0)^2+\frac{\gamma_0\gamma}{\hbar^2}(x_0-x_k)^2\right]-\frac{i}{\hbar}(x_0-x_k)\frac{(\gamma_0 p_j+\gamma p_0)}{(\gamma_0+\gamma)}}$$

(263)

$$e^{-\frac{1}{2(\gamma_\tau+\gamma)}\left[\left(p_{j'}-p_j(\tau;k)\right)^2+\frac{\gamma_\tau\gamma}{\hbar^2}(x_k(\tau;j)-x_{k'})^2\right]+\frac{i}{\hbar}\left[S_\tau^{j,k}-(x_k(\tau;j)-x_{k'})\frac{\left(\gamma_\tau p_{j'}+\gamma p_j(\tau;k)\right)}{(\gamma_\tau+\gamma)}\right]}$$

$$e^{-\frac{1}{2(\gamma_\tau'+\gamma)}\left[\left(p_{j''}-p_{j'}(\tau;k')\right)^2+\frac{\gamma_\tau'\gamma}{\hbar^2}(x_{k'}(\tau;j')-x_{k''})^2\right]+\frac{i}{\hbar}\left[S_\tau^{j',k'}-(x_{k'}(\tau;j')-x_{k''})\frac{\left(\gamma_\tau' p_{j''}+\gamma p_{j'}(\tau;k')\right)}{(\gamma_\tau'+\gamma)}\right]}$$

Equation 263 can be evaluated by Monte Carlo. Given the initial state centered at x_0 and p_0, we could sample x_k and p_j with the corresponding Gaussian distributions, and propagate these dynamical variables for time τ. Then, sample $x_{k'}$ and $p_{j'}$ with the Gaussian distributions centered at $x_k(\tau;j)$ and $p_j(\tau;k)$, respectively, and propagate them for another time τ. The procedure is repeated n times and for many realizations until reaching convergence at the final propagation time $t = n\tau$.

In the limit when $\tau \to 0$, we have the special case where the exponent [] of Eq. (263) with $\gamma_0 = \gamma$ is

$$\begin{aligned}
[] &= -\frac{1}{4\gamma}\left[(p_j-p_0)^2+(p_{j'}-p_j)^2+(p_{j''}-p_{j'})^2\right] \\
&\quad -\frac{\gamma}{4\hbar^2}\left[(x_0-x_k)^2+(x_k-x_{k'})^2+(x_{k'}-x_{k''})^2\right] \\
&\quad +\frac{i}{2\hbar}\left[-(x_0-x_k)(p_j+p_0)-(x_k-x_{k'})(p_{j'}+p_j)-(x_{k'}-x_{k''})(p_{j''}+p_{j'})\right] \\
&= -\frac{1}{4\gamma}\left[(p_j-p_0)^2+(p_{j'}-p_j)^2+(p_{j''}-p_{j'})^2\right] \\
&\quad -\frac{\gamma}{4\hbar^2}\left[(x_0-x_k)^2+(x_k-x_{k'})^2+(x_{k'}-x_{k''})^2\right] \\
&\quad +\frac{i}{2\hbar}\left[-x_0(p_j+p_0)-x_k(p_{j'}+\cancel{p_j})-x_{k'}(p_{j''}+\cancel{p_{j'}})\right] \\
&\quad +\frac{i}{2\hbar}\left[x_k(\cancel{p_j}+p_0)+x_{k'}(\cancel{p_{j'}}+p_j)+x_{k''}(p_{j''}+p_{j'})\right] \\
&= -\frac{1}{4\gamma}\left[(p_0-p_j)^2+(p_j-p_{j'})^2+(p_{j'}-p_{j''})^2\right] \\
&\quad -\frac{\gamma}{4\hbar^2}\left[(x_0-x_k)^2+(x_k-x_{k'})^2+(x_{k'}-x_{k''})^2\right] \\
&\quad -\frac{i}{2\hbar}\left[(p_0+p_j)x_0+(p_{j'}-p_0)x_k-(p_{j''}+p_{j'})x_{j''}+(p_{j''}-p_j)x_{k'}\right]
\end{aligned}$$

(264)

17.4 Short Time Propagator

When the propagation time step τ is sufficiently small, we can expand all dynamical variables to first order in τ:

$$
\begin{aligned}
p_j(\tau;k) &= p_j + F(x_k)\tau, \\
x_k(\tau;j) &= x_k + \frac{p_j}{m}\tau, \\
S_\tau^{j,k} &= \frac{p_j^2}{2m}\tau - V(x_k)\tau, \\
Q_\tau^{j,k} &= \gamma^{-1/2} + i\frac{\gamma^{1/2}}{m}\tau, \\
P_\tau^{j,k} &= P_\tau^{j,k}(x_k) = \gamma^{1/2} + iV''(x_k)\gamma^{-1/2}\tau, \\
\gamma_\tau &= \gamma_\tau(x_k) = \frac{\gamma^{1/2} + iV''(x_k)\gamma^{-1/2}\tau}{\gamma^{-1/2} + i\frac{\gamma^{1/2}}{m}\tau},
\end{aligned}
\tag{265}
$$

Integrating over j, we obtain:

$$
\begin{aligned}
c_{j',k'}(\tau) &= \frac{1}{\Delta p}\sum_k \int dp_j\, \xi \frac{\sqrt{\Delta x}}{(\gamma_0\pi)^{1/4}}\frac{1}{\hbar}\sqrt{\frac{\gamma\gamma_0}{\gamma_0+\gamma}}\frac{\xi\Delta x}{\hbar\sqrt{2\pi}}\sqrt{\frac{P}{\left(\gamma^{-1/2}+i\frac{\gamma^{1/2}}{m}\tau\right)}\frac{\gamma}{(\gamma_\tau+\gamma)}} \\
&\quad \exp\left[-\frac{1}{2(\gamma_0+\gamma)}\left[(p_j-p_0)^2 + \frac{\gamma_0\gamma}{\hbar^2}(x_0-x_k)^2\right] - \frac{i}{\hbar}(x_0-x_k)\frac{(\gamma_0 p_j+\gamma p_0)}{(\gamma_0+\gamma)}\right. \\
&\quad -\frac{1}{2(\gamma_\tau+\gamma)}\left[(p_{j'}-(p_j+F\tau))^2 + \frac{\gamma_\tau\gamma}{\hbar^2}((x_k+\frac{p_j}{m}\tau)-x_{k'})^2\right] \\
&\quad \left. +\frac{i}{\hbar}\left[\left(\frac{p_j^2}{2m}-V\right)\tau - (x_k+\frac{p_j}{m}\tau-x_{k'})\frac{(\gamma_\tau p_{j'}+\gamma(p_j+F\tau))}{(\gamma_\tau+\gamma)}\right]\right],
\end{aligned}
\tag{266}
$$

where V, F and γ_τ are functions of x_k. Rearranging the exponent in powers of p_j, we obtain:

$$
\begin{aligned}
c_{j',k'}(\tau) &= \sum_k \int \frac{dp_j}{\Delta p}\, \xi \frac{\sqrt{\Delta x}}{(\gamma_0\pi)^{1/4}}\frac{1}{\hbar}\sqrt{\frac{\gamma\gamma_0}{\gamma_0+\gamma}}\frac{\xi\Delta x}{\hbar\sqrt{2\pi}}\sqrt{\frac{P}{\left(\gamma^{-1/2}+i\frac{\gamma^{1/2}}{m}\tau\right)}\frac{\gamma}{(\gamma_\tau+\gamma)}}e^{-p_j^2 f_1 + p_j f_2 + f_3} \\
&= \frac{1}{\Delta p}\sum_k \xi \frac{\sqrt{\Delta x}}{(\gamma_0\pi)^{1/4}}\frac{1}{\hbar}\sqrt{\frac{\gamma\gamma_0}{\gamma_0+\gamma}}\frac{\xi\Delta x}{\hbar\sqrt{2\pi}}\sqrt{\frac{P}{\left(\gamma^{-1/2}+i\frac{\gamma^{1/2}}{m}\tau\right)}\frac{\gamma}{(\gamma_\tau+\gamma)}}\sqrt{\frac{\pi}{f_1}}e^{\frac{f_2^2}{4f_1}+f_3},
\end{aligned}
\tag{267}
$$

with

$$f_1 = \frac{1}{2(\gamma + \gamma_0)} + \frac{1}{2(\gamma + \gamma_\tau)} - \frac{i\tau}{2\hbar m} + \frac{i\gamma\tau}{(\gamma + \gamma_\tau)\hbar m} + \frac{\gamma\gamma_\tau\tau^2}{2(\gamma + \gamma_\tau)\hbar^2 m^2},$$

$$f_2 = \frac{p_0}{\gamma + \gamma_0} + \frac{p_{j'}}{\gamma + \gamma_\tau} - \frac{F\tau}{\gamma + \gamma_\tau} - i\frac{\gamma_\tau p_{j'}\tau}{(\gamma + \gamma_\tau)\hbar m} - i\frac{F\gamma\tau^2}{(\gamma + \gamma_\tau)\hbar m} - i\frac{\gamma_0(x_0 - x_k)}{(\gamma + \gamma_0)\hbar}$$
$$- i\frac{\gamma x_k}{(\gamma + \gamma_\tau)\hbar} - \frac{\gamma\gamma_\tau\tau x_k}{(\gamma + \gamma_\tau)\hbar^2 m} + i\frac{\gamma x_{k'}}{(\gamma + \gamma_\tau)\hbar} + \frac{\gamma\gamma_\tau\tau x_{k'}}{(\gamma + \gamma_\tau)\hbar^2 m},$$

$$f_3 = -\frac{p_0^2}{2(\gamma + \gamma_0)} - \frac{p_{j'}^2}{2(\gamma + \gamma_\tau)} + \frac{Fp_{j'}\tau}{\gamma + \gamma_\tau} + \frac{\gamma\gamma_\tau x_k x_{k'}}{(\gamma + \gamma_\tau)\hbar^2} - \frac{\gamma\gamma_\tau x_{k'}^2}{2(\gamma + \gamma_\tau)\hbar^2}$$
$$- \frac{F^2\tau^2}{2(\gamma + \gamma_\tau)} - i\frac{\tau V}{\hbar} - i\frac{\gamma p_0(x_0 - x_k)}{(\gamma + \gamma_0)\hbar} - \frac{\gamma\gamma_0(x_0 - x_k)^2}{2(\gamma + \gamma_0)\hbar^2}$$
$$- i\frac{\gamma_\tau p_{j'} x_k}{(\gamma + \gamma_\tau)\hbar} - i\frac{F\gamma\tau x_k}{(\gamma + \gamma_\tau)\hbar} - \frac{\gamma\gamma_\tau x_k^2}{2(\gamma + \gamma_\tau)\hbar^2} + i\frac{\gamma_\tau p_{j'} x_{k'}}{(\gamma + \gamma_\tau)\hbar} + i\frac{F\gamma\tau x_{k'}}{(\gamma + \gamma_\tau)\hbar}. \tag{268}$$

Approximating V, F and γ_τ as $V(x_{k'})$, $F(x_{k'})$ and $\gamma_\tau(x_{k'})$ and integrating over x_k, we obtain:

$$c_{j',k'}(\tau) = f(\gamma_\tau) \int dx_k e^{\frac{f_2^2}{4f_1} + f_3},$$
$$= f(\gamma_\tau) \int dx_k e^{-g_1 x_k^2 + g_2 x_k + g_3}, \tag{269}$$
$$= f(\gamma_\tau) e^{\frac{g_2^2}{4g_1} + g_3}$$

where

$$f(\gamma_\tau) = \frac{1}{\Delta p}\xi\frac{\sqrt{\Delta x}}{(\gamma_0\pi)^{1/4}}\frac{1}{\hbar}\sqrt{\frac{\gamma\gamma_0}{\gamma_0 + \gamma}}\frac{\xi}{\hbar\sqrt{2\pi}}\sqrt{\frac{P}{\left(\gamma^{-1/2} + i\frac{\gamma^{1/2}}{m}\tau\right)}\frac{\gamma}{(\gamma_\tau + \gamma)}}\sqrt{\frac{\pi}{f_1}}. \tag{270}$$

and

$$\frac{g_2^2}{4g_1} + g_3 = -((\hbar^2 m(F^2\gamma_0\hbar m\tau^2 + \gamma_\tau(\hbar m(p_0 - p_{j'})^2$$
$$+ i\gamma_0(p_{j'}^2\tau + 2m\tau V + 2mp_{j'}(x_0 - x_{k'}))))$$
$$+ \gamma^2(\hbar(\gamma_0(F^2\tau^4 + m(-2\tau^2 V + m(x_0 - x_{k'})^2))$$
$$+ \gamma_\tau\tau(p_0^2\tau + 2m\tau V + 2mp_0(x_0 - x_{k'})))$$
$$+ i\hbar^2 m(p_0^2\tau - F^2\tau^3 + 2m\tau V$$
$$+ 2mp_0(x_0 - x_{k'})) - i\gamma_0\gamma_\tau\tau(-2\tau^2 V + m(x_0 - x_{k'})^2))$$
$$+ \gamma\hbar(\hbar^2 m^2((p_0 - p_{j'})^2 + F^2\tau^2) + \gamma_0\gamma_\tau(p_{j'}\tau + m(x_0 - x_{k'}))^2$$
$$+ i\hbar m(\gamma_0(p_{j'}^2\tau + 2m\tau V + 2mp_{j'}(x_0 - x_{k'})) + \gamma_\tau(-\tau(p_{j'}^2 - 2mV)$$
$$+ 2p_0(p_{j'}\tau + mx_0 - mx_{k'})))))/(2\hbar((\gamma + \gamma_0)\hbar m$$
$$+ i\gamma\gamma_0\tau)((\gamma + \gamma_\tau)\hbar m - i\gamma\gamma_\tau\tau)). \tag{271}$$

65

17.5 Expansion Coefficients: Multidimensional States

Substituting Eqs. (231) and (242) into Eq. (**??**), we obtain:

$$c_{l,k} = \int dp \frac{g_l(p - p_l)}{\sum_k g_k(p - p_k)^2} \int dx \frac{\sqrt{\Delta x}}{2\pi\hbar} f(x) e^{-\frac{i}{\hbar}p(x - x_k)}, \tag{272}$$

For an N-dimensional state

$$f(x) = \sum_\xi c_\xi \prod_{j=1}^N f_j^\xi(x_j), \tag{273}$$

we obtain

$$\begin{aligned}
\sum_k g_k(p - p_k)^2 &= \sum_k \prod_{j=1}^N g_k(p_j - p_{j,k})^2 \\
&= \prod_{j=1}^N \sum_{k_j} g_{k_j}(p_j - p_{k_j})^2,
\end{aligned} \tag{274}$$

where k is the index enumerating the multidimentional basis functions and k_j is the index labeling component j of basis function k, centered at $p_{k_j} = p_{j,k}$. Therefore,

$$c_{l,k} = \frac{(\Delta x)^{N/2}}{(2\pi\hbar)^N} \sum_\xi c_\xi \prod_{j=1}^N \int dp_j \frac{g_{l_j}(p_j - p_{l_j})}{\sum_{k_j} g_{k_j}(p_j - p_{k_j})^2} \int dx_j f_j^\xi(x_j) e^{-\frac{i}{\hbar}p_j(x_j - x_{k_j})}, \tag{275}$$

which is simply a sum of products of 1-dimensional integrals.

For an N-dimensional Gaussian,

$$f(x) = \sum_\xi c_\xi e^{-\sum_{j=1}^N \sum_{k=1}^N A_{j,k}^\xi x_j x_k}, \tag{276}$$

we first analyze the particular case when $N = 2$, $A_{jj}^\xi = (1 - \alpha_{jk}^{\xi}{}^2)^{-1}$ and $A_{jk}^\xi = -\alpha_{jk}^\xi/(1 - \alpha_{jk}^{\xi}{}^2)$. We not that (Sakurai, p. 113),

$$e^{-\frac{x_j^2 + x_k^2 - 2\alpha_{jk}^\xi x_j x_k}{(1 - \alpha_{jk}^{\xi}{}^2)}} = \sqrt{1 - \alpha_{jk}^{\xi}{}^2} e^{-x_j^2 - x_k^2} \sum_{n=0}^\infty \frac{\alpha_{jk}^{\xi}{}^n}{2^n n!} H_n(x_j) H_n(x_k). \tag{277}$$

Substituting Eq. (277) into Eq. (276), we obtain:

$$\begin{aligned}
f(x) dx &= \sum_\xi c_\xi e^{-\sum_{j=1}^2 \sum_{k=1}^2 A_{j,k}^\xi x_j x_k} dx, \\
&= \sum_\xi c_\xi \sqrt{1 - \alpha_{jk}^{\xi}{}^2} e^{-x_1^2 - x_2^2} \sum_{n=0}^\infty \frac{\alpha_{jk}^{\xi}{}^n}{2^n n!} H_n(x_1) H_n(x_2) dx_1 dx_2, \\
&= \sum_\xi c_\xi \sum_{n=0}^\infty \frac{\alpha_{jk}^{\xi}{}^n}{2^n n!} \sqrt{1 - \alpha_{jk}^{\xi}{}^2} \prod_{j=1}^2 e^{-x_j^2} H_n(x_j) dx_j,
\end{aligned} \tag{278}$$

Analogously,

$$
\begin{aligned}
f(x)dx &= \sum_\xi c_\xi e^{-\sum_{j=1}^N \sum_{k=1}^N A_{j,k}^\xi x_j x_k} dx, \\
&= \sum_\xi c_\xi \prod_{j=1}^N \prod_{k=1}^N e^{-\frac{x_j^2 + x_k^2 - 2\alpha_{jk}^\xi x_j x_k}{(1-\alpha_{jk}^{\xi^2})}} dx \\
&= \sum_\xi c_\xi \prod_{j=1}^N \prod_{k=1}^N \sum_{n=0}^\infty \sqrt{1-\alpha_{jk}^{\xi^2}} \frac{\alpha_{jk}^{\xi^n}}{2^n n!} H_n(x_j) e^{-x_j^2} H_n(x_k) e^{-x_k^2} dx \\
&= \sum_\xi c_\xi \sum_{n=0}^\infty \prod_{j=1}^N \prod_{k=1}^N \sqrt{1-\alpha_{jk}^{\xi^2}} \frac{\alpha_{jk}^{\xi^n}}{2^n n!} H_n(x_j) e^{-x_j^2} H_n(x_k) e^{-x_k^2} dx
\end{aligned}
\tag{279}
$$

An approximate scheme, based on diagonalization, can be formulated by introducing the orthogonal transformation $\mathbf{x} = \Gamma^\xi \chi$,

$$
\begin{pmatrix} x_1 \\ \cdots \\ x_N \end{pmatrix} = \begin{pmatrix} \Gamma_{11}^\xi \cdots & \Gamma_{1N}^\xi \\ \cdots\cdots & \cdots \\ \Gamma_{N1}^\xi \cdots & \Gamma_{NN}^\xi \end{pmatrix} \begin{pmatrix} \chi_1 \\ \cdots \\ \chi_N \end{pmatrix},
\tag{280}
$$

where the columns of the matrix Γ^ξ are the eigenvectors of $\mathbf{A}^\xi$, so that $\Gamma^{\xi^{-1}} \mathbf{A}^\xi \Gamma^\xi = \lambda^\xi$, or

$$
\begin{pmatrix} A_{11}^\xi \cdots & A_{1N}^\xi \\ \cdots\cdots & \cdots \\ A_{N1}^\xi \cdots & A_{NN}^\xi \end{pmatrix} \begin{pmatrix} \Gamma_{11}^\xi \cdots & \Gamma_{1N}^\xi \\ \cdots\cdots & \cdots \\ \Gamma_{N1}^\xi \cdots & \Gamma_{NN}^\xi \end{pmatrix} = \begin{pmatrix} \Gamma_{11}^\xi \cdots & \Gamma_{1N}^\xi \\ \cdots\cdots & \cdots \\ \Gamma_{N1}^\xi \cdots & \Gamma_{NN}^\xi \end{pmatrix} \begin{pmatrix} \lambda_1^\xi & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \lambda_N^\xi \end{pmatrix}.
\tag{281}
$$

Therefore,

$$
\begin{aligned}
f(x)dx &= \sum_\xi c_\xi e^{-\sum_{j=1}^N \sum_{k=1}^N A_{j,k}^\xi x_j x_k} dx, \\
&= \sum_\xi c_\xi e^{-\mathbf{x}^T \cdot A^\xi \cdot \mathbf{x}} dx, \\
&= \sum_\xi c_\xi e^{-\chi^T \Gamma^{\xi^{-1}} A^\xi \Gamma^\xi \chi} \left| \frac{\partial(x_1, \cdots, x_N)}{\partial(\chi_1, \cdots, \chi_n)} \right| d\chi_1 \cdots d\chi_N, \\
&= \sum_\xi c_\xi e^{-\chi^T \lambda^\xi \chi} d\chi_1 \cdots d\chi_N, \\
&= \sum_\xi c_\xi \prod_{j=1}^N e^{-\lambda_j^\xi \chi_j^2} d\chi_j, \\
&= \sum_\xi c_\xi \prod_{j=1}^N G_j(\chi_j) d\chi_j,
\end{aligned}
\tag{282}
$$

67

since the Jacobian is equal to 1 (line 3 of Eq. (282)). Substituting Eq. (282) into Eq.(342), we obtain:

$$c_{l,k} = \frac{(\Delta x)^{N/2}}{(2\pi\hbar)^N} \sum_{\xi} c_{\xi} \int dp \frac{g_l(p-p_l)}{\sum_k g_k(p-p_k)^2} \int \prod_{j=1}^{N} d\chi_j G_j(\chi_j) e^{-\frac{i}{\hbar}p_j(\sum_{n=1}^{N} \Gamma_{jn}^{\xi} \chi_n - x_{kj})}, \qquad (283)$$

18   Erhenfest Dynamics

The goal of this section is to show that the expectation values $\langle x \rangle = \langle \psi | x | \psi \rangle$ and $\langle p \rangle = \langle \psi | p | \psi \rangle$ evolve according to the classical Hamilton's equations:

$$
\begin{aligned}
\frac{d}{dt}\langle x \rangle &= \frac{\partial}{\partial p}\langle H \rangle \\
&= \frac{\langle p \rangle}{m}, \\
\frac{d}{dt}\langle p \rangle &= -\frac{\partial}{\partial x}\langle H \rangle \\
&= -\langle V' \rangle,
\end{aligned}
\tag{284}
$$

where $H = p^2/2m + V(x)$, and ψ evolves according to the Schrödinger equation,

$$
i\hbar \frac{\partial \psi}{\partial t} = \hat{H}\psi.
\tag{285}
$$

This remarkable result, introduced by Eq. (284), is known as *'Erhenfest's theorem'* and can be demonstrated, as follows. First, we show that

$$
\langle p \rangle = m \frac{d}{dt}\langle x \rangle,
\tag{286}
$$

by using integration by parts, as follows:

$$
\begin{aligned}
\langle p \rangle &= -i\hbar \langle \psi | \frac{\partial}{\partial x} | \psi \rangle, \\
&= -\frac{i\hbar}{2}\langle \psi | \frac{\partial}{\partial x} + \frac{\partial}{\partial x} | \psi \rangle, \\
&= -\frac{i\hbar}{2}\int dx \left[\psi^* \frac{\partial \psi}{\partial x} + \psi^* \frac{\partial \psi}{\partial x} \right], \\
&= -\frac{i\hbar}{2}\int dx \left[\psi^* \frac{\partial \psi}{\partial x} - \psi \frac{\partial \psi^*}{\partial x} \right], \\
&= m \int j \, dx, \\
&= -m \int x \frac{\partial j}{\partial x} \, dx,
\end{aligned}
\tag{287}
$$

where the current $j = -\frac{i\hbar}{2m}\left[\psi^* \frac{\partial \psi}{\partial x} - \psi \frac{\partial \psi^*}{\partial x} \right]$ satisfies the continuity equation,

$$
\frac{d}{dt}\psi^* \psi + \frac{\partial j}{\partial x} = 0.
\tag{288}
$$

Therefore,

$$\langle p \rangle = m \int x \frac{d}{dt} \psi^* \psi \, dx,$$
$$= m \frac{d}{dt} \int \psi^* x \psi \, dx, \tag{289}$$
$$= m \frac{d}{dt} \langle x \rangle.$$

Next, we show that

$$\frac{d}{dt} \langle p \rangle = -\langle V' \rangle, \tag{290}$$

by substituting Eq. (285) into Eq. (287) and integrating by parts, as follows:

$$\frac{d}{dt} \langle p \rangle = m \frac{d}{dt} \int_{-\infty}^{\infty} dx \, j,$$
$$= -\frac{i\hbar}{2} \int_{-\infty}^{\infty} dx \left[\frac{d\psi^*}{dt} \frac{\partial \psi}{\partial x} + \psi^* \frac{\partial}{\partial x} \frac{d\psi}{dt} - \frac{d\psi}{dt} \frac{\partial \psi^*}{\partial x} - \psi \frac{\partial}{\partial x} \frac{d\psi^*}{dt} \right]$$
$$= i\hbar \int_{-\infty}^{\infty} dx \left[\frac{d\psi}{dt} \frac{\partial \psi^*}{\partial x} + \frac{d\psi^*}{dt} \frac{\partial \psi}{\partial x} \right]$$
$$= \int_{-\infty}^{\infty} dx \left[-\frac{\hbar^2}{2m} \frac{\partial^2 \psi}{\partial x^2} \frac{\partial \psi^*}{\partial x} + V\psi \frac{\partial \psi^*}{\partial x} + c.c. \right] \tag{291}$$
$$= \int_{-\infty}^{\infty} dx \left[-\frac{\hbar^2}{2m} \frac{\partial}{\partial x} \left(\frac{\partial \psi}{\partial x} \frac{\partial \psi^*}{\partial x} \right) + V \left(\psi \frac{\partial \psi^*}{\partial x} + \psi^* \frac{\partial \psi}{\partial x} \right) \right],$$
$$= \int_{-\infty}^{\infty} dx V \left(\psi \frac{\partial \psi^*}{\partial x} + \psi^* \frac{\partial \psi}{\partial x} \right)$$

since $\frac{\partial \psi}{\partial x} \frac{\partial \psi^*}{\partial x} = 0$ when evaluated at $x = \pm\infty$. Therefore,

$$\frac{d}{dt} \langle p \rangle = \int_{-\infty}^{\infty} dx \frac{\partial \psi^* \psi}{\partial x} V,$$
$$= -\int_{-\infty}^{\infty} dx \psi^* \frac{\partial V}{\partial x} \psi. \tag{292}$$

We now consider ψ expanded as a superposition of time-independent states ϕ_j, as follows:

$$\psi(t) = \sum_k c_k(t) \phi_k. \tag{293}$$

The expansion coefficients $c_k(t)$ evolve in time, as determined by Eq. (285), as follows:

$$\sum_k \dot{c}_k(t) \phi_k = \frac{d}{dt} \psi(t),$$
$$= -\frac{i}{\hbar} \sum_k c_k(t) \hat{H} \phi_k, \tag{294}$$

70

giving

$$\dot{c}_j(t) = -\frac{i}{\hbar} \sum_k c_k(t) \langle \phi_j | \hat{H} | \phi_k \rangle,$$
$$= -\frac{i}{\hbar} \sum_k c_k(t) H_{jk}, \tag{295}$$

with $H_{jk} = \langle \phi_j | \hat{H} | \phi_k \rangle$. Writing $c(j)$ in terms of its real and imaginary parts, we obtain:

$$c(j) = \sqrt{n_j} e^{iq_j},$$
$$= \sqrt{n_j} \cos(q_j) + i \sqrt{n_j} \sin(q_j), \tag{296}$$
$$= \frac{1}{\sqrt{2\hbar}} (x_j + ip_j),$$

and substituting Eq. (296) into Eq. (295) gives

$$\dot{x}_j + i\dot{p}_j = \sum_k (-ix_k + p_k) \bar{H}_{jk}, \tag{297}$$

where $\bar{H}_{ij} = H_{ij}/\hbar$. Equating the real and imaginary parts of Eq. (297), we obtain:

$$\dot{x}_j = \sum_k p_k \bar{H}_{jk},$$
$$\dot{p}_j = -\sum_k x_k \bar{H}_{jk}. \tag{298}$$

The rest of this section shows that, according to Eq. (298), x_j and p_j evolve according to the Erhenfest dynamics, as follows:

$$\dot{x}_j = \frac{\partial}{\partial p_j} \langle H \rangle,$$
$$\dot{p}_j = -\frac{\partial}{\partial x_j} \langle H \rangle. \tag{299}$$

71

We consider the expectation value of the Hamiltonian,

$$
\begin{aligned}
\langle H \rangle &= \langle \psi | \hat{H} | \psi \rangle, \\
&= \sum_j \sum_k c_j^* c_k H_{jk} \\
&= \frac{1}{2} \sum_j \sum_k (x_j - ip_j)(x_k + ip_k) \bar{H}_{jk} \\
&= \frac{1}{2} \sum_j \sum_k (x_j x_k + ix_j p_k - ip_j x_k + p_j p_k) \bar{H}_{jk} \\
&= \frac{1}{2} \sum_j \sum_k (x_j x_k + p_j p_k) \bar{H}_{jk} + \frac{i}{2} \sum_j \sum_k (x_j p_k - p_j x_k) \bar{H}_{jk} \\
&= \frac{1}{2} \sum_j \sum_k (x_j x_k + p_j p_k) \bar{H}_{jk} + \frac{i}{2} \sum_j \sum_k x_j p_k \bar{H}_{jk} - \frac{i}{2} \sum_j \sum_k p_j x_k \bar{H}_{jk} \\
&= \frac{1}{2} \sum_j \sum_k (x_j x_k + p_j p_k) \bar{H}_{jk} + \frac{i}{2} \sum_j \sum_k x_j p_k \bar{H}_{jk} - \frac{i}{2} \sum_j \sum_k p_k x_j \bar{H}_{kj}
\end{aligned}
\tag{300}
$$

When H_{jk} are real, $\bar{H}_{jk} = \bar{H}_{kj}$, so

$$
\begin{aligned}
\langle H \rangle &= \frac{1}{2} \sum_j \sum_k (x_j x_k + p_j p_k) \bar{H}_{jk}. \\
&= \sum_j \sum_k \sqrt{n_j} \sqrt{n_k} e^{i(q_k - q_j)} H_{jk},
\end{aligned}
\tag{301}
$$

which is often called the 'classical' Hamiltonian representing coupled harmonic oscillators with coordinates and momenta x_j and p_j evolving according to Hamilton's equations of motion:

$$
\begin{aligned}
\frac{\partial}{\partial p_i} \langle H \rangle &= \frac{1}{2} \sum_k p_k \bar{H}_{ik} + \frac{1}{2} \sum_j p_j \bar{H}_{ji}, \\
&= \sum_k p_k \bar{H}_{ik}, \\
&= \dot{x}_i, \\
-\frac{\partial}{\partial x_i} \langle H \rangle &= -\frac{1}{2} \sum_k x_k \bar{H}_{ik} - \frac{1}{2} \sum_j x_j \bar{H}_{ji}, \\
&= -\sum_k x_k \bar{H}_{ik}, \\
&= \dot{p}_i.
\end{aligned}
\tag{302}
$$

Analogously, classical equations of motion can be obtained for the variables n_j and q_j, which evolve according to the 'classical Hamiltonian' introduced by Eq. (301), following Hamilton's

equations:

$$\dot{n}_j = -\frac{\partial \langle H \rangle}{\partial q_j},$$
$$\dot{q}_j = \frac{\partial \langle H \rangle}{\partial n_j}. \tag{303}$$

18.1 Classical Electron Analog Hamiltonian

The *Meyer-Miller classical electron analog Hamiltonian* H^{MM} is essentially the 'classical Hamiltonian' $\langle H \rangle$, introduced in the previous section, minus the 'zero-point energy' of that system of coupled harmonic oscillators with coordinates and momenta x_j and p_j:

$$H^{MM}(\tilde{\mathbf{x}}, \tilde{\mathbf{p}}) = \frac{1}{2} \sum_j \sum_k (\tilde{x}_j \tilde{x}_k + \tilde{p}_j \tilde{p}_k) H_{jk} - \frac{1}{2} \sum_k H_{kk} \tag{304}$$

The Meyer-Miller Hamiltonian, introduced by Eq. (304), can be obtained by first writing the Hamiltonian in second quantization, as described in Sec. 47.4,

$$\hat{H} = \sum_{j=1}^{N} \sum_{k=1}^{N} H_{jk} |j\rangle\langle k|,$$
$$= \sum_{j=1}^{N} \sum_{k=1}^{N} H_{jk} \hat{a}_j^\dagger \hat{a}_k, \tag{305}$$

and then substituting the creation and anihilation operators of the harmonic oscillator, as follows:

$$\hat{H} = \frac{1}{2} \sum_{j=1}^{N} \sum_{k=1}^{N} H_{jk} (\tilde{x}_j - i\tilde{p}_j)(\tilde{x}_k + i\tilde{p}_k),$$
$$= \frac{1}{2} \sum_{j=1}^{N} \sum_{k=1}^{N} H_{jk} (\tilde{x}_j \tilde{x}_k + \tilde{p}_j \tilde{p}_k - \delta_{jk}). \tag{306}$$

Note that the delta function, in Eq. (306), is introduced by the commutator $[\tilde{x}_j, \tilde{p}_k] = i\delta_{jk}$. Therefore, the 'zero-point energy' term effectively introduces the commutation relation between the coordinates and momenta,

$$\tilde{x}_j = \frac{1}{\sqrt{2}} [\hat{a}_j^\dagger + \hat{a}_j],$$
$$\tilde{p}_j = \frac{i}{\sqrt{2}} [\hat{a}_j^\dagger - \hat{a}_j], \tag{307}$$

with

$$\tilde{x}_j = \hat{x}_j \sqrt{\frac{m\omega}{\hbar}},$$

$$\tilde{p}_j = \hat{p}_j \sqrt{\frac{1}{m\omega\hbar}}, \tag{308}$$

and $[\hat{x}_j, \hat{p}_j] = i\hbar$, corresponding to the single harmonic oscillator Hamiltonians,

$$
\begin{aligned}
H_j &= \frac{\hat{p}_j^2}{2m} + \frac{1}{2}m\omega^2\hat{x}_j^2, \\
&= \frac{\tilde{p}_j^2}{2m}m\omega\hbar + \frac{1}{2}m\omega^2\frac{\hbar}{m\omega}\tilde{x}_j^2, \\
&= \frac{\hbar\omega}{2}\left[\tilde{p}_j^2 + \tilde{x}_j^2\right].
\end{aligned}
\tag{309}
$$

Considering that

$$
\begin{aligned}
\hat{n}_j &= \hat{a}_j^\dagger \hat{a}_j, \\
&= \frac{1}{2}(\tilde{x}_j - i\tilde{p}_j)(\tilde{x}_j + i\tilde{p}_j), \\
&= \frac{1}{2}(\tilde{x}_j^2 + i[\tilde{x}_j, \tilde{p}_j] + \tilde{p}_j^2), \\
&= \frac{1}{2}(\tilde{x}_j^2 + \tilde{p}_j^2 - 1),
\end{aligned}
\tag{310}
$$

we obtain the usual expression of the Hamiltonian in terms of the occupation number,

$$H_j = \hbar\omega\left(\hat{n}_j + \frac{1}{2}\right), \tag{311}$$

with eigenstates

$$
\begin{aligned}
|0\rangle &= \left(\frac{m\omega}{\pi\hbar}\right)^{1/4}\exp(-\frac{m\omega}{2\hbar}x_j^2), \\
|\nu+1\rangle &= \frac{1}{\sqrt{\nu+1}}\hat{a}_j^\dagger|\nu\rangle, \\
|1\rangle &= \frac{1}{\sqrt{2}}\left(\hat{x}_j\sqrt{\frac{m\omega}{\hbar}} - i\frac{1}{\sqrt{m\omega\hbar}}\hat{p}_j\right)\left(\frac{m\omega}{\pi\hbar}\right)^{1/4}\exp(-\frac{m\omega}{2\hbar}x_j^2), \\
|1\rangle &= \frac{1}{\sqrt{2}}\left(\hat{x}_j\sqrt{\frac{m\omega}{\hbar}} - \hbar\frac{1}{\sqrt{m\omega\hbar}}\frac{\partial}{\partial x_j}\right)\left(\frac{m\omega}{\pi\hbar}\right)^{1/4}\exp(-\frac{m\omega}{2\hbar}x_j^2), \\
|1\rangle &= \frac{1}{\sqrt{2}}\left(\hat{x}_j\sqrt{\frac{m\omega}{\hbar}} + \frac{m\omega}{\hbar}\hbar\frac{x_j}{\sqrt{m\omega\hbar}}\right)\left(\frac{m\omega}{\pi\hbar}\right)^{1/4}\exp(-\frac{m\omega}{2\hbar}x_j^2), \\
|1\rangle &= \frac{1}{\sqrt{2}}\sqrt{\frac{m\omega}{\hbar}}2\hat{x}_j\left(\frac{m\omega}{\pi\hbar}\right)^{1/4}\exp(-\frac{m\omega}{2\hbar}x_j^2),
\end{aligned}
\tag{312}
$$

with matrix elements $\langle\mu|a_j^\dagger|\nu\rangle = \sqrt{\nu}\delta_{\mu,\nu+1}$ and $\langle\mu|a_j|\nu\rangle = \langle\nu|a_j^\dagger|\mu\rangle$.

The second quantization mapping $|j\rangle\langle k| \to \hat{a}_j^\dagger\hat{a}_k$, introduced in this section, is often known as the *Jordan-Schwinger representation* [P. Jordan, Z. Phys. 94, 531 (1935); J. Schwinger, in Quantum Theory of Angular Momentum, edited by L. C. Biedenharn and H. V. Dam Academic, New York, 1965], a transformation of a Hamiltonian from the basis of angular momentum into the basis of the Harmonic oscillator.

18.2   Wigner Transform Propagation based on the MM Hamiltonian

The equation of motion for propagation of the Wigner transform

$$\rho^W(\mathbf{x},\mathbf{p}) = \int \frac{ds_1 e^{is_1 p_1}}{2\pi}\cdots\int \frac{ds_N e^{is_N p_N}}{2\pi}\psi^*(\mathbf{x}+\mathbf{s}/2)\psi(\mathbf{x}-\mathbf{s}/2), \tag{313}$$

according the Meyer-Miller Hamiltonian,

$$\begin{aligned}
\hat{H} &= \frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N}H_{jk}(\tilde{x}_j\tilde{x}_k + \tilde{p}_j\tilde{p}_k - \delta_{jk}), \\
&= \frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N}\hbar H_{jk}(x_j x_k + p_j p_k - \delta_{jk})
\end{aligned} \tag{314}$$

can be obtained as follows:

$$\frac{\partial\rho^W}{\partial t} = \int \frac{ds_1 e^{is_1 p_1}}{2\pi}\cdots\int \frac{ds_N e^{is_N p_N}}{2\pi}\left(\frac{\partial\psi^*(\mathbf{x}+\mathbf{s}/2)}{\partial t}\psi(\mathbf{x}-\mathbf{s}/2) + \psi^*(\mathbf{x}+\mathbf{s}/2)\frac{\partial\psi(\mathbf{x}-\mathbf{s}/2)}{\partial t}\right). \tag{315}$$

Considering that

$$\begin{aligned}
i\psi(\mathbf{x}-\mathbf{s}/2)\frac{\partial\psi^*(\mathbf{x}+\mathbf{s}/2)}{\partial t} =& -\frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N}H_{jk}(x_j x_k + p_j p_k - \delta_{jk})\psi^*\psi \\
&- \sum_{i=1}^{N}\frac{s_i}{2}\frac{\partial}{\partial x_i}\frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N}H_{jk}(x_j x_k - \delta_{jk})\psi^*\psi \\
&- \sum_{i=1}^{N}\sum_{l=1}^{N}\frac{s_i}{2}\frac{s_l}{2}\frac{\partial}{\partial x_l}\frac{\partial}{\partial x_i}\frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N}H_{jk}(x_j x_k - \delta_{jk})\psi^*\psi
\end{aligned} \tag{316}$$

and

$$i\psi^*(\mathbf{x} + \mathbf{s}/2)\frac{\partial\psi(\mathbf{x} - \mathbf{s}/2)}{\partial t} = \frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N}H_{jk}(x_jx_k + p_jp_k - \delta_{jk})\psi\psi^*$$

$$-\sum_{i=1}^{N}\frac{s_i}{2}\frac{\partial}{\partial x_i}\frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N}H_{jk}(x_jx_k - \delta_{jk})\psi\psi^* \tag{317}$$

$$+\sum_{i=1}^{N}\sum_{l=1}^{N}\frac{s_i}{2}\frac{s_l}{2}\frac{\partial}{\partial x_l}\frac{\partial}{\partial x_i}\frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N}H_{jk}(x_jx_k - \delta_{jk})\psi\psi^*$$

gives

$$\frac{\partial\rho^W}{\partial t} = \int\frac{ds_1 e^{is_1p_1}}{2\pi}\cdots\int\frac{ds_N e^{is_Np_N}}{2\pi}i\sum_{j=1}^{N}\sum_{k=1}^{N}\frac{H_{jk}}{2}\left[\psi p_jp_k\psi^* - \psi^*p_jp_k\psi + (s_jx_k + s_kx_j)\psi\psi^*\right],$$

$$\tag{318}$$

where

$$[\cdots] = \left[-4\left(\psi\frac{\partial}{\partial s_j}\frac{\partial}{\partial s_k}\psi^* - \psi^*\frac{\partial}{\partial s_j}\frac{\partial}{\partial s_k}\psi\right) + (s_jx_k + s_kx_j)\psi\psi^*\right],$$

$$= \left[-4\frac{\partial}{\partial s_j}\left(\psi\frac{\partial}{\partial s_k}\psi^* - \psi^*\frac{\partial}{\partial s_k}\psi\right) + (s_jx_k + s_kx_j)\psi\psi^*\right],$$

$$= \left[-2\frac{\partial}{\partial s_j}\left(\psi\frac{\partial}{\partial x_k}\psi^* + \psi^*\frac{\partial}{\partial x_k}\psi\right) + (s_jx_k + s_kx_j)\psi\psi^*\right], \tag{319}$$

$$= \left[-2\frac{\partial}{\partial s_j}\frac{\partial}{\partial x_k}(\psi\psi^*) + (s_jx_k + s_kx_j)\psi\psi^*\right],$$

$$= \left[-2p_j\frac{\partial}{\partial x_k}\rho^W + (\frac{\partial}{\partial p_j}x_k + \frac{\partial}{\partial p_k}x_j)\rho^W\right],$$

$$\frac{\partial\rho^W}{\partial t} = \sum_{j=1}^{N}\sum_{k=1}^{N}H_{jk}\left[-p_j\frac{\partial\rho^W}{\partial x_k} + \frac{\partial\rho^W}{\partial p_k}x_j\right],$$

$$= \sum_{k=1}^{N}-\frac{\partial H}{\partial p_k}\frac{\partial\rho^W}{\partial x_k} + \frac{\partial\rho^W}{\partial p_k}\frac{\partial H}{\partial x_k}. \tag{320}$$

Equation (320) shows that classical propagation of the Wigner transform ρ^W gives a rigorous description of quantum nonadiabatic dynamics since the Hamiltonian is quadratic in coordinates and momenta, x_j and p_j.

The survival probability $P_k(t) = \langle\Psi(t)|k\rangle\langle k|\Psi(t)\rangle = c_k^*(t)c_k(t)$, corresponding to the time-evolved state $|\Psi(t)\rangle = \sum_j c_j(t)|j\rangle$, can be obtained from the time-evolved Wigner transform

$\rho^W(\mathbf{x}, \mathbf{p}; t)$, as follows:

$$P_k(t) = (2\pi\hbar)^N \int d\mathbf{p} \int d\mathbf{x} \left(\rho_{kk}^W(\mathbf{x}, \mathbf{p})\right)^* \rho^W(\mathbf{x}, \mathbf{p}; t), \tag{321}$$

where

$$\rho_{kk}^W(\mathbf{x}, \mathbf{p}) = \frac{1}{(2\pi\hbar)^N} \int d\mathbf{s}' e^{\frac{i}{\hbar}\mathbf{P}\cdot\mathbf{s}'} \langle\mathbf{x} - \frac{\mathbf{s}'}{2}|k\rangle\langle k|\mathbf{x} + \frac{\mathbf{s}'}{2}\rangle,$$

$$\rho^W(\mathbf{x}, \mathbf{p}; t) = \frac{1}{(2\pi\hbar)^N} \int d\mathbf{s} e^{\frac{i}{\hbar}\mathbf{P}\cdot\mathbf{s}} \langle\mathbf{x} - \frac{\mathbf{s}}{2}|\Psi(t)\rangle\langle\Psi(t)|\mathbf{x} + \frac{\mathbf{s}}{2}\rangle, \tag{322}$$

$$= \frac{1}{(2\pi\hbar)^N} \sum_j \sum_{j'} c_j^* c_{j'} \int d\mathbf{s} e^{\frac{i}{\hbar}\mathbf{P}\cdot\mathbf{s}} \langle\mathbf{x} - \frac{\mathbf{s}}{2}|j'\rangle\langle j|\mathbf{x} + \frac{\mathbf{s}}{2}\rangle.$$

We note that

$$\rho_{00}^W = \frac{1}{(2\pi\hbar)^N} \int d\mathbf{s} e^{\frac{i}{\hbar}\mathbf{P}\cdot\mathbf{s}} \langle\mathbf{x} - \frac{\mathbf{s}}{2}|0\rangle\langle 0|\mathbf{x} + \frac{\mathbf{s}}{2}\rangle, \tag{323}$$

represents a stationary (vacuum) ground state, with no quantum of excitation in any harmonic oscillator, while state k is represented, as follows:

$$
\begin{aligned}
\rho_{kk}^W &= \frac{1}{(2\pi\hbar)^N} \int d\mathbf{s} e^{\frac{i}{\hbar}\mathbf{P}\cdot\mathbf{s}} \langle\mathbf{x} - \frac{\mathbf{s}}{2}|\hat{a}_k^\dagger|0\rangle\langle 0|\hat{a}_k|\mathbf{x} + \frac{\mathbf{s}}{2}\rangle, \\
&= \frac{1}{(2\pi\hbar)^N} \int d\mathbf{s} e^{\frac{i}{\hbar}\mathbf{P}\cdot\mathbf{s}} \frac{2}{\hbar} \left(\frac{1}{\pi\hbar}\right)^{N/2} (x_k - \frac{s_k}{2})(x_k + \frac{s_k}{2}) e^{-\frac{1}{2\hbar}\left((\mathbf{x}+\frac{\mathbf{s}}{2})^2 + (\mathbf{x}-\frac{\mathbf{s}}{2})^2\right)}, \\
&= \frac{2}{\hbar} \left(\frac{1}{\pi\hbar}\right)^{N/2} \frac{1}{(2\pi\hbar)^N} \left[x_k^2 e^{-\frac{1}{\hbar}\mathbf{x}^2} \int d\mathbf{s} e^{\frac{i}{\hbar}\mathbf{P}\cdot\mathbf{s}} e^{-\frac{\mathbf{s}^2}{4\hbar}} - e^{-\frac{1}{\hbar}\mathbf{x}^2} \frac{1}{4} \int d\mathbf{s} e^{\frac{i}{\hbar}\mathbf{P}\cdot\mathbf{s}} s_k^2 e^{-\frac{\mathbf{s}^2}{4\hbar}}\right], \\
&= \frac{2}{\hbar} \left(\frac{1}{\pi\hbar}\right)^{N/2} \frac{1}{(2\pi\hbar)^N} \left[x_k^2 e^{-\frac{1}{\hbar}\mathbf{x}^2} (\pi 4\hbar)^{N/2} e^{-\frac{\mathbf{p}^2}{\hbar}} + e^{-\frac{1}{\hbar}\mathbf{x}^2} \frac{\hbar^2}{4} \frac{\partial^2}{\partial p_k^2} \int d\mathbf{s} e^{\frac{i}{\hbar}\mathbf{P}\cdot\mathbf{s}} e^{-\frac{\mathbf{s}^2}{4\hbar}}\right], \\
&= \frac{2}{\hbar} \left(\frac{1}{\pi\hbar}\right)^{N/2} \frac{(\pi 4\hbar)^{N/2}}{(2\pi\hbar)^N} e^{-\frac{1}{\hbar}\mathbf{x}^2} \left[x_k^2 e^{-\frac{\mathbf{p}^2}{\hbar}} + \frac{\hbar^2}{4} \frac{\partial^2}{\partial p_k^2} e^{-\frac{\mathbf{p}^2}{\hbar}}\right], \\
&= \frac{2}{\hbar} \left(\frac{1}{\pi\hbar}\right)^{N/2} \frac{(\pi 4\hbar)^{N/2}}{(2\pi\hbar)^N} e^{-\frac{1}{\hbar}\mathbf{x}^2} \left[x_k^2 e^{-\frac{\mathbf{p}^2}{\hbar}} + \frac{\hbar^2}{4} \frac{\partial}{\partial p_k} \left(-\frac{2p_k}{\hbar} e^{-\frac{\mathbf{p}^2}{\hbar}}\right)\right], \\
&= \frac{2}{\hbar} \left(\frac{1}{\pi\hbar}\right)^{N/2} \frac{(\pi 4\hbar)^{N/2}}{(2\pi\hbar)^N} e^{-\frac{1}{\hbar}\mathbf{x}^2} \left[x_k^2 e^{-\frac{\mathbf{p}^2}{\hbar}} + \frac{\hbar^2}{4} \left(\left(-\frac{2p_k}{\hbar}\right)^2 - \frac{2}{\hbar}\right) e^{-\frac{\mathbf{p}^2}{\hbar}}\right], \\
&= \frac{2}{\hbar} \frac{1}{(\pi\hbar)^N} e^{-\frac{1}{\hbar}(\mathbf{x}^2 + \mathbf{p}^2)} \left[x_k^2 + p_k^2 - \frac{\hbar}{2}\right],
\end{aligned}
\tag{324}
$$

Note that by substituting Eq. (322) into Eq. (321), we obtain:

$$P_k(t) = (2\pi\hbar)^N \sum_j \sum_{j'} c_j^* c_{j'} \frac{1}{(2\pi\hbar)^N} \int d\mathbf{s} \int d\mathbf{p} \int d\mathbf{x} \frac{1}{(2\pi\hbar)^N} \int d\mathbf{s}' e^{\frac{i}{\hbar}\mathbf{P}(\mathbf{s}-\mathbf{s}')}$$

$$\times \langle\mathbf{x} + \frac{\mathbf{s}'}{2}|k\rangle\langle k|\mathbf{x} - \frac{\mathbf{s}'}{2}\rangle\langle\mathbf{x} - \frac{\mathbf{s}}{2}|j'\rangle\langle j|\mathbf{x} + \frac{\mathbf{s}}{2}\rangle. \tag{325}$$

Integrating over $\mathbf{p}$, we obtain:

$$P_k(t) = \sum_j \sum_{j'} c_j^* c_{j'} \int d\mathbf{s} \int d\mathbf{x} \int d\mathbf{s}' \delta(\mathbf{s} - \mathbf{s}')\langle \mathbf{x} + \frac{\mathbf{s}'}{2}|k\rangle\langle k|\mathbf{x} - \frac{\mathbf{s}'}{2}\rangle\langle \mathbf{x} - \frac{\mathbf{s}}{2}|j'\rangle\langle j|\mathbf{x} + \frac{\mathbf{s}}{2}\rangle,$$

$$= \sum_j \sum_{j'} c_j^* c_{j'} \int d\mathbf{s} \int d\mathbf{x}\langle \mathbf{x} + \frac{\mathbf{s}}{2}|k\rangle\langle k|\mathbf{x} - \frac{\mathbf{s}}{2}\rangle\langle \mathbf{x} - \frac{\mathbf{s}}{2}|j'\rangle\langle j|\mathbf{x} + \frac{\mathbf{s}}{2}\rangle. \tag{326}$$

Introducing the substitution $\mathbf{r} = \mathbf{x} - \mathbf{s}/2$, with $d\mathbf{r} = d\mathbf{x}$, we obtain:

$$P_k(t) = \sum_j \sum_{j'} c_j^* c_{j'} \int d\mathbf{s} \int d\mathbf{r}\langle \mathbf{r} + \mathbf{s}|k\rangle\langle k|\mathbf{r}\rangle\langle \mathbf{r}|j'\rangle\langle j|\mathbf{r} + \mathbf{s}\rangle, \tag{327}$$

and introducing the substitution $\mathbf{r}' = \mathbf{r} + \mathbf{s}$, with $d\mathbf{r}' = d\mathbf{s}$, we obtain:

$$P_k(t) = \sum_j \sum_{j'} c_j^* c_{j'} \int d\mathbf{r}' \int d\mathbf{r}\langle \mathbf{r}'|k\rangle\langle k|\mathbf{r}\rangle\langle \mathbf{r}|j'\rangle\langle j|\mathbf{r}'\rangle,$$

$$= \sum_j \sum_{j'} c_j^* c_{j'} \int d\mathbf{r}\langle k|\mathbf{r}\rangle\langle \mathbf{r}|j'\rangle \int d\mathbf{r}'\langle j|\mathbf{r}'\rangle\langle \mathbf{r}'|k\rangle,$$

$$= \sum_j \sum_{j'} c_j^* c_{j'}\langle k|j'\rangle\langle j|k\rangle, \tag{328}$$

$$= \sum_j \sum_{j'} c_j^* c_{j'}\delta_{k,j'}\delta_{j,k},$$

$$= c_k^* c_k,$$

which proves Eq. (321).

18.3 Computational Problem 2-level WT

Write a program to propagate the Wigner transform of a 2-level system, described to the MM Hamiltonian introduced by Eq. (314), with $H_{11} = -H_{22} = H_{12} = H_{21} = 1.0$, initialized in one of the 2 states according to Eq. (324) and evolving by Velocity Verlet according to Eq. (320). Compare the Rabi oscillations of the time-dependent survival probability to the corresponding results obtained by SOFT quantum propagation.

Solution in Sec. 61.26

Next, we consider the propagation of the Wigner transform $\rho^W(\mathbf{x}, \mathbf{R}, \mathbf{p}, \mathbf{P})$, evolving according to the Hamiltonian

$$H(\mathbf{x}, \mathbf{R}, \mathbf{p}, \mathbf{P}) = \frac{\mathbf{P}^2}{2M} + \frac{1}{2}\sum_{l=1}^{N}\sum_{k=1}^{N} V_{lk}(\mathbf{R})(x_l x_k + p_l p_k - \delta_{lk}). \tag{329}$$

78

For harmonic V_{jk}, as in the model Hamiltonian of pyrazine, we obtain:

$$
\begin{aligned}
i\frac{\partial \psi^*(\mathbf{x}+\mathbf{s}/2)}{\partial t} =& -\sum_j \frac{P_j^2}{2M_j}\psi^* - \frac{1}{2}\sum_{j=1}^N\sum_{k=1}^N V_{jk}(x_j x_k + p_j p_k - \delta_{jk})\psi^* \\
& -\sum_{i=1}^N \frac{s_i}{2}\frac{\partial}{\partial x_i}\frac{1}{2}\sum_{j=1}^N\sum_{k=1}^N V_{jk}x_j x_k \psi^* \\
& -\sum_i \frac{S_i}{2}\frac{1}{2}\sum_{j=1}^N\sum_{k=1}^N \frac{\partial V_{jk}}{\partial R_i}(x_j x_k + p_j p_k - \delta_{jk})\psi^* \\
& -\frac{1}{2!}\sum_{i=1}^N\sum_{l=1}^N \frac{s_i}{2}\frac{s_l}{2}\frac{\partial}{\partial x_l}\frac{\partial}{\partial x_i}\frac{1}{2}\sum_{j=1}^N\sum_{k=1}^N V_{jk}x_j x_k \psi^* \\
& -\frac{1}{2!}\sum_{i=1}^N\sum_{k'} \frac{s_i}{2}\frac{S_{k'}}{2}\frac{1}{2}\frac{\partial}{\partial x_i}\sum_{j=1}^N\sum_{k=1}^N \frac{\partial V_{jk}}{\partial R_{k'}}x_j x_k \psi^* \\
& -\frac{1}{2!}\sum_{j'}\sum_{k'} \frac{S_{j'}}{2}\frac{S_{k'}}{2}\frac{1}{2}\sum_{j=1}^N\sum_{k=1}^N \frac{\partial^2 V_{jk}}{\partial R_{j'}\partial R_{k'}}(x_j x_k + p_j p_k - \delta_{jk})\psi^* \\
& -\frac{1}{3!}\sum_{j'} \frac{S_{j'}}{2}\sum_{i=1}^N\sum_{l=1}^N \frac{s_i}{2}\frac{s_l}{2}\frac{\partial}{\partial x_l}\frac{\partial}{\partial x_i}\frac{1}{2}\sum_{j=1}^N\sum_{k=1}^N \frac{\partial V_{jk}}{\partial R_{j'}}x_j x_k \psi^* \\
& -\frac{1}{3!}\sum_{j'}\sum_{k'} \frac{S_{k'}}{2}\frac{S_{j'}}{2}\sum_{i=1}^N \frac{s_i}{2}\frac{\partial}{\partial x_i}\frac{1}{2}\sum_{j=1}^N\sum_{k=1}^N \frac{\partial^2 V_{jk}}{\partial R_{j'}\partial R_{k'}}x_j x_k \psi^*
\end{aligned}
\tag{330}
$$

which gives

$$i\frac{\partial}{\partial t}\psi^*(\mathbf{x}+\mathbf{s}/2)\psi(\mathbf{x}-\mathbf{s}/2) = \sum_j \psi^* \frac{P_j^2}{2M_j}\psi - \psi\frac{P_j^2}{2M_j}\psi^* - \frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N}V_{jk}(\psi p_j p_k \psi^* - \psi^* p_j p_k \psi)$$

$$-\frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_i \frac{S_i}{2}\frac{\partial V_{jk}}{\partial R_i}(\psi p_j p_k \psi^* + \psi^* p_j p_k \psi)$$

$$-\frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N}\frac{1}{2!}\sum_{j'}\sum_{k'}\frac{S_{j'}}{2}\frac{S_{k'}}{2}\frac{\partial^2 V_{jk}}{\partial R_{j'}\partial R_{k'}}(\psi p_j p_k \psi^* - \psi^* p_j p_k \psi)$$

$$-\sum_{i=1}^{N}s_i\sum_{j=1}^{N}V_{ji}x_j\psi^*\psi$$

$$-\sum_i S_i\frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N}\frac{\partial V_{jk}}{\partial R_i}(x_j x_k - \delta_{jk})\psi^*\psi$$

$$-\frac{1}{3!}\sum_{j'}S_{j'}\sum_{i=1}^{N}\sum_{l=1}^{N}\frac{s_i}{2}\frac{s_l}{2}\frac{\partial V_{il}}{\partial R_{j'}}\psi^*\psi$$

$$-\frac{1}{3!}\sum_{j'}\sum_{k'}S_{k'}\frac{S_{j'}}{2}\sum_{i=1}^{N}\frac{s_i}{2}\sum_{k=1}^{N}\frac{\partial^2 V_{ik}}{\partial R_{j'}\partial R_{k'}}x_k\psi^*\psi$$

$$(331)$$

or

$$i\frac{\partial}{\partial t}\psi^*(\mathbf{x}+\mathbf{s}/2)\psi(\mathbf{x}-\mathbf{s}/2) = -\sum_j \frac{P_j}{M_j}\frac{\partial}{\partial R_j}(\psi^*\psi) - \frac{1}{2}\sum_{j=1}^{N}\frac{\partial}{\partial x_j}\sum_{k=1}^{N}V_{jk}(-2)\frac{\partial}{\partial s_k}(\psi^*\psi)$$

$$-\frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{i}\frac{S_i}{2}\frac{\partial V_{jk}}{\partial R_i}(\psi p_j p_k \psi^* + \psi^* p_j p_k \psi)$$

$$-\frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N}\frac{1}{2!}\sum_{j'}\sum_{k'}\frac{S_{j'}}{2}\frac{S_{k'}}{2}\frac{\partial^2 V_{jk}}{\partial R_{j'}\partial R_{k'}}(\psi p_j p_k \psi^* - \psi^* p_j p_k \psi)$$

$$-\sum_{i=1}^{N}s_i\sum_{j=1}^{N}V_{ji}x_j\psi^*\psi$$

$$-\sum_i S_i\frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N}\frac{\partial V_{jk}}{\partial R_i}(x_j x_k - \delta_{jk})\psi^*\psi$$

$$-\frac{1}{3!}\sum_{j'}S_{j'}\sum_{i=1}^{N}\sum_{l=1}^{N}\frac{s_i}{2}\frac{s_l}{2}\frac{\partial V_{il}}{\partial R_{j'}}\psi^*\psi$$

$$-\frac{1}{3!}\sum_{j'}\sum_{k'}S_{k'}\frac{S_{j'}}{2}\sum_{i=1}^{N}\frac{s_i}{2}\sum_{k=1}^{N}\frac{\partial^2 V_{ik}}{\partial R_{j'}\partial R_{k'}}x_k\psi^*\psi$$

$$\text{(332)}$$

giving

$$\frac{\partial \rho^W}{\partial t} = -\sum_j \frac{\partial \rho^W}{\partial R_j}\frac{P_j}{M_j} - \sum_{j=1}^{N}\frac{\partial \rho^W}{\partial x_j}\sum_{k=1}^{N}p_k V_{jk}$$

$$+\frac{1}{2}\sum_i \frac{\partial \rho^W}{\partial P_i}\sum_{j=1}^{N}\sum_{k=1}^{N}\frac{\partial V_{jk}}{\partial R_i}(x_j x_k - \delta_{jk}) + \sum_{j=1}^{N}\frac{\partial \rho^W}{\partial p_j}\sum_{k=1}^{N}x_k V_{kj}$$

$$+\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_i \frac{\partial}{\partial P_i}\frac{1}{2}\frac{\partial V_{jk}}{\partial R_i}\frac{1}{2\pi}\int d\mathbf{s}e^{i\mathbf{p}\cdot\mathbf{s}}(\psi\frac{\partial}{\partial x_j}\frac{\partial}{\partial s_k}\psi^* - \psi^*\frac{\partial}{\partial x_j}\frac{\partial}{\partial s_k}\psi)$$

$$+\frac{1}{8}\sum_{j'}\sum_{k'}\sum_{k=1}^{N}\frac{\partial^3 \rho^W}{\partial x_k \partial P_{j'} P_{k'}}\sum_{j=1}^{N}p_j \frac{\partial^2 V_{jk}}{\partial R_{j'}\partial R_{k'}}$$

$$-\frac{1}{24}\sum_{j'}\sum_{i=1}^{N}\sum_{k=1}^{N}\frac{\partial^3 \rho^W}{\partial P_{j'}\partial p_i \partial p_k}\frac{\partial V_{ik}}{\partial R_{j'}}$$

$$-\frac{1}{24}\sum_{j'}\sum_{k'}\sum_{i=1}^{N}\frac{\partial^3 \rho^W}{\partial P_{k'}\partial P_{j'}\partial p_i}\sum_{k=1}^{N}\frac{\partial^2 V_{ik}}{\partial R_{j'}\partial R_{k'}}x_k$$

$$\text{(333)}$$

81

19   Discrete Variable Representation

The goal of this section is to introduce a generic discrete variable representation (DVR) method, introduced by Colbert and Miller [*J. Chem. Phys.* (1992) **96**:1982-1991] to solve the time-independent Schrödinger equation,

$$HC_j - C_j E_j = 0. \tag{334}$$

The method obtains the eigenstates $\chi_j(x)$ in a grid-based representation: $\chi(x) = \sum_j C_j \delta(x - x_j)$ and the corresponding eigenvalues E_j by simple diagonalization of the Hamiltonian matrix H by using standard numerical diagonalization methods –*e.g.*, TRED2, TQLI and EIGSRT, as described in Numerical Recipes (Ch. 11, Numerical Recipes), or Lanczos-type (iterative linear algebra methods) that exploit the sparsity of H. The representation is based on delta functions $\delta(x - x_j)$, equally spaced at coordinates x_j as follows:

$$x_j = x_{min} + j\Delta, \ \ \text{with} \ \ \Delta = (x_{max} - x_{min})/N, \tag{335}$$

with $j = 1$–N.

The rest of this section shows that the Hamiltonian matrix elements can be written in such a discrete (grid-based) representation, as follows:

$$H(j, j') = V(x_j)\delta_{jj'} + \frac{\hbar^2}{2m\Delta^2}(-1)^{j-j'}\left(\delta_{jj'}\frac{\pi^2}{3} + (1 - \delta_{jj'})\frac{2}{(j-j')^2}\right), \tag{336}$$

when the delta functions $\delta(x - x_j)$ are placed on a grid $x_j = j * \Delta$ that extends over the interval $x = (-\infty, \infty)$ with $j = 1, 2, ...$ Furthermore, we show that for the particular case of a radial coordinate, defined in the interval $x = (0, \infty)$, the Hamiltonian matrix elements are:

$$H(j, j') = V(x_j)\delta_{jj'} + \frac{\hbar^2}{2m\Delta^2}(-1)^{j-j'}\left(\delta_{jj'}\left(\frac{\pi^2}{3} - \frac{1}{2j^2}\right) + (1 - \delta_{jj'})\left(\frac{2}{(j-j')^2} - \frac{2}{(j+j')^2}\right)\right). \tag{337}$$

To derive Eq. (336) and Eq. (337), we consider the Hamiltonian,

$$\hat{H} = \hat{T} + V(\hat{x}), \tag{338}$$

where $V(\hat{x})$ and $\hat{T} = \frac{\hat{p}^2}{2m}$ are the potential energy and kinetic energy operators, respectively. The potential energy matrix $V^{(\delta)}$ is diagonal, with matrix elements defined as follows:

$$V^{(\delta)}(j, k) = \langle j|V(\hat{x})|k\rangle = \int dx \delta^*(x - x_j)V(\hat{x})\delta(x - x_k),$$
$$= V(x_k)\delta_{j,k}. \tag{339}$$

The kinetic energy matrix $T^{(\delta)}$ is expressed in the same grid-based representation, by first obtaining the kinetic energy matrix $T^{(\phi)}$ in the representation of eigenstates $\phi_n(x)$ of the particle in the box

$x = (x_{min}, x_{max})$, and then rotating $T^{(\phi)}$ to the representation of delta functions by using the following similarity transformation:

$$T^{(\delta)} = \Gamma^{-1} T^{(\phi)} \Gamma, \tag{340}$$

where Γ is the transformation matrix defined by the linear combinations,

$$\phi_k(x) = \sum_j \Gamma(j, k) \delta(x - x_j) \Delta', \tag{341}$$

where

$$\Gamma(j, k) = \phi_k(x_j). \tag{342}$$

Considering that $1 = \int dx \phi_k^*(x) \phi_k(x) = (\Delta')^2 \int dx \sum_j \phi_k(x_j) \delta(x - x_j) \sum_{j'} \phi_k(x_{j'}) \delta(x - x_{j'})$ we obtain that $\Delta' = \sqrt{\Delta}$ since $1 = (\Delta')^2 / \Delta \sum_j \Delta \phi_k(x_j) \phi_k(x_j)$.

The eigenstates of the particle in the box are:

$$\phi_k(x) = \sqrt{\frac{2}{x_{max} - x_{min}}} \operatorname{Sin}\left(k \frac{\pi(x - x_{min})}{(x_{max} - x_{min})}\right), \tag{343}$$

with $\phi_k(x_{min}) = 0$ and $\phi_k(x_{max}) = 0$. Therefore,

$$\hat{T} \phi_k(x) = \frac{(\hbar \pi k)^2}{2m} \phi_k(x), \tag{344}$$

and $T^{(\phi)}$ is diagonal with matrix elements,

$$\hat{T}^{(\phi)}(j, k) = \langle \phi_j | \hat{T} | \phi_k \rangle = \frac{(\hbar k)^2}{2m} \frac{\pi^2}{(x_{max} - x_{min})^2} \delta_{jk}. \tag{345}$$

Therefore, substituting Eq. (345) and Eq. (342) into Eq. (340) we obtain,

$$T^{(\delta)}(i, i') = \sum_{j,k=1}^{N-1} \Gamma^{-1}(i, j) T^{(\phi)}(j, k) \Gamma(k, i') = \sum_{j,k=1}^{N-1} \Gamma(j, i) T^{(\phi)}(j, k) \Gamma(k, i'),$$

$$= \frac{\Delta \pi^2}{(x_{max} - x_{min})^2} \sum_{j,k=1}^{N-1} \phi_j(x_i) \frac{(\hbar k)^2}{2m} \delta_{jk} \phi_k(x_i') = \frac{\Delta \pi^2}{(x_{max} - x_{min})^2} \sum_{k=1}^{N-1} \phi_k(x_i) \frac{(\hbar k)^2}{2m} \phi_k(x_i'),$$

$$= \frac{\Delta \hbar^2 \pi^2}{2m(x_{max} - x_{min})^2} \frac{2}{(x_{max} - x_{min})} \sum_{k=1}^{N-1} k^2 \operatorname{Sin}\left(k\pi \frac{(x_i - x_{min})}{(x_{max} - x_{min})}\right) \operatorname{Sin}\left(k\pi \frac{(x_{i'} - x_{min})}{(x_{max} - x_{min})}\right). \tag{346}$$

Finally, substituting Eq. (356) into Eq. (346) we obtain:

$$T^{(\delta)}(j, j') = \frac{\hbar^2 \pi^2}{2m(x_{max} - x_{min})^2} \frac{2}{N} \sum_{k=1}^{N-1} k^2 \operatorname{Sin}\left(\frac{k\pi j}{N}\right) \operatorname{Sin}\left(\frac{k\pi j'}{N}\right). \tag{347}$$

To calculate the finite series introduced by Eq. (347) we first note that,

$$2\text{Sin}\left(\frac{k\pi j}{N}\right)\text{Sin}\left(\frac{k\pi j'}{N}\right) = \text{Cos}\left(\frac{k\pi(j-j')}{N}\right) - \text{Cos}\left(\frac{k\pi(j+j')}{N}\right),$$

$$= Re\left[\text{Exp}\left(i\frac{k\pi(j-j')}{N}\right) - \text{Exp}\left(i\frac{k\pi(j+j')}{N}\right)\right]. \tag{348}$$

so that Eq. (347) can be written as follows:

$$T^{(\delta)}(j,j') = \frac{\hbar^2\pi^2}{2m(x_{max}-x_{min})^2}\frac{2}{N}\left[Re\sum_{k=1}^{N-1}k^2\text{Exp}\left(i\frac{k\pi(j-j')}{N}\right) - Re\sum_{k=1}^{N-1}k^2\text{Exp}\left(i\frac{k\pi(j+j')}{N}\right)\right]. \tag{349}$$

Then, we consider the geometric series $S_N = \sum_{k=0}^{N-1}x^k$ and we note that $S_N - xS_N = 1 - x^N$, therefore $S_N = (1-x^N)/(1-x)$. Also, we note that

$$x\frac{\partial}{\partial x}\sum_{k=0}^{N-1}x^k = \sum_{k=0}^{N-1}kx^k,$$

$$x^2\frac{\partial^2}{\partial x^2}\sum_{k=0}^{N-1}x^k = \sum_{k=0}^{N-1}k^2x^k - \sum_{k=0}^{N-1}kx^k, \tag{350}$$

Therefore,

$$\sum_{k=1}^{N-1}k^2x^k = x^2\frac{\partial^2}{\partial x^2}\left(\frac{(1-x^N)}{(1-x)}\right) + x\frac{\partial}{\partial x}\left(\frac{(1-x^N)}{(1-x)}\right). \tag{351}$$

We evaluate the sums over k in Eq. (349) analytically to obtain:

$$T^{(\delta)}(j,j') = \frac{\hbar^2(-1)^{j-j'}}{2m(x_{max}-x_{min})^2}\frac{\pi^2}{2}\left[\frac{1}{\text{Sin}^2[\pi(j-j')/(2N)]} - \frac{1}{\text{Sin}^2[\pi(j+j')/(2N)]}\right], \tag{352}$$

for $j \neq j'$ and

$$T^{(\delta)}(j,j) = \frac{\hbar^2}{2m(x_{max}-x_{min})^2}\frac{\pi^2}{2}\left[\frac{(2N^2+1)}{3} - \frac{1}{\text{Sin}^2[\pi j/N]}\right]. \tag{353}$$

Equation (336) is obtained from Eq. (352) and Eq. (353), by taking the limit $x_{min} \to -\infty$, $x_{max} \to \infty$, at finite Δ. This requires $N \to \infty$. Furthermore, since $\Delta(j+j') = x_j + x_{j'} - 2x_{min}$ and $\Delta(j-j') = x_j - x_{j'}$, this limit implies $(j+j') \to \infty$ while $(j-j')$ remains finite.

Equation (337) is obtained from Eq. (352) and Eq. (353), by making $x_{min} = 0$, and taking the limit $x_{max} \to \infty$, at finite Δ. This requires $N \to \infty$. In this case, $\Delta(j+j') = x_j + x_{j'}$ and $\Delta(j-j') = x_j - x_{j'}$, and therefore both $(j+j')$ and $(j-j')$ remain finite.

19.1 Multidimensional DVR

The multidimensional version of the DVR is straightforward. For example, for three Cartesian degrees of freedom (x,y,z), the Hamiltonian matrix is defined, as follows:

$$H(ijk, i'j'k') = T_{ii'}\delta_{jj'}\delta_{kk'} + T_{jj'}\delta_{ii'}\delta_{kk'} + T_{kk'}\delta_{ii'}\delta_{jj'} + V(x_i, x_j, x_k)\delta_{ii'}\delta_{jj'}\delta_{kk'}. \tag{354}$$

A simple implementation of the multidimensional Hamiltonian, introduced by Eq. (354), reduces the three indices i, j, k to a single index $l = (j-1)n_k n_i + (i-1)n_k + k$ and the three indices i', j', k' to a single index $l' = (j'-1)n_k n_i + (i'-1)n_k + k'$. The three indices i, j, k can be calculated from l, by using the modulo function em mod, as follows: $k = mod(l, (n_k * n_i))$, giving the remainder after division of l by $n_k * n_i$. Analogously, $i = 1 + mod(l-k, n_k)$ and $j = 1 + abs(l - (i-1)n_k - k)$ where n_i, n_j and n_k are the number of grid points representing x, y and z, respectively.

19.2 Computational Problem 15

15.1 Write a program to solve the time independent Schrödinger equation by using the DVR method and apply it to find the first 4 eigenvalues and eigenfunctions of the Harmonic oscillator introduced by Eq. (17) with $m = 1$ and $\omega = 1$. Verify that the eigenvalues are $E(\nu) = (1/2+\nu)\hbar\omega$, $\nu = 0$–10.
15.2 Change the potential of the code written in 15.1 to that of a Morse oscillator $V(\hat{x}) = De(1 - \exp(-a(\hat{x} - x_e)))^2$, with $x_e = 0$, $De = 8$, and $a = \sqrt{k/(2D_e)}$, where $k = m\omega^2$, and recompute the eigenvalues and eigenfunctions.
15.3 Generalize the program developed in 15.1 to solve the 2-dimensional Harmonic oscillator $V(x, y) = 1/2m\omega^2(x^2 + y^2)$ and apply it to find the first 4 eigenvalues and eigenfunctions of the Harmonic oscillator introduced by Eq. (17) with $m = 1$ and $\omega = 1$. Verify that the eigenvalues are $E(\nu) = (1 + \nu_1 + \nu_2)\hbar\omega$.
15.4 Change the potential of the code written in 15.3 to that of a 2-dimensional Morse oscillator $V(\hat{x}, \hat{y}) = De(1 - \exp(-a(\hat{x} - x_e)))^2 + De(1 - \exp(-a(\hat{y} - x_e)))^2$, with $x_e = 0$, $De = 8$, and $a = \sqrt{k/(2D_e)}$, where $k = m\omega^2$, and recompute the eigenvalues and eigenfunctions.
15.5 Propagate a 1-d or 2-d wavepacket, initialized as a superposition of ground ψ_0 and first excited state ψ_1, $\psi(0) = 1/\sqrt{2}\psi_0 - 1/\sqrt{2}\psi_1$ of the 1-d or 2-d Morse oscillator defined in 15.2 and 15.3, by using the DVR Hamiltonian of item 15.4 and the Lanczos propagation scheme in the Krylov basis as implemented in *Expokit*. Compare your results with the analytic solution, based on the eigenvalues and eigenvectors: $\psi(t) = 1/\sqrt{2}\psi_0\exp(-iE_0 t) - 1/\sqrt{2}\psi_1\exp(-iE_1 t)$.

20 Fourier Grid Hamiltonian

The goal of this section is to introduce the Fourier grid Hamiltonian (FGH),

$$\begin{aligned} H(j, j') &= V(x_j)\langle x_j|x_j'\rangle + \frac{\Delta x \Delta p}{2\pi\hbar}\sum_{k=1}^{n_p} e^{\frac{i}{\hbar}(x_{j'}-x_j)p_k}\frac{p_k^2}{2m}, \\ &= V(x_j)\delta_{jj'} + \frac{\Delta x \Delta p}{2\pi\hbar}\sum_{k=1}^{n_p} e^{\frac{i}{\hbar}(x_{j'}-x_j)p_k}\frac{p_k^2}{2m}, \end{aligned} \tag{355}$$

introduced by Marston and Balint-Kurti [*J. Chem. Phys.* (1989) **91**:3571-3576], which is a special case of the DVR in the representation of equally spaced delta functions $\delta(x - x_j)$, at coordinates

$$x_j = (j - n_x/2)\Delta_x, \tag{356}$$

with $\Delta_x = (x_{max} - x_{min})/n_x$ and $j = 1$–n_x, where $p_k = \Delta p(k - n_p/2)$ with $\Delta p = 2\pi/(x_{max} - x_{min})$. Equation (355) is derived, as in the previous section although writing the kinetic energy in the basis of plane waves, as follows:

$$
\begin{aligned}
\langle x_l|\hat{T}|x_j\rangle &= \langle x_l|\frac{\hat{p}^2}{2m}|x_j\rangle, \\
&= \int dp \int dp' \langle x_l|p'\rangle \langle p'|\frac{\hat{p}^2}{2m}|p\rangle \langle p|x_j\rangle, \\
&= \int dp \int dp' \langle x_l|p'\rangle \frac{p^2}{2m}\langle p'|p\rangle \langle p|x_j\rangle, \\
&= \int dp \langle x_l|p\rangle \frac{p^2}{2m}\langle p|x_j\rangle = \frac{\Delta x}{2\pi\hbar}\int dp e^{\frac{i}{\hbar}(x_l-x_j)p}\frac{p^2}{2m}, \\
&= \frac{\Delta x \Delta p}{2\pi\hbar}\sum_{k=1}^{n_p} e^{\frac{i}{\hbar}(x_l-x_j)p_k}\frac{p_k^2}{2m},
\end{aligned}
\tag{357}
$$

since the identity operator is $\mathbf{I} = \sum_j |x_j\rangle\Delta x\langle x_j|$, in the discretized version of the delta function representation, and $\Delta x\langle x_j|x_k\rangle = \delta_{jk}$.

20.1 Computational Problem FGH

Write a program to solve the time independent Schrödinger equation by using the FGH method and apply it to find the first 4 eigenvalues and eigenfunctions of the Harmonic oscillator introduced by Eq. (17) with $m = 1$ and $\omega = 1$. Verify that the eigenvalues are $E(\nu) = (1/2 + \nu)\hbar\omega$, $\nu = 0$–10.

21 Tunneling Dynamics

The goal of this section is to show that calculations of eigenstates, based on the DVR method introduced in the previous section, can be used to compute the time-evolution of a wave-packet as an alternative approach to the SOFT method introduced in Sec. 6. Since the method is based on the solution of the time-independent Schrödinger equation, it is often called the 'time-independent method' for wave packet propagation. Here, we illustrate the method as applied to the simulation of quantum tunneling through a potential energy barrier in double-well potential energy surface. We show that, according to the description provided by quantum mechanics, motion (including tunneling) is simply the result of interference. Furthermore, we show that motion (including tunneling) can be manipulated by changing the relative phases of terms in coherent superposition states (see [*J. Mod. Optics* (2007) **54**:2617-2627]).

We consider a particle in a symmetric double-well, described by the following unperturbed Hamiltonian,

$$H_0(x, p) = \frac{p^2}{2} - \alpha(x^2 - \beta x^4), \tag{358}$$

with $\alpha = 1/2^2$ and $\beta = 1/2^5$. In the absence of an external perturbation, the initial non-stationary state

$$\Phi_0(x) = \pi^{-1/4} e^{-(x-x_0)^2/2}, \tag{359}$$

with $x_0 = -4$, has less energy than the height of the potential energy barrier centered at $x = 0$. Nevertheless, $\Phi_0(x)$ evolves in time, tunneling and recrossing back and forth.

The description of tunneling can be explained by considering the evolution of a non-stationary state (very similar to the initial state introduced in Eq. (359)),

$$\Phi_0(x) = \frac{1}{\sqrt{2}} \left(\chi_0(x) + \chi_1(x) \right), \tag{360}$$

where $\chi_0(x)$ and $\chi_1(x)$ are the ground and first excited states of the double-well that can be obtained by using the DVR method introduced in the previous section. Since $\hat{H}|\chi_j\rangle = E_j|\chi_j\rangle$,

$$|\Phi_t\rangle = \frac{1}{\sqrt{2}} \left(\chi_0(x) e^{-\frac{i}{\hbar} E_0 t} + \chi_1(x) e^{-\frac{i}{\hbar} E_1 t} \right), \tag{361}$$

and

$$|\xi(t)|^2 = |\langle \Phi_0 | \Phi_t \rangle|^2 = \frac{1}{2} + \frac{1}{2} cos(\Omega t), \tag{362}$$

with the tunneling frequency $\Omega = (E_1 - E_0)/\hbar$. Note that Ω is defined by the energy eigenvalues E_0 and E_1 and determines how frequently the particle recrosses the potential energy barrier by tunneling and maximizes the overlap with the initial state $|\Phi_0\rangle$.

An important observation, suggested by Eqs. (362) and (362), is that tunneling is the result of interference between the two components of the coherent superposition defined by Eq. (362) since changing the relative phases of the two terms would affect the underlying tunneling dynamics. For example, introducing a phase of θ in the first term of Eq. (361) we obtain,

$$|\Phi_t\rangle = \frac{1}{\sqrt{2}} \left(\chi_0(x) e^{-\frac{i}{\hbar} E_0 t} e^{i\theta} + \chi_1(x) e^{-\frac{i}{\hbar} E_1 t} \right), \tag{363}$$

and

$$|\xi(t)| = |\langle \Phi_0 | \Phi_t \rangle|^2 = \frac{1}{2} + \frac{1}{2} cos(\Omega t + \theta). \tag{364}$$

This equation indicates that the probability of having the system overalpping with the initial state on the left of the barrier at time t is a function of θ. Therefore, manipulating θ with an external field could be an effective method for coherently controlling the underlying tunneling dynamics.

21.1 Coherent Control of Tunneling Dynamics

As an example of coherent control of tunneling dynamics we consider the perturbational influence of instanteneous 2-θ pulses described by the following operator:

$$\hat{U}^{2\theta} = cos\left(\frac{\Gamma\tau}{2}\right)(|\Phi_0\rangle\langle\Phi_0| + |\Phi_a\rangle\langle\Phi_a|) - isin\left(\frac{\Gamma\tau}{2}\right)(|\Phi_0\rangle\langle\Phi_a| + |\Phi_a\rangle\langle\Phi_0|), \qquad (365)$$

with $\tau = 2\theta/\Gamma$. In particular, when $\theta = \pi$,

$$\hat{U}^{2\pi} = -(|\Phi_0\rangle\langle\Phi_0| + |\Phi_a\rangle\langle\Phi_a|), \qquad (366)$$

a pulse that induces a π phase-shift along the direction $|\Phi_0\rangle$. The goal of this subsection is to show that (bound to bound state) tunneling dynamics in the double-well can be delayed (and eventually halted) by coherently perturbing the system with a train of 2-π pulses.

Applying the 2 π pulse, described by Eq. (366), to a coherent state $|\Psi_{t_0}\rangle = c_0(t_0)|\Phi_0\rangle + c_1(t_0)|\Phi_1\rangle + ...$, that has neglible population in the auxiliary state $|\Phi_a\rangle$ (*i.e.*, $c_a(t_0) = 0$), we obtain

$$\hat{U}^{2\pi}|\Psi_t\rangle = -c_0(t_0)|\Phi_0\rangle + c_1(t_0)|\Phi_1\rangle + ... \qquad (367)$$

The pulse can also be represented as

$$\hat{U}^{2\pi} = 1 - 2|\Phi_0\rangle\langle\Phi_0|, \qquad (368)$$

since

$$\hat{U}^{2\pi}|\Psi_{t_0}\rangle = (1 - 2|\Phi_0\rangle\langle\Phi_0|)|\Psi_t\rangle = -c_0(t_0)|\Phi_0\rangle + c_1(t_0)|\Phi_1\rangle + ... \qquad (369)$$

The propagation of the system under the influence of N instantaneous 2-π pulses, applied at 2τ intervals, generates the time-evolved state,

$$|\Psi_{t+2N\tau}\rangle = c_0\left(e^{-\frac{i}{\hbar}\hat{H}\tau}\hat{U}^{2\pi}e^{-\frac{i}{\hbar}\hat{H}\tau}\right)^N|\Phi_0\rangle + e^{-\frac{i}{\hbar}\hat{H}2N\tau}(c_1|\Phi_1\rangle + ...),$$

$$= c_0(-1)^N e^{-\frac{i}{\hbar}(E_0+E_1)2N\tau}|\Phi_0\rangle + e^{-\frac{i}{\hbar}\hat{H}2N\tau}(c_1|\Phi_1\rangle + ...). \qquad (370)$$

The second equality in Eq. (370) is obtained by substituting $\hat{U}^{2\pi}$ as defined by Eq. (368) and Φ_0 according to Eq. (360).

Equation (370) shows that the square of the expansion coefficient associated with state Φ_0 remains constant, for as long as the train of 2-π pulses is applied. This indicates that tunneling is completely suppressed due to the repetitive change of the phase of the term associated with $|\Phi_0\rangle$, relative to the other terms in the coherent-state expansion.

21.2 Computational Problem 16

Modify the program for wave-packet propagation developed in Problem 12 and simulate the propagation of a wave packet in the symmetric double well

$$V(x) = -0.5x^2 + 1.0/(16.0*1.3544)x^4, \qquad (371)$$

88

using the initial state

$$\Phi_0(x) = \pi^{-1/4} e^{-0.5(x-x0)^2}, \tag{372}$$

with $x_0 = -2.1$.

16.1: Propagate the state for 1000 a.u., using a propagation step $\tau = 0.1$ a.u. and compute $|\xi(t)|^2$.

16.2: Compare your results with the corresponding results obtained by propagating the system under the influence of a train of 2-π pulses, as described by Eq. (368), applied in the time-window $t = 305$–500 a.u.

22   Optimal Control: Constrained Optimization

The goal of this section is to introduce the concept of *constrained optimization* as implemented in Optimal Control Theory (OCT). We consider the problem of finding the smallest perturbational field $\varepsilon(t)$ capable of evolving a given initial state $\psi(t_i) = \varphi_i$, according to the Schrödinger equation,

$$i\hbar \frac{\partial}{\partial t} \psi(t) = [H - \mu\varepsilon(t)] \psi(t), \tag{373}$$

to generate a state $\psi(t_f)$, at time t_f, with maximum expectation value $O(t_f) = \langle\psi(t_f)|O|\psi(t_f)\rangle$ of the observable O. We illustrate the method for the example discussed by Rabitz and co-workers where a 1-dimensional Morse oscillator described by the Hamiltonian $H = p^2/2m + V(x)$, with Morse potential $V(x)$, is initially prepared in state ϕ_0 and is then driven by the field $\varepsilon(t)$ to a state $\psi(t_f)$ with maximum overlap with ϕ_1 (e.g., with maximum $O(t_f) = \langle\psi(t_f)|O|\psi(t_f)\rangle$ with $O = |\phi_1\rangle\langle\phi_1|$). We implement a a general iterative approach to optimize the expectation $O(t_f)$, although other methodologies for shaping $\varepsilon(t)$ have also been proposed.

The standard way to deal with the maximization of $O(t_f)$, subject to the constraints of interest, is to maximize the functional,

$$K(\psi, \varepsilon, \alpha, \lambda) = O(t_f) - \int_{t_i}^{t_f} dt\alpha(t)|\varepsilon(t)|^2 - 2\text{Im} \int_{t_i}^{t_f} dt \int_{-\infty}^{\infty} dx\lambda(x,t) \left[H - \mu\varepsilon(t) - i\hbar\frac{\partial}{\partial t}\right] \psi(t),$$

$$\tag{374}$$

where $\alpha(t)$ are the Lagrange multipliers ensuring minimum field amplitude for all times t, and $\lambda(x,t)$ are the Lagrange multipliers ensuring that $\psi(t)$ and $\psi^*(t)$ satisfy Eq. (373) for all values of x and t. To find the conditions under which the Lagrange multipliers enforce these conditions, we find the extreme of K by making the derivatives of K with respect to ε, ψ and λ equal to zero.[2] For simplicity we make $\alpha(t) = \alpha/s(t)$, where α is a real constant and $s(t) = \sin^2(\pi t/t_f)$.

---

[2]Note that the imaginary part of the third term in Eq. (374) ensures that K remains *real* even when ψ and λ are *complex*, thus, the derivatives with respect to ψ^* and λ^* are simply the complex conjugate of equations for ψ and λ.

Considering that $2\text{Im}[z] = i(z^* - z)$, the derivative of K with respect to λ gives:

$$\frac{\partial K}{\partial \lambda(t)} = \mathrm{i} \int\limits_{t_i}^{t_f} \mathrm{d}t \left[H - \mu\varepsilon(t) - \mathrm{i}\hbar\frac{\partial}{\partial t} \right] \psi(t) = 0. \tag{375}$$

Therefore, $\psi(t)$ must satisfy the Schrödinger equation:

$$\boxed{i\hbar\frac{\partial}{\partial t}\psi(t) = [H - \mu\varepsilon(t)]\psi(t)} \tag{376}$$

with the boundary condition $\psi(t_i) = \phi_0$. Furthermore, the derivative of K with respect to ψ gives:

$$\begin{aligned}
\frac{\partial K}{\partial \psi(t)} &= O\psi^*(t_f)\delta(t - t_f) - [\lambda^*(t)\,\delta(t - t_f)]|_{t_i}^{t_f} \\
&\quad - \mathrm{i}\left(\mathrm{i}\hbar\frac{\partial}{\partial t} + [H - \mu\varepsilon(t)] \right)\lambda^*(t) = 0.
\end{aligned} \tag{377}$$

since by partial integration we obtain

$$\int\limits_{t_i}^{t_f} \left\langle \lambda(t) \left| \mathrm{i}\hbar\frac{\partial}{\partial t} - [H - \mu\varepsilon(t)] \right| \psi(t) \right\rangle \mathrm{d}t = \mathrm{i}\left\langle \lambda(t) | \psi(t) \right\rangle|_{t_i}^{t_f} - \mathrm{i}\int\limits_{t_i}^{t_f} \left\langle \hbar\frac{\partial}{\partial t}\lambda(t) | \psi(t) \right\rangle \mathrm{d}t$$

$$- \int\limits_{t_i}^{t_f} \left\langle [H - \mu\varepsilon(t)]\lambda(t) | \psi(t) \right\rangle \mathrm{d}t \tag{378}$$

$$= \mathrm{i}\left\langle \lambda(t) | \psi(t) \right\rangle|_{t_i}^{t_f}$$

$$+ \int\limits_{t_i}^{t_f} \left\langle \left(\mathrm{i}\hbar\frac{\partial}{\partial t} - [H - \mu\varepsilon(t)] \right)\lambda(t) | \psi(t) \right\rangle \mathrm{d}t \tag{379}$$

Equation (377) is satisfied when

$$\boxed{\lambda(t_f) = O\psi(t_f)} \tag{380}$$

and $\lambda(t)$ satisfies the Schrödinger equation,

$$\boxed{i\hbar\frac{\partial}{\partial t}\lambda(t) = [H - \mu\varepsilon(t)]\lambda(t)} \tag{381}$$

Finally, the variation with respect to the field gives,

$$\begin{aligned}
\frac{\delta K}{\delta \varepsilon} &= -2\alpha\frac{|\varepsilon(t)|}{s(t)} \\
&\quad -2\text{Im}\left\langle \lambda(t) |\mu| \psi(t) \right\rangle = 0,
\end{aligned} \tag{382}$$

90

hence

$$\boxed{\varepsilon\left(t\right) = -\frac{s\left(t\right)}{\hbar\alpha} \cdot \mathrm{Im}\left\langle \lambda\left(t\right) \left|\mu\right| \psi\left(t\right)\right\rangle} \tag{383}$$

The boxed equations are the so-called 'control-field equations' since they can be used to obtain the *optimum control field* according to the following iterative procedure. Starting with an initial guess for the field ε, propagate ψ from t_i to t_f, according to Eq. (376). Having $\psi(t_f)$, we compute $\lambda(t_f)$ according to Eq. (380) and we propagate it backwards to time t_i according to Eq. (381) to obtain $\lambda(t_i)$. Propagating $\lambda(t_i)$ and $\psi(t_i)$ from t_i to t_f allow us to update $\varepsilon(t)$ according to Eq. (383). The propagation also yields a new $\psi(t_f)$ which can be used to obtain a new $\lambda(t_f)$, and therefore a new $\lambda(t_i)$ by backward propagation. This procedure is repeated to update $\varepsilon(t)$ and $\psi(t_f)$ multiple times until convergence.

Simple inspection of the OCT functional, introduced by Eq. (374), shows that the first term (the expectation value) and the second term (penalty term due to the field strength) are in competition with each other, which often leads to the trivial solution $\varepsilon(t) = 0$. Such a difficulty can be removed by using a swept penalty parameter α, allowing the method to find a regime in which the yield is large, then slowly increasing α to minimize the field strength and making it follow the oscillations of the wave functions more closely, resulting in smooth oscillations of the electric field.

22.1   Problem OCT

(*This assignment was suggested, organized and solved by Dr. Andreas Markmann*) Using the code for propagation a particle in a Morse potential (Problem 8), modify the code to write an optimal control loop to find an electric field that approximately evolves an initial state $\psi(t_i) = \phi_0$ into a state $\psi(t_f) = \phi_1$, where ϕ_0 and ϕ_1 are the ground and first excited states to the harmonic oscillator that approximates the Morse potential, as follows:

1. Write a subroutine that saves the ground and first excited vibrational state of that harmonic oscillator into an array.

2. Propagate these states in the Morse oscillator and visualize the evolution of the probability density to demonstrate that these are not exactly energy eigenstates of the Morse oscillator.

3. Write an optimal control loop around the propagation time loop that finds an electric field that approximately takes an initial state that is the harmonic oscillator ground state to the harmonic oscillator excited state.

4. Find an appropriate initial guess for ε and a range of penalty parameters α that give a yield larger than 0.5. Assume a linear dipole moment of $\mu\left(x\right) = x$ (in atomic units).

5. To interpret your results, use the Poynting vector of the electromagnetic field to convert the electric field strength from atomic units to a laser power in W/cm^2.

22.2 D-MORPH Optimization

We consider a probe system prepared in the inital state $\psi_i = \psi(t_0)$, evolving on the potential energy surface $\hat{V}_0(x)$ under the influence of the perturbational field $E(t, \beta)$ according to the time-dependent Hamiltonian:

$$\hat{H}(t; \beta) = \frac{\hat{p}^2}{2m(t, \beta)} + \hat{V}(x, t, \beta), \tag{384}$$

where $\beta = (\beta_1, \cdots, \beta_N)$ are control parameters defining the time-dependent interaction potential $\hat{V}(x, t, \beta) = V_0(x) - \mu(x, t, \beta) \cdot E(t, \beta)$ determined by the external field $E(t, \beta)$ and time-dependent dipole moment $\mu(x, t, \beta)$. The mass of the system $m(t, \beta)$ is also assumed to be time-dependent and determined by the controls. For simplicity, we consider notation in atomic units, so $\hbar = 1$, the electron mass $m_e = 1$ au, the electron charge $q_e = 1$ au, and the permittivity of a vacuum $4\pi\varepsilon_0 = 1$ au.

The aim is to find parameters β so that the final state $\psi_f(\beta) = U(t_f, t_0; \beta)\psi(t_0)$, obtained by propagation of the initial state $\psi(t_0)$ for time $T = t_f - t_0$, minimizes the expectation value of the potential energy:

$$\begin{aligned}\left\langle \hat{V}_0(T; \beta) \right\rangle &= \langle \psi_f(\beta) | V_0(x) | \psi_f(\beta) \rangle, \\ &= \left\langle \psi_i \left| \hat{V}_0(T, x; \beta) \right| \psi_i \right\rangle.\end{aligned} \tag{385}$$

Here, $\hat{V}_0(T, x; \beta) = U^\dagger(t_f, t_0; \beta) V_0(x) U(t_f, t_0; \beta)$ with the time-evolution operator,

$$U(t, t_0; \beta) = \mathrm{e}^{-\frac{i}{\hbar} \int\limits_{t_0}^{t} \hat{H}(t'; \beta) \mathrm{d}t'}. \tag{386}$$

We minimize $\left\langle \hat{V}_0(T; \beta) \right\rangle$ by considering that the controls β_j are parametrically dependent on a scalar s and changed as follows:

$$\beta_j(s + \Delta s) = \beta_j(s) + \frac{\partial \beta_j(s)}{\partial s} \Delta s. \tag{387}$$

By choosing the direction of the gradients,

$$\frac{\partial \beta_j(s)}{\partial s} = -\frac{\partial \left\langle \hat{V}_0(T; \beta) \right\rangle}{\partial \beta_j}, \tag{388}$$

$\left\langle \hat{V}_0(T, x; \beta) \right\rangle$ decreases by steepest descent, as follows:

$$\begin{aligned}\left\langle \hat{V}_0(T, x; \beta(s + \Delta s)) \right\rangle &= \left\langle \hat{V}_0(T; \beta(s)) \right\rangle + \sum_j \frac{\partial \left\langle \hat{V}_0(T, x; \beta) \right\rangle}{\partial \beta_j} \frac{\partial \beta_j(s)}{\partial s} \Delta s \\ &= \left\langle \hat{V}_0(T, x; \beta(s)) \right\rangle - \sum_j \left| \frac{\partial \left\langle \hat{V}_0(T; \beta) \right\rangle}{\partial \beta_j} \right|^2 \Delta s\end{aligned} \tag{389}$$

As shown below, the gradients are efficiently computed, as follows [Ho, T.; Rabitz, H. J. Photochem. Photobio. A: Chem. 2006, 180, 226240]:

$$\frac{\partial \left\langle \hat{V}_0(T; \beta) \right\rangle}{\partial \beta_j} = -\frac{i}{\hbar} \int_0^T dt \left[\left\langle \psi_B(t) \left| H_{\beta_j}(\beta, t) \right| \psi_C(t) \right\rangle - \left\langle \psi_C(t) \left| H_{\beta_j}(\beta, t) \right| \psi_B(t) \right\rangle \right],$$

$$= \frac{2}{\hbar} \int_0^T dt \ \Im \left[\left\langle \psi_B(t) \left| H_{\beta_j}(\beta, t) \right| \psi_C(t) \right\rangle \right], \tag{390}$$

where $H_{\beta_j} = \frac{\partial \hat{H}}{\partial \beta_j}$. Here, $|\psi_C(t)\rangle = U(t, 0; \beta) |\psi_i\rangle$ is obtained by forward propagation of the initial state $|\psi_i\rangle$. The state $|\psi_B(t)\rangle = U(t, 0; \beta) |\psi_A\rangle$ is obtained by forward propagation of the state $|\psi_A\rangle = U^\dagger(T, 0; \beta) V_0(x) U(T, 0; \beta) |\psi_i\rangle$ previously prepared by forward and backward propagation. Therefore, only 4 propagations are necessary to obtain the complete set of gradients, according to Eq. (390), for any arbitrary number N of control parameters β_j. This is the main advantage of the D-MORPH method when compared to other approaches, such as the second order differencing scheme that would require $N+1$ propagations to obtain the same number of gradients. Another advantage is that the gradient-based optimization in control space allows for global optimization, since the number of local minima can in principle be reduced by increasing the number of controls. Previous studies have integrated Eq. (387) by using fourth-order variable step-size Runge-Kutta, which is computationally demanding. A more efficient approach is based on optimization instead of integration by using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method.

To show that the gradients can be computed according to Eq. (390), we first consider that the gradients of the potential energy with respect to the control parameters are

$$\frac{\partial \left\langle \hat{V}_0(T; \beta) \right\rangle}{\partial \beta_j} = \left\langle \psi_i \left| \frac{\partial}{\partial \beta_j} \left[U^\dagger(T, 0, \beta) V_0(x) U(T, 0, \beta) \right] \right| \psi_i \right\rangle, \tag{391}$$

and can be computed according to Eq. (390) since

$$\frac{\partial}{\partial \beta_j} \left[U^\dagger(T, 0, \beta) V_0(x) U(T, 0, \beta) \right] = (-i/\hbar) \int_0^T \left[U^\dagger(T, 0, \beta) V_0(x) U(T, t, \beta) \hat{H}_{\beta_j}(t', \beta) U(T, 0, \beta) \right.$$

$$\left. - U^\dagger(T, 0, \beta) \hat{H}_{\beta_j}(t', \beta) U^\dagger(t, T, \beta) V_0(x) U(T, 0, \beta) \right] dt, \tag{392}$$

where subscript β_j denotes the partial derivative with respect to β_j.

To derive Eq. (392), we consider that

$$\frac{\partial}{\partial \beta_j} \left[U^\dagger(T, 0, \beta) V_0(x) U(T, 0, \beta) \right] = U_{\beta_j}^\dagger(T, 0, \beta) V_0(x) U(T, 0, \beta)$$

$$+ U^\dagger(T, 0, \beta) V_0(x) U_{\beta_j}(T, 0, \beta), \tag{393}$$

93

where the propagator, introduced by Eq. (386), satisfies the time-dependent Schrödinger equation,

$$i\hbar \frac{\partial}{\partial t} U(t, 0, \beta) = \hat{H}(t, \beta) U(t, 0, \beta), \tag{394}$$

and its conjugate satisfies

$$-i\hbar \frac{\partial U^\dagger(t, 0, \beta)}{\partial t} = U^\dagger(t, 0, \beta) \hat{H}(t, \beta). \tag{395}$$

Its partial derivative with respect to β_j, U_{β_j}, satisfies the following equation:

$$i\hbar \frac{\partial}{\partial t} U_{\beta_j}(t, 0, \beta) = \hat{H}_{\beta_j}(t, \beta) U(t, 0, \beta) + \hat{H}(t, \beta) U_{\beta_j}(t, 0, \beta). \tag{396}$$

Therefore, according to Eqs. (396) and (395), we obtain:

$$\begin{aligned}
i\hbar \frac{\partial}{\partial t} \left[U^\dagger(t, 0) U_{\beta_j}(t, 0) \right] &= i\hbar \left[\frac{\partial U^\dagger(t, 0)}{\partial t} U_{\beta_j}(t, 0) + U^\dagger(t, 0) \frac{\partial U_{\beta_j}(t, 0)}{\partial t} \right] \\
&= -U^\dagger(t, 0) H(t) U_{\beta_j}(t, 0) + \\
&\quad + U^\dagger(t, 0) \left[H_{\beta_j}(\beta, t) U(t, 0) + H(\beta, t) U_{\beta_j}(t, 0) \right] \\
&= U^\dagger(t, 0) H_{\beta_j}(\beta, t) U(t, 0).
\end{aligned} \tag{397}$$

Integration with respect to time and multiplication by the forward propagator yields,

$$\begin{aligned}
U_{\beta_j}(t, 0) &= (-i/\hbar) U(t, 0) \int_0^t dt' U^\dagger(t', 0) H_{\beta_j}(\beta, t') U(t', 0) \\
&= (-i/\hbar) \int_0^t dt' U(t, t') H_{\beta_j}(\beta, t') U(t', 0).
\end{aligned} \tag{398}$$

and the conjugate of Eq. (398), gives:

$$U_{\beta_j}^\dagger(t, 0) = (i/\hbar) \int_0^t dt' U^\dagger(t', 0) H_{\beta_j}(\beta, t') U^\dagger(t, t'). \tag{399}$$

Substituting Eqs. (398) and (399) into Eq. (393) gives Eq. (392).

23   Linear Photoabsorption Lineshape: A Time Dependent Picture

The goal of this section is to show that the linear photoabsorption lineshape $I_0(\omega)$ of a system (at 0 K) can be obtained from the Fourier transform of the survival amplitude $\xi(t) \equiv \langle \Psi_0 | \Psi_t \rangle = \langle \Psi_0 | e^{-i\hat{H}t/\hbar} | \Psi_0 \rangle$ as follows:

$$I_0(\omega) = \frac{3}{2\pi\hbar} \int_{-\infty}^{\infty} dt\, e^{\frac{i}{\hbar}(\hbar\omega + E_0)t} \xi(t), \tag{400}$$

where $|\Psi_0\rangle = \lambda\overrightarrow{\varepsilon_0} \cdot \hat{\mu}|\Phi_0\rangle$, with $\hat{\mu}$ the dipole moment operator and $|\Phi_0\rangle$ the ground state of the unperturbed system described by the Hamiltonian $\hat{H}$.

Computations of $P(\omega)$, based on Eq. (400), can be performed by propagation of $|\Psi_0\rangle$ (*e.g.*, according to the SOFT method introduced in Sec. 6); computation of the survival amplitude $\xi(t)$ by overlaping the time evolved state $|\Psi_t\rangle$ and the initial state $|\Psi_0\rangle$; and finally calculation of the Fourier transform of $\xi(t)$ by using the FFT algorithm. The initial state $|\Phi_0\rangle$ can be obtained by using the DVR method, introduced in Sec. 19, to solve the eigenvalue problem $\hat{H}|\Phi_0\rangle = E_0|\Phi_0\rangle$.

Calculations of the spectrum $I_\beta(\omega)$, at finite temperature $T = 1/(\beta k_B)$, can be performed as follows:

$$I_\beta(\omega) = \sum_j \rho_j(\beta)I_j(\omega), \tag{401}$$

where $\hat{H}|\Phi_j\rangle = E_j|\Phi_j\rangle$, $\rho_j(\beta) = Z^{-1}e^{-\beta E_j}$, and $Z = \sum_j e^{-\beta E_j}$. The computations of $I_j(\omega)$ are analogous to the computation of $I_0(\omega)$ but using $|\Phi_j\rangle$ as the initial state, instead of $|\Phi_0\rangle$.

The total transition probability (at 0 K) due to the interaction of the system with the external radiation field can be obtained by first computing the transition probablity to state $|\Phi_k\rangle$ as follows:

$$P_0^{(k)}(\omega) = \lim_{t\to\infty} |c_k^{(1)}(t)|^2, \tag{402}$$

where $c_k^{(1)}(t)$ is defined by first order time-dependent perturbation theory,

$$c_k^{(1)}(t) = -\frac{i}{\hbar} \int_{-\infty}^t dt' \langle\Phi_k|e^{\frac{i}{\hbar}\hat{H}t'}\hat{H}_1(t')e^{-\frac{i}{\hbar}\hat{H}t'}|\Phi_0\rangle. \tag{403}$$

The derivation of Eq. (403), presented in the following section, assumes that the photoabsorption results from the interaction of the system with the monochromatic radiation field,

$$\overrightarrow{\epsilon(t)} = \lambda\overrightarrow{\varepsilon_0}(e^{i\omega t} + e^{-i\omega t}), \tag{404}$$

where $\lambda << 1$ is a small dimensionless parameter that defines the dipolar interaction,

$$\hat{H}_1(t) = -\lambda\overrightarrow{\varepsilon_0} \cdot \hat{\mu}(e^{i\omega t} + e^{-i\omega t}), \tag{405}$$

in the weak field limit.

Substituting the expression of the dipolar interaction, introduced by Eq. (405), into Eq. (403), we obtain:

$$c_k^{(1)}(t) = -\frac{i}{\hbar} \int_{-\infty}^t dt' \langle\Phi_k|\lambda\overrightarrow{\varepsilon_0} \cdot \hat{\mu} \left[e^{-\frac{i}{\hbar}(\hat{H}-E_k-\hbar\omega)t'} + e^{-\frac{i}{\hbar}(\hat{H}-E_k+\hbar\omega)t'}\right] |\Phi_0\rangle, \tag{406}$$

and substituting Eq. (406) into Eq. (402) we obtain:

$$P_0^{(k)}(\omega) = |\langle\Phi_k|\overrightarrow{\lambda\varepsilon_0} \cdot \hat{\mu}|\Phi_0\rangle|^2 \left[\delta(E_0 + \hbar\omega - E_k) + \delta(E_0 - \hbar\omega - E_k)\right]. \tag{407}$$

The total energy lost from the radiation to the system (at 0 K), due to the transition to state $|\Phi_k\rangle$, can be obtained by multiplying $P_0^{(k)}$ by the energy of that transition $(E_k - E_0)$ and summing over all final states as follows:

$$\alpha_0(\omega) = \sum_k (E_k - E_0)|\langle\Phi_k|\overrightarrow{\lambda\varepsilon_0} \cdot \hat{\mu}|\Phi_0\rangle|^2 \left[\delta(E_0 + \hbar\omega - E_k) + \delta(E_0 - \hbar\omega - E_k)\right]. \qquad (408)$$

The absorption spectrum $\alpha_\beta(\omega)$, at finite temperature $T = 1/(\beta k_B)$, can be obtained from Eq. (408) as follows:

$$\begin{aligned}
\alpha(\omega) &= \sum_j \rho_j \sum_k (E_k - E_j)\delta(E_j + \hbar\omega - E_k)|\langle\Phi_k|\overrightarrow{\lambda\varepsilon_0} \cdot \hat{\mu}|\Phi_j\rangle|^2 \\
&+ \sum_j \rho_j \sum_k \sum_k (E_k - E_j)\delta(E_j - \hbar\omega - E_k)|\langle\Phi_k|\overrightarrow{\varepsilon_0} \cdot \hat{\mu}|\Phi_j\rangle|^2,
\end{aligned} \qquad (409)$$

where $\hat{H}|\Phi_j\rangle = E_j|\Phi_j\rangle$, $\rho_j = Z^{-1}e^{-\beta E_j}$, and $Z = \sum_j e^{-\beta E_j}$.

Interchanging the indices j and k in the second term of Eq. (409) and noting that $\rho_k = \rho_j e^{-\beta(E_k - E_j)}$ we obtain:

$$\begin{aligned}
\alpha(\omega) &= \sum_j \sum_k \rho_j(E_k - E_j)\delta(E_j + \hbar\omega - E_k)|\langle\Phi_k|\overrightarrow{\lambda\varepsilon_0} \cdot \hat{\mu}|\Phi_j\rangle|^2 \\
&- \rho_j e^{-\beta(E_k - E_j)}(E_k - E_j)\delta(E_j + \hbar\omega - E_k)|\langle\Phi_j|\overrightarrow{\varepsilon_0} \cdot \hat{\mu}|\Phi_k\rangle|^2,
\end{aligned} \qquad (410)$$

which gives the absorption lineshape

$$I(\omega) = \frac{3\alpha(\omega)}{\hbar\omega(1 - e^{-\beta\omega})} = 3\sum_j \sum_k \rho_j\delta(E_j + \hbar\omega - E_k)|\langle\Phi_k|\overrightarrow{\lambda\varepsilon_0} \cdot \hat{\mu}|\Phi_j\rangle|^2. \qquad (411)$$

At 0 K, the absorption lineshape is obtained from Eq. (411) as follows:

$$I_0(\omega) = 3\sum_k \delta(E_0 + \hbar\omega - E_k)|\langle\Phi_k|\overrightarrow{\lambda\varepsilon_0} \cdot \hat{\mu}|\Phi_0\rangle|^2, \qquad (412)$$

that is equivalent to Eq. (400), since according to Eq. (400),

$$I_0(\omega) = \frac{3}{2\pi\hbar}\int_{-\infty}^{\infty} dt\langle\Phi_0|(\overrightarrow{\lambda\varepsilon_0} \cdot \hat{\mu})e^{\frac{i}{\hbar}(\hbar\omega + E_0 - \hat{H})t}(\overrightarrow{\lambda\varepsilon_0} \cdot \hat{\mu})|\Phi_0\rangle. \qquad (413)$$

Note that introducing the closure relation, $\hat{1} = \sum|\Phi_j\rangle\langle\Phi_j|$ into Eq. (413), we obtain:

$$\begin{aligned}
I_0(\omega) &= 3\sum_k \langle\Phi_0|\overrightarrow{\lambda\varepsilon_0} \cdot \hat{\mu}|\Phi_k\rangle\langle\Phi_k|\delta(\hbar\omega + E_0 - \hat{H})\overrightarrow{\lambda\varepsilon_0} \cdot \hat{\mu}|\Phi_0\rangle, \\
&= 3\sum_k \delta(\hbar\omega + E_0 - E_k)\langle\Phi_0|\overrightarrow{\lambda\varepsilon_0} \cdot \hat{\mu}|\Phi_k\rangle\langle\Phi_k|\overrightarrow{\lambda\varepsilon_0} \cdot \hat{\mu}|\Phi_0\rangle,
\end{aligned} \qquad (414)$$

that is identical to Eq. (412).

Finally, we note that Eq. (414) gives the linear photoabsorption lineshape in terms of the dipole-dipole correlation function as follows:

$$
\begin{aligned}
I_0(\omega) &= \frac{3}{2\pi\hbar} \sum_k \int_{-\infty}^{\infty} dt e^{i\omega t} \langle \Phi_0 | e^{\frac{i}{\hbar}\hat{H}t} (\lambda \overrightarrow{\varepsilon_0} \cdot \hat{\mu}) e^{-\frac{i}{\hbar}\hat{H}t} | \Phi_k \rangle \langle \Phi_k | (\lambda \overrightarrow{\varepsilon_0} \cdot \hat{\mu}) | \Phi_0 \rangle, \\
&= \frac{3}{2\pi\hbar} \int_{-\infty}^{\infty} dt e^{i\omega t} \langle \Phi_0 | (\lambda \overrightarrow{\varepsilon_0} \cdot \hat{\mu}(t))(\lambda \overrightarrow{\varepsilon_0} \cdot \hat{\mu}(0)) | \Phi_0 \rangle,
\end{aligned}
\tag{415}
$$

as well as the finite temperature photoabsorption lineshape,

$$
I_\beta(\omega) = \frac{3}{2\pi\hbar} \int_{-\infty}^{\infty} dt e^{i\omega t} Tr[\hat{\rho}\hat{\mu}(t)\hat{\mu}(0)],
\tag{416}
$$

where $\hat{\rho} = Z^{-1} exp(-\beta\hat{H})$ and $Z = Tr[\hat{\rho}]$.

24 Time Dependent Perturbation Theory

The goal of this section is to derive Eq. (403) as well as the corresponding expression at second order, necessary for the discussion of non-linear (pump-probe) spectroscopy presented in the following section.

Given an arbitary state,**R2(410)**

$$
\tilde{\psi}(x,t) = \sum_j C_j \Phi_j(x) e^{-\frac{i}{\hbar}E_j t},
$$

for the initially unperturbed system described by the Hamiltonian $\hat{H}$, for which $\hat{H}\hat{\Phi}_j = E_j \Phi_j$ and $i\hbar\frac{\partial\tilde{\psi}}{\partial t} = \hat{H}\tilde{\psi}$, let us obtain the solution of the time dependent Schrödinger equation:

$$
i\hbar\frac{\partial\psi}{\partial t} = [\hat{H} + \lambda\hat{\omega}(t)]\psi,
\tag{417}
$$

assuming that such solution can be written as a rapidly convergent expansion in powers of λ,

$$
\psi_\lambda(x,t) = \sum_j \sum_{l=0}^{\infty} C_{jl}(t)\lambda^l \Phi_j(x) e^{-\frac{i}{\hbar}E_j t}.
\tag{418}
$$

Substituting Eq. (418) into Eq. (417) we obtain,

$$
i\hbar\sum_{l=0}^{\infty}\left(\dot{C}_{kl}(t)\lambda^l + C_{kl}(t)\lambda^l(-\frac{i}{\hbar}E_k)\right) e^{-\frac{i}{\hbar}E_k t} = \sum_j \sum_{l=0}^{\infty} C_{jl}(t)\lambda^l \left(\langle\Phi_k|\Phi_j\rangle E_j + \lambda\langle\Phi_k|\hat{\omega}|\Phi_j\rangle\right) e^{-\frac{i}{\hbar}E_j t}.
\tag{419}
$$

Terms with λ^0: (Zero-order time dependent perturbation theory)

$$+i\hbar[\dot{C}_{k_0}(t)e^{-\frac{i}{\hbar}E_k t} + C_{k_0}(t)(-\frac{i}{\hbar}E_k)e^{-\frac{i}{\hbar}E_k t}] = \sum_j C_{j_0}(t)\delta_{kj}E_j e^{-\frac{i}{\hbar}E_j t} = C_{k_0}(t)E_k e^{-\frac{i}{\hbar}E_k t}.$$

Since,

$$\dot{C}_{k_0}(t) = 0, \qquad \Rightarrow \qquad C_{k_0}(t) = C_{k_0}(0).$$

Therefore, the unperturbed wave function is correct to zeroth order in λ.

Terms with λ^0: (Zero-order time dependent perturbation theory)

$$+i\hbar[\dot{C}_{k_0}(t)e^{-\frac{i}{\hbar}E_k t} + C_{k_0}(t)(-\frac{i}{\hbar}E_k)e^{-\frac{i}{\hbar}E_k t}] = \sum_j C_{j_0}(t)\delta_{kj}E_j e^{-\frac{i}{\hbar}E_j t} = C_{k_0}(t)E_k e^{-\frac{i}{\hbar}E_k t}.$$

Since,

$$\dot{C}_{k_0}(t) = 0, \qquad \Rightarrow \qquad C_{k_0}(t) = C_{k_0}(0).$$

Therefore, the unperturbed wave function is correct to zeroth order in λ.

Terms with λ: (First-order time dependent perturbation theory)

$$i\hbar[\dot{C}_{k_1}(t)e^{-\frac{i}{\hbar}E_k t} + C_{k_1}(t)(-\frac{i}{\hbar}E_k)e^{-\frac{i}{\hbar}E_k t}] = \sum_j C_{j_1}(t)\delta_{kj}E_j e^{-\frac{i}{\hbar}E_j t} + C_{j_0}(t) < \Phi_k|\hat{\omega}|\Phi_j > e^{-\frac{i}{\hbar}E_j t},$$

$$\dot{C}_{k_1}(t) = -\frac{i}{\hbar}\sum_j \left(C_{j_0}(0) < \Phi_k|\hat{\omega}|\Phi_j > e^{-\frac{i}{\hbar}(E_j - E_k)t} \right).$$

Therefore,

$$\dot{C}_{k_1}(t) = -\frac{i}{\hbar}\sum_j C_{j_0}(0) < \Phi_k|e^{\frac{i}{\hbar}E_k t}\hat{\omega}e^{-\frac{i}{\hbar}E_j t}|\Phi_j >= -\frac{i}{\hbar}\sum_j C_{j_0}(0) < \Phi_k|e^{\frac{i}{\hbar}\hat{H}t}\hat{\omega}e^{-\frac{i}{\hbar}\hat{H}t}|\Phi_j >,$$

$$(420)$$

Equation (420) is obtained by making the substitution $e^{-\frac{i}{\hbar}\hat{H}t}|\Phi_j >= e^{-\frac{i}{\hbar}E_j t}|\Phi_j >$, as justified in the note presented below. Integrating Eq. (420) we obtain,

$$C_{k_1}(t) = -\frac{i}{\hbar}\int_{-\infty}^t dt' \sum_j C_{j_0}(0) < \Phi_k|e^{\frac{i}{\hbar}\hat{H}t'}\hat{\omega}e^{-\frac{i}{\hbar}\hat{H}t'}|\Phi_j > \qquad (421)$$

which can also be written as follows:

$$C_{k_1}(t) = -\frac{i}{\hbar}\int_{-\infty}^t dt' < \Phi_k|e^{\frac{i}{\hbar}\hat{H}t'}\hat{\omega}e^{-\frac{i}{\hbar}\hat{H}t'}|\tilde{\psi}_0 > . \qquad (422)$$

This expression gives the correction of the expansion coefficients to first order in λ.

Note: The substitution made in Eq. (420) can be justified as follows. The exponential function is defined in powers series as follows,

$$e^A = \sum_{n=0}^{\infty} \frac{A^n}{n!} = 1 + A + \frac{1}{2!}AA + ..., \qquad \textbf{R4(169)}$$

In particular, when $A = -i\hat{H}t/\hbar$,

$$e^{-\frac{i}{\hbar}\hat{H}t} = 1 + (-\frac{i}{\hbar}\hat{H}t) + \frac{1}{2!}(-\frac{i}{\hbar}t)^2\hat{H}\hat{H} + ...$$

Furthermore, since

$$\hat{H}|\Phi_j> = E_j|\Phi_j>,$$

and,

$$\hat{H}\hat{H}|\Phi_j> = E_j\hat{H}|\Phi_j> = E_j^2|\phi_j>,$$

we obtain,

$$e^{-\frac{i}{\hbar}\hat{H}t}|\Phi_j> = [1 + (-\frac{i}{\hbar}E_jt) + \frac{1}{2!}(-\frac{i}{\hbar}t)^2E_j^2 + ...]|\Phi_j> = e^{-\frac{i}{\hbar}E_jt}|\Phi_j>,$$

which is the substitution implemented in Eq. (420).

   Terms with λ^2: (Second-order time dependent perturbation theory)

$$i\hbar[\dot{C}_{k_2}(t) + C_{k_2}(t)(-\frac{i}{\hbar}E_k)]e^{-\frac{i}{\hbar}E_kt} = \sum_j[C_{j_2}(t)\delta_{kj}E_j + C_{j_1}(t) < \Phi_k|\hat{\omega}|\Phi_j>]e^{-\frac{i}{\hbar}E_jt},$$

$$\dot{C}_{k_2}(t) = -\frac{i}{\hbar}\sum_j < \Phi_k|e^{\frac{i}{\hbar}\hat{H}t}\hat{\omega}e^{-\frac{i}{\hbar}\hat{H}t}|\Phi_j > C_{j_1}(t),$$

$$C_{k_2}(t) = \left(-\frac{i}{\hbar}\right)\int_{-\infty}^t dt' \sum_j < \Phi_k|e^{\frac{i}{\hbar}\hat{H}t'}\hat{\omega}e^{-\frac{i}{\hbar}\hat{H}t'}|\Phi_j > C_{j_1}(t'),$$

$$C_{k_2}(t) = \left(-\frac{i}{\hbar}\right)^2 \sum_j \int_{-\infty}^t dt' \int_{-\infty}^{t'} dt'' < \Phi_k|e^{\frac{i}{\hbar}\hat{H}t'}\hat{\omega}e^{-\frac{i}{\hbar}\hat{H}t'}|\Phi_j >< \Phi_j|e^{\frac{i}{\hbar}\hat{H}t''}\hat{\omega}e^{-\frac{i}{\hbar}\hat{H}t''}|\tilde{\psi}_0 > .$$

Since $1 = \sum_j |\Phi_j >< \Phi_j|$,

$$C_{k_2}(t) = \left(-\frac{i}{\hbar}\right)^2 \int_{-\infty}^t dt' \int_{-\infty}^{t'} dt'' < \Phi_k|e^{\frac{i}{\hbar}\hat{H}t'}\hat{\omega}e^{-\frac{i}{\hbar}\hat{H}(t'-t'')}\hat{\omega}e^{-\frac{i}{\hbar}\hat{H}t''}|\tilde{\psi}_0 > .$$

This expression gives the correction of the expansion coefficients to second order in λ.

25   Nonlinear (Pump-Probe) Spectroscopy

The goal of this section is to obtain the nonlinear pump-probe photoabsorption lineshape $I(\omega_2, \Delta t)$ due to the interaction of a molecular system with the radiation field,

$$\overrightarrow{\varepsilon(t)} = \lambda F_1(t - t_0)\overrightarrow{\varepsilon_{01}}e^{-i\omega_1 t} + \lambda F_2(t - t_0 - \Delta t)\overrightarrow{\varepsilon_{02}}e^{-i\omega_2 t} + c.c. \tag{423}$$

The field corresponds to pump and probe pulses with temporal profiles F_1 and F_2 centered at $t = t_0$ and $t = t_0 + \Delta t$, respectively. The time delay Δt between the pump and probe pulses allows this technique to probe the excited state dynamics at various times Δt after photoexcitiation of the system.

The total transition probability P_0 (at 0 K), due to the two-photon interaction of the system with the external radiation field, is obtained by first computing the transition probablity to state $|\Phi_k\rangle$ and then summing over all possible final states as follows:

$$P_0 = \sum_k P_0^{(k)} = \lim_{t_f \to \infty} \sum_k |c_k^{(2)}(t_f)|^2, \tag{424}$$

where $c_k^{(2)}(t_f)$ is defined by second order time-dependent perturbation theory,

$$c_k^{(2)}(t_f) = \hbar^{-2} \int_{-\infty}^{t_f} dt' \int_{-\infty}^{t'} dt'' \langle\Phi_k|e^{\frac{i}{\hbar}\hat{H}t''}\hat{H}_1(t'')e^{-\frac{i}{\hbar}\hat{H}(t''-t')}\hat{H}_1(t')e^{-\frac{i}{\hbar}\hat{H}t'}|\Phi_0\rangle, \tag{425}$$

with

$$\hat{H}_1(t) = -(\lambda\overrightarrow{\varepsilon_{01}} \cdot \hat{\mu})F_1(t - t_0)e^{-i\omega_1 t} - (\lambda\overrightarrow{\varepsilon_{02}} \cdot \hat{\mu})F_2(t - t_0 - \Delta t)e^{-i\omega_2 t} + c.c. \tag{426}$$

Substituting Eq. (426) into Eq. (425) and then substituting Eq. (424) into Eq. (424) we obtain a sum of 16 terms, associated with all possible pairs of interactions $(\pm\omega_j, \pm\omega_k)$ with $j = 1, 2$ and $k = 1, 2$. In particular, the term $(+\omega_1, +\omega_2)$ is

$$P_0(+\omega_2, +\omega_1) = \hbar^{-4} \int_{-\infty}^{\infty} dt \int_{-\infty}^{t} dt' \int_{-\infty}^{\infty} dt'' \int_{-\infty}^{t''} dt''' e^{i\omega_1(t'''-t')}e^{i\omega_2(t''-t)}F_1(t' - t_0)$$
$$\times F_2(t - t_0 - \Delta t)F_1(t''' - t_0)F_2(t'' - t_0 - \Delta t)\langle\Phi_0|e^{\frac{i}{\hbar}\hat{H}t'''}(\lambda\overrightarrow{\varepsilon_{01}} \cdot \hat{\mu})e^{-\frac{i}{\hbar}\hat{H}t'''}$$
$$\times e^{\frac{i}{\hbar}\hat{H}t}(\lambda\overrightarrow{\varepsilon_{02}} \cdot \hat{\mu})e^{-\frac{i}{\hbar}\hat{H}t}e^{\frac{i}{\hbar}\hat{H}t''}(\lambda\overrightarrow{\varepsilon_{02}} \cdot \hat{\mu})e^{-\frac{i}{\hbar}\hat{H}t''}e^{\frac{i}{\hbar}\hat{H}t'}(\lambda\overrightarrow{\varepsilon_{01}} \cdot \hat{\mu})e^{-\frac{i}{\hbar}\hat{H}t'}|\Phi_0\rangle, \tag{427}$$

and corresponds to the contribution due to absorption at ω_1, often promoting the system from the ground state to an intermediate excited state, followed by absorption at ω_2 to promote the system to a final state of even higher energy. At low temperature, this term often dominates the total transition probability since the integrands of off-resonant terms are more highly oscillatory. Equation (427) allows one to simulate pump-probe process in the time-dependent picture, as an alternative to density-matrix formulations [S. Mukamel, Principles of Nonlinear Optical Spectroscopy (Roxford University Press, New York, 1996)].

As an example, consider an experiment to probe the dynamics of a polyatomic system (I_2) in an excited electronic state B. The pump pulse of frequency ω_1 photoexcites the molecule from the ground state X to that excited electronic state B, and the probe pulse photoexcites the system from B to an even higher electronic state f as follows:

$$
\begin{aligned}
I_2(X) + \hbar\omega_1 &\to I_2(B), \\
I_2(B) + \hbar\omega_2 &\to I_2(f).
\end{aligned}
\tag{428}
$$

Considering that the electronic transition dipole moments are independent of nuclear coordinates, we obtain:

$$
\begin{aligned}
I(\omega_2, \Delta t) = \hbar^{-4} (\lambda \overrightarrow{\varepsilon_{01}} \cdot \overrightarrow{\mu_{BX}})^2 (\lambda \overrightarrow{\varepsilon_{02}} \cdot \overrightarrow{\mu_{fB}})^2 \sum_j \rho_j \int_{-\infty}^{\infty} dt \int_{-\infty}^{\infty} dt'' e^{-i\omega_2(t''-t)} \\
\times \int_{-\infty}^{t} dt' \int_{-\infty}^{t''} dt''' e^{\frac{i}{\hbar}(E_j + \hbar\omega_1)(t'''-t')} F_1(t'-t_0) F_2(t-t_0-\Delta t) \\
\times F_1(t'''-t_0) F_2(t''-t_0-\Delta t) \xi_j(t-t''', t''-t, t'-t''),
\end{aligned}
\tag{429}
$$

where

$$
\xi_j(t-t''', t''-t, t'-t'') = \langle \Phi_j | e^{-\frac{i}{\hbar}\hat{H}_B(t-t''')} e^{\frac{i}{\hbar}\hat{H}_f(t''-t)} e^{\frac{i}{\hbar}\hat{H}_B(t'-t'')} | \Phi_j \rangle.
\tag{430}
$$

If desired, Eq. (429) could also be written as

$$
I(\omega_2, \Delta t) = \int_{-\infty}^{\infty} dt\, e^{i\omega_2 t} C(t, \Delta t),
\tag{431}
$$

where $C(t, \Delta t)$ is readily identifiable from Eq. (429):

$$
\begin{aligned}
C(t, \Delta t) = \hbar^{-4} (\lambda \overrightarrow{\varepsilon_{01}} \cdot \overrightarrow{\mu_{BX}})^2 (\lambda \overrightarrow{\varepsilon_{02}} \cdot \overrightarrow{\mu_{fB}})^2 \sum_j \rho_j \int_{-\infty}^{\infty} dt'' \int_{-\infty}^{t''} dt' \int_{-\infty}^{t+t''} dt''' e^{\frac{i}{\hbar}(E_j + \hbar\omega_1)(t'''-t')} \\
\times F_1(t'-t_0) F_2(t''+t-t_0-\Delta t) F_1(t'''-t_0) F_2(t+t''-t_0-\Delta t) \\
\times \xi_j(t''-t''', t, t'-t''-t).
\end{aligned}
\tag{432}
$$

26 Pump-Probe Photoelectron Spectroscopy

Pump-probe photoelectron spectroscopy is essentially the same technique discussed Sec. 25 but using a probe pulse that can photodetach electrons. The pump-probe signal is reported in terms of the distribution $P(\epsilon, \Delta t)$ of kinetic energy ϵ of the photodetached electrons as a function of the time delay Δt between pump and probe pulses. The goal of this section is to show that these pump-probe photoelectron detachment signals can also be modeled by using the same general formalism of second order time dependent perturbation theory, introduced in Sec. 25, as shown in [Batista, V.S.; Zanni, M.T; Greenblatt, B.J.; Neumark, D.M.; Miller, W.H. J. Chem. Phys. 110, 3736-3747 (1999)].

As an example, we consider the photoelectron spectroscopy of I_2^- due to the photoelectron detachment process

$$
\begin{aligned}
I_2^-(X) + \hbar\omega_1 &\rightarrow I_2^-(A'), \\
I_2^-(A') + \hbar\omega_2 &\rightarrow I_2(K,\nu) + e^-(\epsilon),
\end{aligned}
\tag{433}
$$

where K and ν indicate the electronic and vibrational states of I_2, and ϵ the kinetic energy of the photodetached electron. The initial state of I_2^- is

$$
|\Phi_0\rangle = |\psi_g\rangle|\chi_g\rangle,
\tag{434}
$$

where $|\Psi_g\rangle$ is the ground (X) electronic state and $|\chi_g\rangle$ is the ground vibrational state of I_2^- in the X state. Final states are of the form

$$
|\Phi_f\rangle = |\psi_K\rangle|\chi_{E_K}\rangle,
\tag{435}
$$

where $|\psi_K\rangle$ is the electronic state K of I_2 and $|\chi_{E_K}\rangle$ the nuclear state of I_2. The corresponding initial and final energies are $E_0 = E_g$, and $E_f = E_K(\nu) + \epsilon$.

For simplicity, we consider only one intermediate state of the I_2^-, populated by the pump pulse: the A' excited state where the system evolves according to the the time evolution operator as follows,

$$
e^{-\frac{i}{\hbar}\hat{H}(t''-t')} = e^{-\frac{i}{\hbar}\hat{H}_{A'}(t''-t')}|\psi_{A'}\rangle\langle\psi_{A'}|,
\tag{436}
$$

Here, $|\psi_{A'}\rangle$ is the electronic wave function of I_2^- in the A' state and $\hat{H}_{A'}$ is the nuclear Hamiltonian for this electronic state.

According to Eq. (425), the transition probability to the final state $|\Phi_f\rangle$ due to a 2-photon excitation process is given by second order time-dependent perturbation theory as follows,

$$
P_K = \left| \hbar^{-2} \int_{-\infty}^{\infty} dt' \int_{-\infty}^{t'} dt'' e^{\frac{i}{\hbar}(E_K+\epsilon)t''} \langle\Phi_f|\hat{H}_1(t'')e^{-\frac{i}{\hbar}\hat{H}_{A'}(t''-t')}\hat{H}_1(t')e^{-\frac{i}{\hbar}\hat{H}t'}|\Phi_0\rangle \right|^2.
\tag{437}
$$

and the probability of a 2-photon transition to the electronic state K, leaving the photodetached electron with kinetic energy ϵ is

$$
P_K(\epsilon) = \hbar^{-4} \int dE_K \left| \int_{-\infty}^{\infty} dt' \int_{-\infty}^{t'} dt'' e^{\frac{i}{\hbar}(E_K+\epsilon)t''} \langle\Phi_f|\hat{H}_1(t'')e^{-\frac{i}{\hbar}\hat{H}(t''-t')}\hat{H}_1(t')e^{-\frac{i}{\hbar}\hat{H}t'}|\Phi_0\rangle \right|^2.
\tag{438}
$$

Therefore, the total probability of a 2-photon transition, leaving the photodetached electron with kinetic energy ϵ is $P(\epsilon) = \sum_K P_K(\epsilon)$:

$$
P(\epsilon) = \hbar^{-4} \sum_K \int dE_K \left| \int_{-\infty}^{\infty} dt' \int_{-\infty}^{t'} dt'' e^{\frac{i}{\hbar}(E_K+\epsilon)t''} \langle\Phi_f|\hat{H}_1(t'')e^{-\frac{i}{\hbar}\hat{H}_{A'}(t''-t')}\hat{H}_1(t')e^{-\frac{i}{\hbar}\hat{H}t'}|\Phi_0\rangle \right|^2.
\tag{439}
$$

102

Explicitly squaring the r.h.s. of Eq. (439) and using the relation,

$$\int dE_K |\chi_{E_K}\rangle\langle\chi_{E_K}|e^{-\frac{i}{\hbar}E_K(t''-t)} = e^{-\frac{i}{\hbar}H_K(t''-t)}, \tag{440}$$

we obtain the $(+\omega_1, +\omega_2)$ term, analogous to Eq. (427):

$$\begin{aligned}
P(\epsilon) = \hbar^{-4}\sum_K &\int_{-\infty}^{\infty} dt \int_{-\infty}^{t} dt' \int_{-\infty}^{\infty} dt'' \int_{-\infty}^{t''} dt''' e^{i\omega_1(t'''-t')}e^{i\omega_2(t''-t)}F_1(t'-t_0)F_2(t-t_0-\Delta t)\\
&\times F_1(t'''-t_0)F_2(t''-t_0-\Delta t)e^{\frac{i}{\hbar}\epsilon(t''-t)}\langle\Phi_0|e^{\frac{i}{\hbar}\hat{H}_X t'''}(\lambda\overrightarrow{\varepsilon_{01}}\cdot\overrightarrow{\mu}_{XA'})e^{-\frac{i}{\hbar}\hat{H}_{A'}t'''}\\
&\times e^{\frac{i}{\hbar}\hat{H}_{A'}t}(\lambda\overrightarrow{\varepsilon_{02}}\cdot\overrightarrow{\mu}_{A'K})e^{-\frac{i}{\hbar}\hat{H}_K t}e^{\frac{i}{\hbar}\hat{H}_K t''}(\lambda\overrightarrow{\varepsilon_{02}}\cdot\overrightarrow{\mu}_{KA'})e^{-\frac{i}{\hbar}\hat{H}_{A'}t''}\\
&\times e^{\frac{i}{\hbar}\hat{H}_{A'}t'}(\lambda\overrightarrow{\varepsilon_{01}}\cdot\overrightarrow{\mu}_{A'X})e^{-\frac{i}{\hbar}\hat{H}_X t'}|\Phi_0\rangle,
\end{aligned} \tag{441}$$

Here, we assumed that the transition dipole moments are independent of the kinetic energy of the photodetached electron. Here, we have also neglected all other 15 terms associated with the remaining pairs of interactions $(\pm\omega_j, \pm\omega_k)$ different from $(+\omega_1, +\omega_2)$, assuming the so-called 'rotating wave approximation' –i.e., that such other terms correspond to off-resonance transitions (absorptions, or emissions) for the specific example of I_2^-.

Finally, considering that the transition dipole moments are independent of nuclear coordinates, and that the system is prepared at finite temperature $T = 1/(\beta k_B)$, we obtain a compact expression of $P_\beta(\epsilon, \Delta t)$ analogous to Eq. (429):

$$\begin{aligned}
P_\beta(\epsilon, \Delta t) = \hbar^{-4}(\lambda\overrightarrow{\varepsilon_{01}}\cdot\overrightarrow{\mu_{A'X}})^2(\lambda\overrightarrow{\varepsilon_{02}}\cdot\overrightarrow{\mu_{KA'}})^2\sum_j\rho_j &\int_{-\infty}^{\infty} dt \int_{-\infty}^{\infty} dt'' e^{i(\omega_2+\epsilon)(t''-t)}\\
\times\int_{-\infty}^{t} dt' \int_{-\infty}^{t''} dt''' e^{\frac{i}{\hbar}(E_j+\hbar\omega_1)(t'''-t')}&F_1(t'-t_0)F_2(t-t_0-\Delta t)\\
\times F_1(t'''-t_0)F_2(t''-t_0-\Delta t)&\xi_j(t-t''', t''-t, t'-t''),
\end{aligned} \tag{442}$$

where

$$\xi_j(t-t''', t''-t, t'-t'') = \langle\Phi_j|e^{-\frac{i}{\hbar}\hat{H}_{A'}(t-t''')}e^{\frac{i}{\hbar}\hat{H}_K(t''-t)}e^{\frac{i}{\hbar}\hat{H}_{A'}(t'-t'')}|\Phi_j\rangle, \tag{443}$$

and $\rho_j = Z^{-1}exp(-\beta E_j)$, with $Z = \sum_j exp(-\beta E_j)$. If desired, Eq. (442) could also be written as

$$P_\beta(\epsilon, \Delta t) = \int_{-\infty}^{\infty} dt e^{i\epsilon t}C(t, \Delta t), \tag{444}$$

where $C(t, \Delta t)$ is readily identifiable from Eq. (442):

$$\begin{aligned}
C(t, \Delta t) = \hbar^{-4}(\lambda\overrightarrow{\varepsilon_{01}}\cdot\overrightarrow{\mu_{XB}})^2(\lambda\overrightarrow{\varepsilon_{02}}\cdot\overrightarrow{\mu_{BG}})^2\sum_j\rho_j &\int_{-\infty}^{\infty} dt'' \int_{-\infty}^{t''} dt' \int_{-\infty}^{t+t''} dt''' e^{\frac{i}{\hbar}(E_j+\hbar\omega_1)(t'''-t')}\\
\times F_1(t'-t_0)F_2(t''+t-t_0-\Delta t)&F_1(t'''-t_0)F_2(t+t''-t_0-\Delta t)e^{\frac{i}{\hbar}\omega_2 t}\\
\times \xi_j(t''-t''', t, t'-t''-t).&
\end{aligned} \tag{445}$$

27 Direct Photoelectron-Detachment Spectroscopy

The goal of this section is to show that the one-photon photoelectron detachment spectrum can be obtained according to the formalism introduced in Sec 23. As an example, we consider the direct photoelectron detachment spectroscopy of I_2^-, as studied in [*J. Chem. Phys.* (1999) **110**:3736],

$$I_2^-(X) + \hbar\omega_1 \rightarrow I_2(K, \nu) + e^-(\epsilon). \tag{446}$$

As in Sec. 26, the initial state of I_2^- is

$$|\Phi_0\rangle = |\psi_g\rangle|\chi_g\rangle, \tag{447}$$

where $|\Psi_g\rangle$ is the ground (X) electronic state and $|\chi_g\rangle$ is the ground vibrational state of I_2^- in the X state. Final states are of the form

$$|\Phi_f\rangle = |\psi_K\rangle|\chi_{E_K}\rangle, \tag{448}$$

where $|\psi_K\rangle$ is the electronic state K of I_2 and $|\chi_{E_K}\rangle$ the nuclear vibrational state of I_2. The corresponding initial and final energies are $E_0 = E_g$ and $E_f = E_K(\nu) + \epsilon$, respectively.

According to Eq. (415), the photoabsorption lineshape (at 0 K) for the one-photon photodetachment process is,

$$P_0(\epsilon) = \frac{3}{2\pi\hbar} \sum_K \int dE_K \int_{-\infty}^{\infty} dt e^{i\omega t} \langle\Phi_0|e^{\frac{i}{\hbar}\hat{H}t}(\lambda\overrightarrow{\varepsilon_0} \cdot \hat{\mu})e^{-\frac{i}{\hbar}(E_K+\epsilon)t}|\Phi_k\rangle\langle\Phi_k|(\lambda\overrightarrow{\varepsilon_0} \cdot \hat{\mu})|\Phi_0\rangle, \tag{449}$$

and using the relation

$$\int dE_K|\chi_{E_K}\rangle\langle\chi_{E_K}|e^{-\frac{i}{\hbar}E_K t} = e^{-\frac{i}{\hbar}H_K t}, \tag{450}$$

the finite temperature distribution is

$$P_\beta(\epsilon) = \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} dt e^{-\frac{i}{\hbar}\epsilon t}C(t), \tag{451}$$

with

$$C(t) = 3\sum_j \rho_j e^{\frac{i}{\hbar}(E_j+\hbar\omega t)} \sum_K \langle\Phi_j|(\lambda\overrightarrow{\varepsilon_0} \cdot \hat{\mu})e^{-\frac{i}{\hbar}\hat{H}_K t}(\lambda\overrightarrow{\varepsilon_0} \cdot \hat{\mu})|\Phi_j\rangle. \tag{452}$$

27.1 Computational Problem 17

17.1. Compute the photoabsorption spectrum of I_2. Assume that the transition dipole moments are independent of nuclear coordinates, and that the only allowed electronic transition induced by photoabsorption of I_2 is the $B \leftarrow X$ excitation. Assume the ground (g) and excited (e) states of I_2 can be described by the Morse Potential $V(R) = D_e\left(1 - e^{-\beta(R-R_{eq})}\right)^2 + V_0$, where R is the bond-length of I_2 and $V_0(g) = 0.00\ eV$; $V_0(e) = 0.94\ eV$; $D_e(g) = 18941\ cm^{-1}$; $D_e(e) = 4911\ cm^{-1}$; $\beta(g) = 1.517\ Å^{-1}$; $\beta(e) = 1.535\ Å^{-1}$; $R_{eq}(g) = 2.66\ Å$ and $R_{eq}(e) = 3.105\ Å$.

17.2. Compute the direct photoelectron detachment spectrum of I_2^- assuming that the electronic transitions induced by photoelectron detachment of I_2^-(X) generate I_2 in the electronic states X and B.

Assume that the potential energy surfaces of the states $I_2^-(X)$, $I_2(X)$ and $I_2(B)$ can be described by simple Morse potentials, as reported by Batista et al in [*J. Chem. Phys.* (1997) **106**:7102-7116] and [*J. Chem. Phys.* (1999) **110**:3736].

Solution in Sec. 61.24 .

28 Golden Rule

The goal of this section is to introduce the so-called *Fermi Golden Rule* expression, given by first-order time dependent perturbation theory.

We consider a system initially prepared in state $|i\rangle$. At time $t = 0$, we turn on the perturbation $W(t)$ and we analyze the decay to the final state $|f\rangle$, as described by first order time-dependent perturbation theory:

$$c_f(t) = -\frac{i}{\hbar} \int_0^t dt' \langle f|\hat{W}(t')|i\rangle e^{\frac{i}{\hbar}(E_f - E_i)t'}, \tag{453}$$

Therefore, the probability of observing the system in the final state is

$$P_{fi}(t) = \frac{1}{\hbar^2} \int_0^t dt'' \int_0^t dt' \langle i|\hat{W}^*(t'')|f\rangle \langle f|\hat{W}(t')|i\rangle e^{\frac{i}{\hbar}(E_f - E_i)(t' - t'')}, \tag{454}$$

28.1 Monochromatic Plane Wave

Assuming that the perturbation involves a single frequency component, $\hat{W}(t') = \hat{A}e^{-iwt'}$, we obtain:

$$\begin{aligned} c_f(t) &= \langle f|\hat{A}|i\rangle \frac{[1 - e^{i(w_{fi} - w)t}]}{\hbar(w_{fi} - \omega)}, \\ &= -\frac{i}{\hbar} t \langle f|\hat{A}|i\rangle e^{i(w_{fi} - w)t/2} \frac{\sin[(w_{fi} - w)t/2]}{(w_{fi} - \omega)t/2}. \end{aligned} \tag{455}$$

Therefore, the probability of observing the system in the final state is

$$P_{fi}(t) = \frac{t^2}{\hbar^2} |\langle f|\hat{A}|i\rangle|^2 \frac{\sin^2[(w_{fi} - w)t/2]}{[(w_{fi} - \omega)t/2]^2}. \tag{456}$$

To compute the survival probability that the system remains in the initial state, we must add up the probability over all possible final states,

$$\begin{aligned} P(t) &= 1 - \frac{t^2}{\hbar^2} \sum_f |\langle f|\hat{A}|i\rangle|^2 \frac{\sin^2[(w_{fi} - w)t/2]}{[(w_{fi} - \omega)t/2]^2} \\ &= 1 - \frac{t^2}{\hbar^2} \int_{-\infty}^{\infty} dE_f \rho(E_f)|\langle f|\hat{A}|i\rangle|^2 \frac{\sin^2[(w_{fi} - w)t/2]}{[(w_{fi} - \omega)t/2]^2} \end{aligned} \tag{457}$$

If the very short time limit, $P(t) = exp(-\alpha t^2) \approx 1 - \alpha t^2 + \cdots$, where

$$
\begin{aligned}
\alpha &= \lim_{t \to 0} \frac{1}{\hbar^2} \int_{-\infty}^{\infty} dE_f |\langle f|\hat{A}|i\rangle|^2 \rho(E_f) \frac{\sin^2[(E_f - E_i - \hbar w)t/(2\hbar)]}{[(E_f - E_i - \hbar w)t/(2\hbar)]^2}, \\
&= \frac{1}{\hbar^2} \int_{-\infty}^{\infty} dE_f |\langle f|\hat{A}|i\rangle|^2 \rho(E_f),
\end{aligned}
\tag{458}
$$

In the longer time limit, the kernel of Eq. (457) is approximated as the delta function to obtain:

$$
\begin{aligned}
P(t) &= 1 - \frac{t}{\hbar^2} \int_{-\infty}^{\infty} d(tE_f) \rho(E_f) |\langle f|\hat{A}|i\rangle|^2 \pi \delta((E_f t - (E_i + \hbar w)t)/(2\hbar)) \\
&= 1 - t\frac{2\pi}{\hbar} \int_{-\infty}^{\infty} d\xi \rho(\xi 2\hbar/t) |\langle f|\hat{A}|i\rangle|^2 \delta(\xi - (E_i + \hbar w)t/(2\hbar)) \\
&= 1 - t\frac{2\pi}{\hbar} \rho(E_i + \hbar w) |\langle E_i + \hbar w|\hat{A}|i\rangle|^2
\end{aligned}
\tag{459}
$$

so $P(t) = exp(-\Gamma t) \approx 1 - \Gamma t + \cdots$, where

$$
\begin{aligned}
\Gamma &= \frac{2\pi}{\hbar} \rho(E_i + \hbar w) |\langle E_i + \hbar w|\hat{A}|i\rangle|^2, \\
&= \frac{2\pi}{\hbar} \int_{-\infty}^{\infty} dE_f \rho(E_f) |\langle f|\hat{A}|i\rangle|^2 \delta(E_f - (E_i + \hbar w)),
\end{aligned}
\tag{460}
$$

or as a discrete sum over states,

$$
\Gamma = \frac{2\pi}{\hbar} \sum_f |\langle f|\hat{A}|i\rangle|^2 \delta(E_f - E_i - \hbar w),
\tag{461}
$$

which is known as Fermi's Golden rule.

Without introducing the approximation of the kernel of Eq. (457), we obtain:

$$
\begin{aligned}
P(t) &= 1 - \frac{t^2}{\hbar^2} \int_{-\infty}^{\infty} dE_f \rho(E_f) |\langle f|\hat{A}|i\rangle|^2 \frac{\sin^2[(w_{fi} - w)t/2]}{[(w_{fi} - \omega)t/2]^2} \\
&= 1 - t\frac{2}{\hbar} \int_{-\infty}^{\infty} dE_f \rho(E_f) \frac{t}{2\hbar} |\langle f|\hat{A}|i\rangle|^2 \frac{\sin^2((E_f - E_i - \hbar w)t/(2\hbar))}{((E_f - E_i - \hbar\omega)t/(2\hbar))^2} \\
&= 1 - t\frac{2}{\hbar} \int_{-\infty}^{\infty} d\xi \rho(\xi 2\hbar/t) |\langle f|\hat{A}|i\rangle|^2 \frac{\sin^2(\xi - (E_i + \hbar w)t/(2\hbar))}{(\xi - (E_i + \hbar\omega)t/(2\hbar))^2} \\
&= 1 - t\frac{2}{\hbar} \int_{-\infty}^{\infty} d\xi \rho(\xi 2\hbar/t) |\langle f|\hat{A}|i\rangle|^2 \frac{\sin^2(\xi)}{\xi^2}
\end{aligned}
\tag{462}
$$

which gives, in the time-range when the decay is exponential (i.e., $P(t) = exp(-\Gamma t) \approx 1 - \Gamma t$),

$$
\begin{aligned}
\Gamma &= \frac{2}{\hbar} \int_{-\infty}^{\infty} d\xi \rho(\xi 2\hbar/t) |\langle f|\hat{A}|i\rangle|^2 \frac{\sin^2(\xi)}{\xi^2}, \\
&= \frac{2}{\hbar} \int_{-\infty}^{\infty} dE_f \rho(E_f) |\langle f|\hat{A}|i\rangle|^2 \frac{\sin^2((E_f - E_i - \hbar w)t/(2\hbar))}{(E_f - E_i - \hbar\omega)^2 t/(2\hbar)}.
\end{aligned}
\tag{463}
$$

106

Substituting the delta function in Eq. (461) by its integral form, we obtain:

$$
\begin{aligned}
\Gamma_{fi} &= \frac{1}{\hbar^2} \int_{-\infty}^{\infty} dt \langle f|\hat{A}|i\rangle \langle i|\hat{A}|f\rangle e^{\frac{i}{\hbar}(E_f - E_i - \hbar w)t}, \\
&= \frac{1}{\hbar^2} \int_{-\infty}^{\infty} dt \langle f|e^{\frac{i}{\hbar}\hat{H}t}\hat{A}e^{-\frac{i}{\hbar}\hat{H}t}|i\rangle \langle i|\hat{A}|f\rangle e^{-iwt}, \\
&= \frac{1}{\hbar^2} \int_{-\infty}^{\infty} dt e^{-iwt} A_{fi}(t) A_{if}(0), \\
&= \frac{1}{\hbar^2} \int_{-\infty}^{\infty} dt e^{-iwt} A_{fi}(t) A_{if}(0).
\end{aligned}
\tag{464}
$$

The equilibrium ensemble average is

$$
\begin{aligned}
\langle \Gamma_{fi} \rangle &= \frac{1}{\hbar^2} \int_{-\infty}^{\infty} dt e^{-iwt} \sum_{\alpha} p_{\alpha} \langle \alpha|A_{fi}(t)A_{if}(0)|\alpha\rangle, \\
&= \frac{1}{\hbar^2} \int_{-\infty}^{\infty} dt e^{-iwt} \langle A_{fi}(t)A_{if}(0)\rangle, \\
&= \frac{[1 + e^{-\beta\hbar\omega}]^{-1}}{\hbar^2} \int_{-\infty}^{\infty} dt e^{-iwt} \langle [A_{fi}(t)A_{if}(0) + A_{fi}(0), A_{if}(t)]\rangle,
\end{aligned}
\tag{465}
$$

where $p_{\alpha} = Z^{-1}e^{-\beta E_{\alpha}}$ with $Z = \sum_{\alpha} e^{-\beta E_{\alpha}}$.

The rest of this subsection shows that, according to Eq. (465), $\langle \Gamma_{fi} \rangle$ can be written as follows:

$$
\langle \Gamma_{fi} \rangle = \frac{[1 + e^{-\beta\hbar\omega}]^{-1}}{\hbar^2} \int_{-\infty}^{\infty} dt e^{-iwt} \langle [A_{fi}(t)A_{if}(0) + A_{fi}(0), A_{if}(t)]\rangle,
\tag{466}
$$

where the symmetrized correlation function $C(t) = A_{fi}(t)A_{if}(0) + A_{fi}(0), A_{if}(t)$ is real, and is an even function of time just like its classical analogue correlation function. Therefore, Eq. (466) has often been used for estimations of $\langle \Gamma_{fi} \rangle$ based on classical simulations. However, it has been pointed out by Berne and co-workers that the classical version of $C(t)$ underestimates $\langle \Gamma_{fi} \rangle$ by a factor of $(\beta\hbar\omega/2)\coth(\beta\hbar\omega/2)$ [J. Chem. Phys. (1994) 100: 8359-8366].

The derivation of the last line of Eq. (466) is as follows:

$$
\begin{aligned}
\langle \Gamma_{fi} \rangle &= \frac{1}{\hbar^2} \int_{-\infty}^{\infty} dt e^{-iwt} \sum_{\alpha,\gamma} p_\alpha \langle \alpha | A_{fi}(t) | \gamma \rangle \langle \gamma | A_{if}(0) | \alpha \rangle, \\
&= \frac{1}{\hbar^2} \int_{-\infty}^{\infty} dt e^{-iwt} \frac{[1 + e^{-\beta\hbar\omega}]}{[1 + e^{-\beta\hbar\omega}]} \sum_{\alpha,\gamma} p_\alpha \langle \alpha | A_{fi}(t) | \gamma \rangle \langle \gamma | A_{if}(0) | \alpha \rangle, \\
&= \frac{2\pi}{\hbar} \frac{1}{[1 + e^{-\beta\hbar\omega}]} \sum_{\alpha,\gamma} p_\alpha \langle \alpha | A_{fi}(0) | \gamma \rangle \langle \gamma | A_{if}(0) | \alpha \rangle \delta(E_\gamma - E_\alpha - \hbar\omega)[1 + e^{-\beta\hbar\omega}], \\
&= \frac{2\pi}{\hbar} \frac{1}{[1 + e^{-\beta\hbar\omega}]} \sum_{\alpha,\gamma} p_\alpha \langle \alpha | A_{fi}(0) | \gamma \rangle \langle \gamma | A_{if}(0) | \alpha \rangle \delta(E_\gamma - E_\alpha - \hbar\omega)[1 + e^{-\beta(E_\gamma - E_\alpha)}], \\
&= \frac{[1 + e^{-\beta\hbar\omega}]^{-1}}{\hbar^2} \int_{-\infty}^{\infty} dt e^{-iwt} \sum_{\alpha,\gamma} p_\alpha \langle \alpha | A_{fi}(t) | \gamma \rangle \langle \gamma | A_{if}(0) | \alpha \rangle [1 + e^{-\beta(E_\gamma - E_\alpha)}], \\
&= \frac{[1 + e^{-\beta\hbar\omega}]^{-1}}{\hbar^2} \int_{-\infty}^{\infty} dt e^{-iwt} \left[\sum_{\alpha,\gamma} p_\alpha \langle \alpha | A_{fi}(t) | \gamma \rangle \langle \gamma | A_{if}(0) | \alpha \rangle \right. \\
&\qquad\qquad \left. + \sum_{\alpha,\gamma} p_\alpha \langle \alpha | A_{fi}(t) | \gamma \rangle \langle \gamma | A_{if}(0) | \alpha \rangle e^{-\beta(E_\gamma - E_\alpha)} \right], \\
&= \frac{[1 + e^{-\beta\hbar\omega}]^{-1}}{\hbar^2} \int_{-\infty}^{\infty} dt e^{-iwt} \left[\sum_{\alpha,\gamma} p_\alpha \langle \alpha | A_{fi}(t) | \gamma \rangle \langle \gamma | A_{if}(0) | \alpha \rangle \right. \\
&\qquad\qquad \left. + \sum_{\alpha,\gamma} p_\gamma \frac{p_\alpha}{p_\gamma} \langle \gamma | A_{if}(0) | \alpha \rangle \langle \alpha | A_{fi}(t) | \gamma \rangle e^{-\beta(E_\gamma - E_\alpha)} \right],
\end{aligned}
\tag{467}
$$

$$
\begin{aligned}
\langle \Gamma_{fi} \rangle &= \frac{[1 + e^{-\beta\hbar\omega}]^{-1}}{\hbar^2} \int_{-\infty}^{\infty} dt e^{-iwt} \left[\sum_{\alpha,\gamma} p_\alpha \langle \alpha | A_{fi}(t) | \gamma \rangle \langle \gamma | A_{if}(0) | \alpha \rangle + \sum_{\alpha,\gamma} p_\gamma \langle \gamma | A_{if}(0) | \alpha \rangle \langle \alpha | A_{fi}(t) | \gamma \rangle \right], \\
&= \frac{[1 + e^{-\beta\hbar\omega}]^{-1}}{\hbar^2} \int_{-\infty}^{\infty} dt e^{-iwt} \sum_{\alpha,\gamma} p_\alpha \left[\langle \alpha | A_{fi}(t) | \gamma \rangle \langle \gamma | A_{if}(0) | \alpha \rangle + \langle \alpha | A_{if}(0) | \gamma \rangle \langle \gamma | A_{fi}(t) | \alpha \rangle \right], \\
&= \frac{[1 + e^{-\beta\hbar\omega}]^{-1}}{\hbar^2} \int_{-\infty}^{\infty} dt e^{-iwt} \sum_{\alpha} p_\alpha \left[\langle \alpha | A_{fi}(t) A_{if}(0) + A_{if}(0) A_{fi}(t) | \alpha \rangle \right], \\
&= \frac{[1 + e^{-\beta\hbar\omega}]^{-1}}{\hbar^2} \int_{-\infty}^{\infty} dt e^{-iwt} \langle [A_{fi}(t) A_{if}(0) + A_{if}(0) A_{fi}(t)] \rangle, \\
&= \frac{[1 + e^{-\beta\hbar\omega_{fi}}]^{-1}}{\hbar^2} \int_{-\infty}^{\infty} dt e^{-iwt} \langle [A_{fi}(t) A_{if}(0) + A_{if}(0) A_{fi}(t)] \rangle,
\end{aligned}
\tag{468}
$$

28.2   Field with Exponential Decay

Assuming that the perturbation decays exponentially with a single frequency component, $\hat{W}(t') = \hat{A}e^{-t'/\tau}e^{-iwt'}$, we obtain the probability of observing the system in the final state:

$$
\begin{aligned}
P_{fi}(t) &= \frac{1}{\hbar^2}\int_0^t dt'' \int_0^t dt' \langle i|\hat{W}^*(t'')|f\rangle\langle f|\hat{W}(t')|i\rangle e^{\frac{i}{\hbar}(E_f-E_i)(t'-t'')}, \\
&= \frac{1}{\hbar^2}\int_0^t dt'' e^{-t''/\tau+i(\omega-\omega_{fi})t''}\int_0^t dt' \langle i|\hat{W}(0)^*|f\rangle\langle f|\hat{W}(t')|i\rangle e^{\frac{i}{\hbar}(E_f-E_i)t'}.
\end{aligned}
\tag{469}
$$

where $\omega_{fi} = (E_f - E_i)/\hbar$. Taking the limit $t \to \infty$, we obtain:

$$
\begin{aligned}
\lim_{t\to\infty} P_{fi}(t) &= \frac{1}{\hbar^2}\int_0^\infty dt'' e^{-t''/\tau+i(\omega-\omega_{fi})t''}\int_0^\infty dt' \langle i|\hat{W}^*(0)|f\rangle\langle f|\hat{W}(t')|i\rangle e^{\frac{i}{\hbar}(E_f-E_i)t'}, \\
&= \frac{1}{\hbar^2}\frac{e^{-t''/\tau+i\omega t''}}{(-1/\tau+i(\omega-\omega_{fi}))}\bigg|_0^\infty \int_0^\infty dt' \langle i|\hat{W}^*(0)|f\rangle\langle f|\hat{W}(t')|i\rangle e^{\frac{i}{\hbar}(E_f-E_i)t'}, \\
&= \frac{1}{\hbar^2}\frac{1}{(-1/\tau+i(\omega-\omega_{fi}))}\int_0^\infty dt' \langle i|\hat{A}^*|f\rangle\langle f|\hat{A}|i\rangle e^{-t''/\tau-i(\omega-\omega_{fi})t'}, \\
&= \frac{|\langle i|\hat{A}|f\rangle|^2}{\hbar^2(1/\tau^2-(\omega-\omega_{fi})^2)}
\end{aligned}
\tag{470}
$$

Therefore, the transfer probability from $|i\rangle$ to all possible final states $|f\rangle$ is:

$$
\lim_{t\to\infty} P(t) = \sum_f \frac{|\langle i|\hat{A}|f\rangle|^2}{\hbar^2(1/\tau^2-(\omega-\omega_{fi})^2)}
\tag{471}
$$

28.3   Vibrational Cooling

In the subsection, we illustrate the Golden Rule as applied to the description of vibrational cooling of a harmonic diatomic molecule coupled to a metal surface as implemented by Head-Gordon and Tully [*J. Chem. Phys.* (1992) **96**:3939-3949].

For a molecule interacting with a surface, the general form of the coupling matrix element A_{fi} follows from the Born-Oppenheimer approximation in which the nuclear kinetic energy term is neglected when obtaining the electronic wave functions. The states are Born-Oppenheimer products of nuclear (vibrational) and electronic wave functions which we write, as follows:

$$
\begin{aligned}
|i\rangle &= |\nu_i\rangle|e_i\rangle, \\
|f\rangle &= |\nu_f\rangle|e_f\rangle,
\end{aligned}
\tag{472}
$$

Therefore, the coupling between the two states is the matrix element of the nuclear kinetic-energy operator (for simplicity, we consider only a single normal mode x, which will be the high-frequency

adsorbate vibration):

$$\langle f|\hat{A}|i\rangle = -\frac{\hbar^2}{2m}\langle\nu_f e_f|\frac{\partial^2}{\partial x^2}|\nu_i e_i\rangle$$
$$= -\frac{\hbar^2}{2m}\left[\langle\nu_f|\nu_i\rangle\langle e_f|\frac{\partial^2}{\partial x^2}|e_i\rangle + 2\langle\nu_f e_f|\frac{\partial}{\partial x}|e_i\rangle\frac{\partial}{\partial x}|\nu_i\rangle\right] \tag{473}$$

where m is the reduced mass of the vibrational mode x.

To obtain a tractable expression, the nuclear and electronic coordinates in the matrix element must be separated. Following Brivio and Grimley (G. P. Brivio and T. B. Grimley, J. Phys. C 10, 2351 (1977); G. P. Brivio and T. B. Grimley, Surf. Sci. 89, 226 (1979)), this can be done by observing that since the amplitude of vibration is small (about 1/100 of the bondlength for diatomics in low vibrational states), it is reasonable to expand the electronic matrix elements in powers of the displacement x, keeping only the leading term. If we also assume that the electronic states $|e_i\rangle$ and $|e_f\rangle$ represent parallel potential-energy surfaces then $|\nu\rangle$ and $|\nu'\rangle$ will be orthogonal members of the same complete set, and the first term of Eq. (473) vanishes, leaving

$$\langle f|\hat{A}|i\rangle = -\frac{\hbar^2}{m}\left[\langle\nu_f|\frac{\partial}{\partial x}|\nu_i\rangle\langle e_f|\frac{\partial}{\partial x}|e_i\rangle\right] \tag{474}$$

To compute $\langle\nu_f|\frac{\partial}{\partial x}|\nu_i\rangle$, we assume that the vibrational states can be approximated as harmonic oscillator wave functionsm with frequency w, for which:

$$\hat{a}|\nu\rangle = \sqrt{\nu}|\nu - 1\rangle,$$
$$\hat{N}|\nu\rangle = \nu|\nu\rangle = \hat{a}^\dagger\hat{a}|\nu\rangle = \sqrt{\nu}\hat{a}^\dagger|\nu - 1\rangle, \tag{475}$$

so $\hat{a}^\dagger|\nu\rangle = \sqrt{\nu + 1}|\nu + 1\rangle$, with $\hat{a} = \frac{1}{\sqrt{2}}(\tilde{x} + i\tilde{p})$, $\hat{a}^\dagger = \frac{1}{\sqrt{2}}(\tilde{x} - i\tilde{p})$. Here, $\tilde{x} = \hat{x}\sqrt{\frac{mw}{\hbar}}$ and $\tilde{p} = \hat{p}\sqrt{\frac{1}{\hbar mw}}$, with $\hat{p} = -i\hbar\frac{\partial}{\partial x}$. Therefore, $\hat{a} - \hat{a}^\dagger = i\sqrt{\frac{2}{\hbar mw}}\hat{p} = \sqrt{\frac{2\hbar}{mw}}\frac{\partial}{\partial x}$ which gives,

$$\langle\nu_f|\frac{\partial}{\partial x}|\nu_i\rangle = \sqrt{\frac{mw}{2\hbar}}\left[\langle\nu_f|\hat{a}|\nu_i\rangle - \langle\nu_f|\hat{a}^\dagger|\nu_i\rangle\right],$$
$$= \sqrt{\frac{mw}{2\hbar}}\left[\sqrt{\nu_i}\delta_{\nu_f,\nu_i-1} - \sqrt{\nu_i + 1}\delta_{\nu_f,\nu_i+1}\right] \tag{476}$$

Substituting Eq. (476) into Eq. (474), with $\nu_f = \nu_i - 1$, we obtain:

$$\langle f|\hat{A}|i\rangle = -\hbar\sqrt{\frac{\hbar w\nu_i}{2m}}\langle e_f|\frac{\partial}{\partial x}|e_i\rangle. \tag{477}$$

and

$$|\langle f|\hat{A}|i\rangle|^2 = \hbar^2\frac{\hbar w\nu_i}{2m}|\langle e_f|\frac{\partial}{\partial x}|e_i\rangle|^2. \tag{478}$$

Substituting Eq. (478) into Eq. (461), we obtain:

$$\Gamma = \frac{\hbar}{m} \sum_f \hbar w \nu_i |\langle e_f| \frac{\partial}{\partial x} |e_i\rangle|^2 \delta(e_f - e_i - \hbar w), \tag{479}$$

Integrating over the number of electrons that could be promoted to an unoccupied state by absorbing a quantum of energy $\hbar w$, we obtain the total vibrational linewidth:

$$\Gamma = 2\frac{\hbar}{m} \int de_i \frac{\rho(e_i)}{e^{\beta(e_i - \mu_F)} + 1} \sum_f \hbar w \nu_i |\langle e_f| \frac{\partial}{\partial x} |e_i\rangle|^2 \delta(e_f - e_i - \hbar w),$$

$$= 2\frac{\hbar}{m} \int de_i \int de_f \frac{\rho(e_i)}{e^{\beta(e_i - \mu_F)} + 1} \frac{\rho(e_f)}{e^{-\beta(e_f - \mu_F)} + 1} \hbar w \nu_i |\langle e_f| \frac{\partial}{\partial x} |e_i\rangle|^2 \delta(e_f - e_i - \hbar w), \tag{480}$$

where μ_F is the Fermi energy. In the low-temperature limit ($\beta \to \infty$), for $\nu_i = 1$, we obtain:

$$\Gamma = 2\frac{\hbar}{m} \sum_{i<F} \sum_{f>F} \hbar w |\langle e_f| \frac{\partial}{\partial x} |e_i\rangle|^2 \delta(e_f - e_i - \hbar w) \tag{481}$$

Reinserting the kernel of Eq. (458) and introducing the change of variables $\Delta = e_f - e_i$, we can rewrite Eq. (480), as follows:

$$\Gamma = 2\frac{\hbar}{m} \int de_i \frac{\rho(e_i)}{e^{\beta(e_i - \mu_F)} + 1} \sum_f \hbar w \nu_i |\langle e_f| \frac{\partial}{\partial x} |e_i\rangle|^2 \delta(e_f - e_i - \hbar w),$$

$$= 2\frac{\hbar}{m} \sum_\Delta \sum_i \frac{\rho(e_i)}{e^{\beta(e_i - \mu_F)} + 1} \frac{\rho(e_i + \Delta)}{e^{-\beta(e_i + \Delta - \mu_F)} + 1} \hbar w \nu_i |\langle e_i + \Delta| \frac{\partial}{\partial x} |e_i\rangle|^2 \frac{\sin^2[(\Delta - \hbar w)t/(2\hbar)]}{[(\Delta - \hbar w)t/(2\hbar)]^2},$$

$$= \sum_\Delta f(\Delta) \frac{\sin^2[(\Delta - \hbar w)t/(2\hbar)]}{[(\Delta - \hbar w)t/(2\hbar)]^2}, \tag{482}$$

where

$$f(\Delta) = \sum_i \frac{\rho(e_i)}{e^{\beta(e_i - \mu_F)} + 1} \frac{\rho(e_i + \Delta)}{e^{-\beta(e_i + \Delta - \mu_F)} + 1} \hbar w \nu_i |\langle e_i + \Delta| \frac{\partial}{\partial x} |e_i\rangle|^2, \tag{483}$$

and t is a time sufficiently long so that the decay is no longer Gaussian but rather exponential so that Γ is time independent.

28.4   Electron Transfer

The rate of electron transfer from the initial state $|i\rangle = |\nu_i\rangle|e_i\rangle$ to the final state $|f\rangle = |\nu_f\rangle|e_f\rangle$ of weakly coupled redox species can also be described according to the Golden rule expression

derived from first order perturbation theory:

$$\Gamma_i \approx \frac{2\pi}{\hbar} \sum_f |\langle f|\hat{A}|i\rangle|^2 \delta(E_f - \hbar(w_i + w)),$$

$$\approx \frac{2\pi}{\hbar} |H_{if}|^2 \sum_f |\langle \nu_f|\nu_i\rangle|^2 \delta(E_f - E_i - \Delta E), \tag{484}$$

where $H_{if} = \langle e_f|\hat{A}|e_i\rangle$ is the coupling between electronic states, assumed to be independent of vibrational coordinates, and $\Delta E = \hbar\omega$ is the change in vibrational energy.

For the harmonic potentials shown in Fig. (28.4), it can be shown that

$$\sum_f |\langle \nu_f|\nu_i\rangle|^2 \delta(E_f - E_i - \Delta E) \approx \frac{1}{\hbar\Omega\pi x_{if}k^*}, \tag{485}$$

when $E_i, E_f > E^*$, with $k^* = \sqrt{2m(E_i - E^*)}/\hbar$ and $x_{if} = \sqrt{2E_\lambda/(m\Omega^2)}$, giving

$$\Gamma_i \approx 2|H_{if}|^2 \frac{1}{\sqrt{2\hbar E_\lambda/m}\sqrt{2m(E_i - E^*)}}$$

$$\approx |H_{if}|^2 \frac{1}{\hbar\sqrt{E_\lambda}\sqrt{(E_i - E^*)}} \tag{486}$$

Computing the thermal average over all initial states, we obtain the overall rate:

$$\Gamma \approx |H_{if}|^2 \frac{1}{\hbar\sqrt{E_\lambda}} \frac{\int dE_i exp(-\beta E_i)\frac{1}{\sqrt{(E_i - E^*)}}}{\int dE_i exp(-\beta E_i)}$$

$$\approx |H_{if}|^2 \frac{1}{\hbar\sqrt{E_\lambda}} exp(-\beta E^*) \frac{\int dE exp(-\beta E)\frac{1}{\sqrt{E}}}{\int dE exp(-\beta E)}$$

$$\approx |H_{if}|^2 \frac{1}{\hbar\sqrt{E_\lambda}} exp(-\beta E^*) \frac{\int_0^\infty dp 2p exp(-\beta p^2)\frac{1}{p}}{\int dE exp(-\beta E)}$$

$$\approx |H_{if}|^2 \frac{1}{\hbar\sqrt{E_\lambda}} exp(-\beta E^*) \frac{\int_0^\infty dp exp(-\beta p^2)}{\int dE exp(-\beta E)} \tag{487}$$

$$\approx |H_{if}|^2 \frac{1}{\hbar\sqrt{E_\lambda}} exp(-\beta E^*) \frac{\sqrt{\pi/\beta}}{\beta^{-1}}$$

$$\approx |H_{if}|^2 \frac{1}{\hbar\sqrt{E_\lambda}} exp(-\beta E^*)\sqrt{\pi\beta}$$

$$\approx |H_{if}|^2 \frac{\sqrt{2\pi\beta/m}}{\hbar\Omega x_{if}} exp(-\beta E^*)$$

Figure 28.4 shows a schematic description of the energy diagram, assuming that the potential energy surfaces along the vibronic coordinates are parabolas displaced in $x_{fi} = x_f - x_i$ with
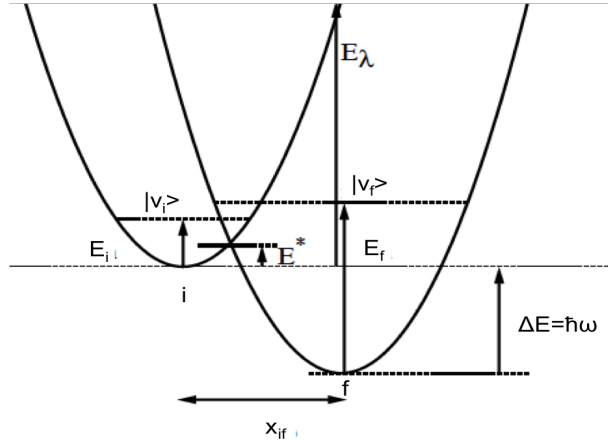
frequency Ω, we have $V_i = 1/2m\Omega^2(x - x_i)^2$ and $V_f = 1/2m\Omega^2(x - x_f)^2 - \Delta E$, which cross at x^* with energy

$$E^* = (E_\lambda - \Delta E)^2/(4E_\lambda). \tag{488}$$

To derive Eq. (488), we observe that $E^* = 1/2m\Omega^2(x^* - x_i)^2 = 1/2m\Omega^2(x^* - x_i - x_{fi})^2 - \Delta E$. Therefore, $E^* = 1/2m\Omega^2(x^* - x_i)^2 + 1/2m\Omega^2 x_{fi}^2 - m\Omega^2(x^* - x_i)x_{fi} - \Delta E$ which gives $E^* = E^* + E_\lambda - m\Omega^2(x^* - x_i)x_{fi} - \Delta E$. Simplifying, we obtain: $E_\lambda = m\Omega^2(x^* - x_i)x_{fi} + \Delta E = m\Omega^2\sqrt{2E^*/(m\Omega^2)}\sqrt{2E_\lambda/(m\Omega^2)} + \Delta E$.

28.4.1 Jortner Marcus Expression

The goal of this section is to introduce a simple expression for Monte Carlo calculations of electron transfer rates $k = A_N B_e$, as described by the Marcus-Jortner equation derived in J. Chem. Phys. (1978) 63:4358-4369, with the nuclear contribution

$$A_N = \sqrt{\frac{\pi}{\hbar^2 E_\lambda k_B T}} e^{-\frac{1}{2}\sum_{i=1}^{N}\Delta_{ci}^2} \sum_{\nu_1=0}^{\infty} \cdots \sum_{\nu_N=0}^{\infty} \prod_{i=1}^{N} \frac{(\Delta_{ci}^2)^{\nu_i}}{\nu_i!} exp\left(-\beta\left(\Delta E - E_\lambda - \sum_{i=1}^{N}\nu_i\hbar\omega_{ci}\right)^2 /(4E_\lambda)\right) \tag{489}$$

where $\Delta_{ci} = \Delta Q_i^0\sqrt{\mu_i\omega_{ci}/\hbar}$ are defined in terms of the displacements ΔQ_i^0 of the i-th normal mode minimum energy position. The electronic contribution is defined, as follows:

$$B_e = \int d\mathbf{R}|H_{ij}(\mathbf{R})|^2 \mathbf{exp}(-\beta\mathbf{E}(\mathbf{R})) \tag{490}$$

Introducing the sampling function,

$$f_\gamma(\nu_1, \cdots, \nu_N) = Z_\gamma^{-1} exp\left(-\gamma\sum_{i=1}^{N}\nu_i\hbar\omega_{ci}\right), \tag{491}$$

with $Z_\gamma = \prod_{i=1}^{N} \sum_{\nu_i=0}^{\infty} exp(-\gamma \nu_i \hbar \omega_{ci}) = \prod_{i=1}^{N}(1 - exp(-\gamma \hbar \omega_{ci}))^{-1}$, we obtain

$$A_N = \chi_\gamma \sum_{\nu_1=0}^{\infty} \cdots \sum_{\nu_N=0}^{\infty} \prod_{i=1}^{N} f_\gamma \frac{(\Delta_{ci}^2)^{\nu_i}}{\nu_i \ !} e^{-\beta\left(\Delta E - E_\lambda - \sum_{i=1}^{N} \nu_i \hbar \omega_{ci}\right)^2/(4E_\lambda)} e^{\gamma \sum_{i=1}^{N} \nu_i \hbar \omega_{ci}}, \tag{492}$$

with $\chi_\gamma = Z_\gamma \sqrt{\frac{\pi}{\hbar^2 E_\lambda k_B T}} e^{-\frac{1}{2} \sum_{i=1}^{N} \Delta_{ci}^2}$, which can be estimated by Monte Carlo, as follows:

$$A_N = \frac{\chi_\gamma}{N_c} \sum_{k=1}^{N_c} \prod_{i=1}^{N} \frac{(\Delta_{ci}^2)^{\nu_i^{(k)}}}{\nu_i^{(k)} \ !} e^{-\beta\left(\Delta E - E_\lambda - \sum_{i=1}^{N} \nu_i^{(k)} \hbar \omega_{ci}\right)^2/(4E_\lambda)} e^{\gamma \sum_{i=1}^{N} \nu_i \hbar \omega_{ci}}, \tag{493}$$

where the quantum numbers $\nu_j^{(k)}$ are sampled according to the distribution function f_γ.

28.4.2 Computational Problem: Jortner-Marcus

1. Write a program to implement the Monte Carlo estimator of A_N, introduced by Eq. (493) for a system with 4 normal modes with $\beta = 3.0$, frequencies $\omega_{ci} = 0.75$, reduced masses $\mu_i = 1.0$, displacements $\Delta Q_i^0 = 1.0$ and $E_\lambda = 0.25$ in atomic units (i.e., $\hbar = 1.0$). Compare your results for $N_c = 10^6$ with the corresponding calculations obtained by summing the contributions of each normal mode up to $\nu = 3$.
Solution: Sec. 61.28.

29　Intramolecular Vibrational Energy Redistribution

The dynamics of intramolecular vibrational energy redistribution can be simulated by using the anharmonic Hamiltonian written in terms of normal modes q_i, as follows:

$$\hat{H} = \sum_{i=1}^{N} \left(-\frac{1}{2} \frac{\partial^2}{\partial q_i^2} + \frac{1}{2} \omega_i^2 q_i^2 \right) + \hat{V} \tag{494}$$

where the anharmonic coupling term is

$$\hat{V} = \frac{1}{6} \sum_{ijk=1}^{N} \eta_{ijk} q_i q_j q_k + \frac{1}{24} \sum_{ijkl=1}^{N} \eta_{ijkl} q_i q_j q_k q_l. \tag{495}$$

We consider the 1-quantum basis,

$$\chi_j = \phi_j^{(1)}(q_j) \prod_{k \neq j}^{N} \phi_k^{(0)}(q_k), \tag{496}$$

with

$$\phi_j^{(0)}(q_j) = \left(\frac{\mu_j \omega_j}{\pi \hbar} \right)^{1/4} exp\left(-\frac{\mu_j \omega_j}{2\hbar} q_j^2 \right) \tag{497}$$

and

$$\phi_j^{(1)}(q_j) = \sqrt{\frac{2\mu_j\omega_j}{\hbar}}q_j\phi_j^{(0)}(q_j) \tag{498}$$

for which $\langle\phi_j^{(0)}|q_j|\phi_j^{(1)}\rangle = \sqrt{\frac{2\mu_j\omega_j}{\hbar}}\langle\phi_j^{(0)}|q_j^2|\phi_j^{(0)}\rangle = -\sqrt{\frac{2\mu_j\omega_j}{\hbar}}\sqrt{\frac{\mu_j\omega_j}{\pi\hbar}}\frac{\partial}{\partial(\mu_j\omega_j/\hbar)}\left(\frac{\mu_j\omega_j}{\pi\hbar}\right)^{-1/2} = \sqrt{\frac{2\mu_j\omega_j}{\hbar}}\frac{\hbar}{2\mu_j\omega_j} = $
$\sqrt{\frac{\hbar}{2\mu_j\omega_j}}$ and $\langle\phi_j^{(0)}|q_j^3|\phi_j^{(1)}\rangle = \sqrt{\frac{2\mu_j\omega_j}{\hbar}}\langle\phi_j^{(0)}|q_j^4|\phi_j^{(0)}\rangle = \sqrt{\frac{2\mu_j\omega_j}{\hbar}}\frac{\partial^2}{\partial(\mu_j\omega_j/\hbar)^2}\left(\frac{\mu_j\omega_j}{\pi\hbar}\right)^{-1/2} = -\sqrt{\frac{2\mu_j\omega_j}{\hbar}}\frac{\partial}{\partial(\mu_j\omega_j/\hbar)}\frac{\hbar}{2\mu_j\omega_j} = $
$\sqrt{\frac{2\mu_j\omega_j}{\hbar}}\frac{\hbar^2}{2\mu_j^2\omega_j^2} = \sqrt{\frac{\mu_j\omega_j\hbar^4}{2\hbar\mu_j^4\omega_j^4}} = \sqrt{\frac{\hbar^3}{2\mu_j^3\omega_j^3}}$.

Therefore,

$$H_{j,k} = \sum_{i=1}^{N}\hbar\omega_i(\frac{1}{2}+\delta_{\delta_{i,j}})\delta_{jk} + \langle\chi_j|\hat{V}|\chi_k\rangle, \tag{499}$$

with

$$\langle\chi_n|\hat{V}|\chi_m\rangle = \frac{1}{6}\sum_{ijk=1}^{N}\eta_{ijk}\langle\chi_n|q_iq_jq_k|\chi_m\rangle + \frac{1}{24}\sum_{ijkl=1}^{N}\eta_{ijkl}\langle\chi_n|q_iq_jq_kq_l|\chi_m\rangle,$$
$$= \frac{1}{6}V_{nm}^{(c)} + \frac{1}{24}V_{nm}^{(q)}. \tag{500}$$

In the 1-quantum basis $V_{nm}^{(c)} = 0$ and

$$V_{nm}^{(q)} = \delta_{nm}\sum_{jk=1}^{N}\eta_{jjkk}\langle\chi_n|q_j^2q_k^2|\chi_m\rangle + (1-\delta_{nm})\sum_{k=1}^{N}\eta_{nmkk}\langle\chi_n|q_nq_mq_k^2|\chi_m\rangle,$$

$$= \delta_{nm}\sum_{jk\neq n}^{N}\eta_{jjkk}\langle\chi_n|q_j^2q_k^2|\chi_m\rangle + 2\delta_{nm}\sum_{j\neq n}^{N}\eta_{jjnn}\langle\chi_n|q_j^2q_n^2|\chi_m\rangle + \delta_{nm}\eta_{nnnn}\langle\chi_n|q_n^4|\chi_m\rangle$$

$$+ (1-\delta_{nm})\sum_{k\neq(n,m)}^{N}\eta_{nmkk}\sqrt{\frac{\hbar}{2\mu_n\omega_n}}\sqrt{\frac{\hbar}{2\mu_m\omega_m}}\frac{\hbar}{2\mu_k\omega_k}$$

$$+ (1-\delta_{nm})(\eta_{nmmm}\langle\chi_n|q_nq_m^3|\chi_m\rangle + \eta_{nmnn}\langle\chi_n|q_mq_n^3|\chi_m\rangle) \tag{501}$$

$$V_{nm}^{(q)} = \delta_{nm}\sum_{jk\neq n}^{N}\eta_{jjkk}\frac{\hbar}{2\mu_j\omega_j}\frac{\hbar}{2\mu_k\omega_k} + \delta_{nm}\sum_{j=1}^{N}\eta_{jjnn}\frac{\hbar}{\mu_j\omega_j}\frac{\hbar}{\mu_n\omega_n}$$

$$+ (1-\delta_{nm})\sum_{k\neq(n,m)}^{N}\eta_{nmkk}\sqrt{\frac{\hbar}{2\mu_n\omega_n}}\sqrt{\frac{\hbar}{2\mu_m\omega_m}}\frac{\hbar}{2\mu_k\omega_k}$$

$$+ (1-\delta_{nm})\left(\eta_{nmmm}\sqrt{\frac{\hbar}{2\mu_n\omega_n}}\sqrt{\frac{\hbar^3}{2\mu_m^3\omega_m^3}} + \eta_{nmnn}\sqrt{\frac{\hbar}{2\mu_m\omega_m}}\sqrt{\frac{\hbar^3}{2\mu_n^3\omega_n^3}}\right)$$

115

29.1 Computational Problem: IVR

Given a system with 84 vibrational modes with frequencies ω_j given by the second column of the file freq.txt and the anharmonic coupling constants χ_{jk} given in the file xmat.txt. Initialize the system with a quantum of vibronic excitation in state $j = 9$ and evolve it in time by using the Lanczos propagation scheme in the Krylov basis as implemented in *Expokit*, assuming that state 13 has an absorbing potential. Display the evolution of populations in states j=9,10 and 11.
Solution: Sec. 61.29.

30 Classical Time-Dependent Perturbation Theory

The goal of this section is to derive the equations of classical time-dependent perturbation theory for the density of states $\rho(x, p; t)$, following a derivation analogous to the quantum version of time-dependent perturbation theory in dual space, introduced in Sec. (32) for the density matrix.

The aim is to find a solution of the Liouville equation:

$$i\frac{\partial \rho}{\partial t} = \hat{\mathcal{L}}\rho, \tag{502}$$

where

$$i\hat{\mathcal{L}} = \frac{\partial H}{\partial p}\frac{\partial}{\partial x} - \frac{\partial H}{\partial x}\frac{\partial}{\partial p}, \tag{503}$$

is the classical Liouvillian corresponding to the Hamiltonian $H(x, p) = H_0(x, p) + \lambda W(x; t)$, with $H_0 = \frac{p^2}{2m} + V(x)$ the Hamiltonian of the unperturbed system and $W(x; t)$ a time-dependent perturbation.

Therefore,

$$i\hat{\mathcal{L}} = i\hat{\mathcal{L}}_0 - \lambda\frac{\partial W}{\partial x}\frac{\partial}{\partial p},$$
$$= i\hat{\mathcal{L}}_0 + \lambda\hat{\mathcal{A}}, \tag{504}$$

where

$$i\hat{\mathcal{L}}_0 = \frac{\partial H_0}{\partial p}\frac{\partial}{\partial x} - \frac{\partial H_0}{\partial x}\frac{\partial}{\partial p}, \tag{505}$$

and

$$\hat{\mathcal{A}} = -\frac{\partial W}{\partial x}\frac{\partial}{\partial p}. \tag{506}$$

Expanding the density of states in powers of the small parameter λ, we obtain:

$$\rho = \rho^{(0)} + \lambda\rho^{(1)} + \lambda^2\rho^{(2)} + \cdots, \tag{507}$$

and substituting Eq. (507) into Eq. (502), we obtain:

$$\frac{\partial \rho^{(0)}}{\partial t} + \lambda\frac{\partial \rho^{(1)}}{\partial t} + \lambda^2\frac{\partial \rho^{(2)}}{\partial t}\cdots = -i\hat{\mathcal{L}}\rho,$$
$$= -i\hat{\mathcal{L}}_0\rho - \lambda\hat{\mathcal{A}}\rho,$$
$$= -i\hat{\mathcal{L}}_0[\rho^{(0)} + \lambda\rho^{(1)} + \lambda^2\rho^{(2)} + \cdots]$$
$$- \lambda\hat{\mathcal{A}}\left[\rho^{(0)} + \lambda\rho^{(1)} + \cdots\right]. \tag{508}$$

Making equal terms with equal powers of λ, we obtain:

$$\frac{\partial \rho^{(0)}}{\partial t} = -i\hat{\mathcal{L}}_0 \rho^{(0)},$$

$$\frac{\partial \rho^{(1)}}{\partial t} = -i\hat{\mathcal{L}}_0 \rho^{(1)} - \hat{\mathcal{A}}\rho^{(0)}, \tag{509}$$

$$\frac{\partial \rho^{(2)}}{\partial t} = -i\hat{\mathcal{L}}_0 \rho^{(2)} - \hat{\mathcal{A}}\rho^{(1)},$$

$$\ldots$$

According to the first row of Eq. (509),

$$\frac{d\rho^{(0)}}{dt} = \frac{\partial \rho^{(0)}}{\partial t} + \frac{\partial \rho^{(0)}}{\partial x}\dot{x} + \frac{\partial \rho^{(0)}}{\partial p}\dot{p} = 0, \tag{510}$$

along the phase-space characteristic curve for $x(t)$ and $p(t)$ defined by Hamilton's equations:

$$\dot{x} = \frac{\partial H_0}{\partial p},$$

$$\dot{p} = -\frac{\partial H_0}{\partial x}. \tag{511}$$

Therefore, the zero-th order density $\rho^{(0)}(x(t), p(t); t)$ can be obtained directly from the initial conditions $\rho^{(0)}(x(t), p(t); t) = \rho^{(0)}(x(0), p(0); 0)$, when $x(t)$ and $p(t)$ are obtained by intergrating the classical equations of motion introduced by Eq. (511) with initial conditions $x(0)$ and $p(0)$ by using 4th-order Runge-Kutta or Velocity Verlet.

According to the second row of Eq. (509),

$$\frac{d}{dt}\rho^{(1)}(x, p; t) = -\hat{\mathcal{A}}\rho^{(0)}(x, p; t), \tag{512}$$

which can be integrated after obtaining $\rho_p^{(0)}(x, p; t)$, as follows:

$$\rho^{(1)}(x, p; t) = -\int_0^t dt' \hat{\mathcal{A}}\rho^{(0)}(x, p; t'). \tag{513}$$

Analogously, higher order corrections are obtained as follows:

$$\rho^{(n)}(x, p; t) = -\int_0^t dt' \hat{\mathcal{A}}\rho^{(n-1)}(x, p; t'). \tag{514}$$

Therefore, according to the procedure described in this section, we can obtain the density of states ρ to order n, as introduced in Eq. (507), simply by propagating the initial density of states $\rho^{(0)}(x, p; 0)$ according to the unperturbed Hamiltonian H_0 to obtain $\rho^{(0)}(x, p; t)$. The resulting $\rho^{(0)}(x, p; t')$ obtained at intermediate times t' are used to compute the higher order corrections of Eq. (507) according to Eqs. (513) and (514).

30.1 Quantum Effects

According to Sec. 34.3, the Wigner transform ρ^W evolves fully quantum mechanically according to the Liouville equation, introduced by Eq. (502), so long as the Liouvillian operator is defined, as follows:

$$i\hat{\mathcal{L}} = i\hat{\mathcal{L}}_c + \lambda\hat{\mathcal{A}}, \tag{515}$$

where $\hat{\mathcal{L}}_c$ is the usual classical Liouvillian,

$$i\hat{\mathcal{L}}_c = \frac{p}{m}\frac{\partial}{\partial q} - \frac{\partial V(q)}{\partial q}\frac{\partial}{\partial p}, \tag{516}$$

and

$$\hat{\mathcal{A}} = -\frac{1}{\lambda}\sum_{j=1}^{\infty}\frac{(-1)^j}{(2j+1)!}\left(\frac{\hbar}{2}\right)^{2j}\frac{\partial^{2j+1}V(q)}{\partial q^{2j+1}}\frac{\partial^{2j+1}}{\partial p^{2j+1}}. \tag{517}$$

Comparing Eq. (515) with Eq. (504), we find that quantum dynamics can be described by classical time-dependent perturbation theory, after obtaining $\rho^{(0)}$ by integrating Hamilton's equations, simply by expanding ρ^W according to Eq. (507) and computing the corrections $\rho^{(1)}\cdots\rho^{(n)}$ according to Eqs. (513) and (514), with the operator $\hat{\mathcal{A}}$ defined according to Eq. (517).

31   Linear Response: Kubo Equation

The goal of this section is to introduce the so-called Kubo equation of linear response theory:

32   Perturbation Theory in Dual Space

The goal of this section is to derive the equations of time-dependent perturbation theory for the density matrix following the analogous derivation, introduced in Sec. (24) for the wavefunction picture.

The aim is to find a solution of the quantum Liouville equation:

$$i\hbar\frac{\partial\hat{\rho}}{\partial t} = \left[\hat{H}, \hat{\rho}\right], \tag{518}$$

where the Hamiltonian $\hat{H}$ is defined as

$$\begin{aligned}\hat{H} &= \hat{H}_0 + \lambda\hat{W}, \\ &= \hat{H}_0 + \hat{H}_1,\end{aligned} \tag{519}$$

where $\hat{H}_0$ is the Hamiltonian that describes the unperturbed system while $\hat{W}$ is the time-dependent perturbation.

Expanding the density matrix in powers of the small parameter λ, we obtain:

$$\hat{\rho} = \hat{\rho}^{(0)} + \lambda\hat{\rho}^{(1)} + \lambda^2\hat{\rho}^{(2)} + \cdots, \tag{520}$$

and substituting Eq. (520) into Eq. (518), we obtain:

$$
\begin{aligned}
i\hbar\frac{\partial}{\partial t}\hat{\rho}^{(0)} + i\hbar\lambda\frac{\partial}{\partial t}\hat{\rho}^{(1)} + i\hbar\lambda^2\frac{\partial}{\partial t}\hat{\rho}^{(2)} + \cdots &= \left[\hat{H}, \hat{\rho}\right], \\
&= \left[(\hat{H}_0 + \lambda\hat{W}), (\hat{\rho}^{(0)} + \lambda\hat{\rho}^{(1)} + \cdots)\right], \\
&= \left[\hat{H}_0, \hat{\rho}^{(0)}\right] + \lambda\left(\left[\hat{W}, \hat{\rho}^{(0)}\right] + \left[\hat{H}_0, \hat{\rho}^{(1)}\right]\right) \\
&\quad + \lambda^2\left(\left[\hat{W}, \hat{\rho}^{(1)}\right] + \left[\hat{H}_0, \hat{\rho}^{(2)}\right]\right) + \cdots,
\end{aligned}
\tag{521}
$$

Making equal terms with equal powers of λ, we obtain:

$$
\begin{aligned}
i\hbar\frac{\partial}{\partial t}\hat{\rho}^{(0)} &= \left[\hat{H}_0, \hat{\rho}^{(0)}\right], \\
i\hbar\frac{\partial}{\partial t}\hat{\rho}^{(1)} &= \left[\hat{W}, \hat{\rho}^{(0)}\right] + \left[\hat{H}_0, \hat{\rho}^{(1)}\right], \\
i\hbar\frac{\partial}{\partial t}\hat{\rho}^{(2)} &= \left[\hat{W}, \hat{\rho}^{(1)}\right] + \left[\hat{H}_0, \hat{\rho}^{(2)}\right], \\
&\cdots
\end{aligned}
\tag{522}
$$

To solve Eq. (522), we expand $\hat{\rho}^{(n)}$ in Fourier series, as follows:

$$
\begin{aligned}
\hat{\rho}^{(n)}(t) &= \sum_j e^{-i\omega_j t}\hat{\rho}^{(n)}(\omega_j), \\
&= \sum_j \hat{\rho}^{(n)}(\omega_j, t).
\end{aligned}
\tag{523}
$$

Substituting Eq. (523) into Eq. (522), we obtain:

$$
\hbar\omega_j\hat{\rho}^{(0)}(\omega_j, t) = \hat{H}_0\hat{\rho}^{(0)}(\omega_j, t) - \hat{\rho}^{(0)}(\omega_j, t)\hat{H}_0,
\tag{524}
$$

Evaluating Eq. (524) in the basis set of the eigenstates $|n\rangle$ of the unperturbed Hamiltonian satisfying the eigenvalue problem $\hat{H}_0|n\rangle = E_n|n\rangle$, we obtain:

$$
\begin{aligned}
\hbar\omega_j\rho_{n,n'}^{(0)}(\omega_j, t) &= E_n\rho_{n,n'}^{(0)}(\omega_j, t) - \rho_{n,n'}^{(0)}(\omega_j, t)E_{n'}, \\
(\hbar\omega_j + E_{n'} - E_n)\,\rho_{n,n'}^{(0)}(\omega_j, t) &= 0,
\end{aligned}
\tag{525}
$$

which is satisfied by $\hbar\omega_j = E_n - E_{n'}$, or $\hbar\omega_j \neq E_n - E_{n'}$ and $\hat{\rho}_{n,n'}^{(0)}(\omega_j, t) = 0$.

Analogously, we obtain:

$$
\hbar\omega_j\hat{\rho}^{(1)}(\omega_j, t) = \hat{H}_0\hat{\rho}^{(1)}(\omega_j, t) - \hat{\rho}^{(1)}(\omega_j, t)\hat{H}_0 + \hat{W}(\omega_k, t)\hat{\rho}^{(0)}(\omega_l, t) - \hat{\rho}^{(0)}(\omega_l, t)\hat{W}(\omega_k, t)
\tag{526}
$$

with $\omega_j = \omega_k + \omega_l$ and

$$
\begin{aligned}
\hbar\omega_j\rho_{n,n'}^{(1)}(\omega_j, t) &= E_n\rho_{n,n'}^{(1)}(\omega_j, t) - \rho_{n,n'}^{(1)}(\omega_j, t)E_{n'} \\
&\quad + \sum_m W_{n,m}(\omega_k, t)\rho_{m,n'}^{(0)}(\omega_l, t) - \rho_{n,m}^{(0)}(\omega_l, t)W_{m,n'}(\omega_k, t)
\end{aligned}
\tag{527}
$$

giving

$$\rho_{n,n'}^{(1)}(\omega_k + \omega_l, t) = \sum_m \frac{W_{n,m}(\omega_k, t)\rho_{m,n'}^{(0)}(\omega_l, t) - \rho_{n,m}^{(0)}(\omega_l, t)W_{m,n'}(\omega_k, t)}{E_{n'} - E_n + \hbar(\omega_k + \omega_l)} \tag{528}$$

Considering that $\rho_{n,m}^{(0)} = \rho_{n,n}^{(0)}\delta_{nm}$ and $\hbar\omega_l = E_n - E_m$, we obtain that $\omega_l = 0$ and

$$\rho_{n,n'}^{(1)}(\omega_k, t) = \frac{\rho_{n',n'}^{(0)} - \rho_{n,n}^{(0)}}{E_{n'} - E_n + \hbar\omega_k}W_{n,n'}(\omega_k, t) \tag{529}$$

Therefore, the first order polarization $\hat{P} = N\hat{\mu}$, with $\hat{\mu} = -e\hat{r}$, is

$$
\begin{aligned}
P^{(1)}(\omega_k, t) &= N \sum_{n,n'} \mu_{n',n}\rho_{n,n'}^{(1)}(\omega_k, t) \\
&= N \sum_{n,n'} \mu_{n',n}\frac{\rho_{n',n'}^{(0)} - \rho_{n,n}^{(0)}}{E_{n'} - E_n + \hbar\omega_k}W_{n,n'}(\omega_k, t)
\end{aligned} \tag{530}
$$

Considering that the perturbation is due to a dipolar interaction $\hat{W}(\omega_k, t) = -\hat{\mu}\cdot\epsilon(\omega_k, t)$, we obtain:

$$
\begin{aligned}
P^{(1)}(\omega_k, t) &= N \sum_{n,n'} \frac{\rho_{n,n}^{(0)} - \rho_{n',n'}^{(0)}}{E_{n'} - E_n + \hbar\omega_k}\mu_{n',n}\mu_{n,n'} \cdot \epsilon(\omega_k, t) \\
&= \chi^{(1)}(\omega_k) \cdot \epsilon(\omega_k, t)
\end{aligned} \tag{531}
$$

where

$$
\begin{aligned}
\chi_{pq}^{(1)}(\omega_k) &= N \sum_{g,n} \frac{\rho_{g,g}^{(0)} - \rho_{n,n}^{(0)}}{E_n - E_g + \hbar\omega_k}\mu_{n,g}^p\mu_{g,n}^q \\
&= N \sum_g \sum_n \frac{\mu_{n,g}^p\mu_{g,n}^q}{E_n - E_g + \hbar\omega_k}\rho_{g,g}^{(0)} - N \sum_g \sum_n \frac{\mu_{n,g}^p\mu_{g,n}^q}{E_n - E_g + \hbar\omega_k}\rho_{n,n}^{(0)} \\
&= N \sum_n \rho_{g,g}^{(0)} \sum_n \frac{\mu_{n,g}^p\mu_{g,n}^q}{E_n - E_g + \hbar\omega_k} - N \sum_n \rho_{n,n}^{(0)} \sum_g \frac{\mu_{n,g}^p\mu_{g,n}^q}{E_n - E_g + \hbar\omega_k} \\
&= N \sum_g \rho_{g,g}^{(0)} \sum_n \frac{\mu_{n,g}^p\mu_{g,n}^q}{E_n - E_g + \hbar\omega_k} - N \sum_g \rho_{g,g}^{(0)} \sum_n \frac{\mu_{g,n}^p\mu_{n,g}^q}{E_g - E_n + \hbar\omega_k} \\
&= N \sum_g \rho_{g,g}^{(0)} \sum_n \frac{\mu_{n,g}^p\mu_{g,n}^q}{E_n - E_g + \hbar\omega_k} - N \sum_g \rho_{g,g}^{(0)} \sum_n \frac{\mu_{g,n}^p\mu_{n,g}^q}{E_g - E_n + \hbar\omega_k} \\
&= N \sum_g \rho_{g,g}^{(0)} \sum_n \frac{\mu_{n,g}^p\mu_{g,n}^q}{E_n - E_g + \hbar\omega_k} + \frac{\mu_{n,g}^q\mu_{g,n}^p}{E_n - E_g - \hbar\omega_k}
\end{aligned} \tag{532}
$$

121

or

$$\chi_{pq}^{(1)}(\omega_k) = N\hbar^{-1} \sum_{g,n} \rho_{g,g}^{(0)} \left[\frac{\mu_{n,g}^p \mu_{g,n}^q}{\omega_{ng} + \omega_k} + \frac{\mu_{n,g}^q \mu_{g,n}^p}{\omega_{ng} - \omega_k} \right]$$
$$= \hbar^{-1} \sum_{g,n} \rho_{g,g}^{(0)} \alpha_{g,n}^{pq}(\omega_k) \tag{533}$$

We note that $\alpha_{g,n}^{pq*} = \alpha_{n,g}^{pq}$ since

$$\alpha_{g,n}^{pq*} = \left[\frac{\mu_{n,g}^p \mu_{g,n}^q}{\omega_{ng} + \omega_k} + \frac{\mu_{n,g}^q \mu_{g,n}^p}{\omega_{ng} - \omega_k} \right]^*$$
$$= \left[\frac{\mu_{g,n}^p \mu_{n,g}^q}{\omega_{gn} + \omega_k} + \frac{\mu_{g,n}^q \mu_{n,g}^p}{\omega_{gn} - \omega_k} \right] \tag{534}$$
$$= \alpha_{n,g}^{pq}$$

Furthermore,

$$\hbar\omega_j \hat{\rho}^{(2)}(\omega_j, t) = \hat{H}_0 \hat{\rho}^{(2)}(\omega_j, t) - \hat{\rho}^{(2)}(\omega_j, t) \hat{H}_0$$
$$+ \hat{W}(\omega_k, t) \hat{\rho}^{(1)}(\omega_l, t) - \hat{\rho}^{(1)}(\omega_l, t) \hat{W}(\omega_k, t) \tag{535}$$
$$+ \hat{W}(\omega_l, t) \hat{\rho}^{(1)}(\omega_k, t) - \hat{\rho}^{(1)}(\omega_k, t) \hat{W}(\omega_l, t)$$

with $\omega_j = \omega_k + \omega_l$. Therefore,

$$\hbar\omega_j \rho_{n,n'}^{(2)}(\omega_j, t) = E_n \rho_{n,n'}^{(2)}(\omega_j, t) - \rho_{n,n'}^{(2)}(\omega_j, t) E_{n'}$$
$$+ \sum_m W_{n,m}(\omega_k, t) \rho_{m,n'}^{(1)}(\omega_l, t) - \rho_{n,m}^{(1)}(\omega_l, t) W_{m,n'}(\omega_k, t)$$
$$+ \sum_m W_{n,m}(\omega_l, t) \rho_{m,n'}^{(1)}(\omega_k, t) - \rho_{n,m}^{(1)}(\omega_k, t) W_{m,n'}(\omega_l, t) \tag{536}$$
$$\left(\hbar\omega_j - E_n + E_{n'}\right) \rho_{n,n'}^{(2)}(\omega_j, t) = \sum_m W_{n,m}(\omega_k, t) \rho_{m,n'}^{(1)}(\omega_l, t) - \rho_{n,m}^{(1)}(\omega_l, t) W_{m,n'}(\omega_k, t)$$
$$+ \sum_m W_{n,m}(\omega_l, t) \rho_{m,n'}^{(1)}(\omega_k, t) - \rho_{n,m}^{(1)}(\omega_k, t) W_{m,n'}(\omega_l, t)$$

which gives

$$\rho_{n,n'}^{(2)}(\omega_j, t) = \frac{1}{(\hbar\omega_j - E_n + E_{n'})} \sum_m W_{n,m}(\omega_k, t) W_{m,n'}(\omega_l, t) \frac{\rho_{n',n'}^{(0)} - \rho_{m,m}^{(0)}}{E_{n'} - E_m + \hbar\omega_l}$$
$$- W_{n,m}(\omega_l, t) W_{m,n'}(\omega_k, t) \frac{\rho_{m,m}^{(0)} - \rho_{n,n}^{(0)}}{E_m - E_n + \hbar\omega_k}$$
$$+ W_{n,m}(\omega_l, t) W_{m,n'}(\omega_k, t) \frac{\rho_{n',n'}^{(0)} - \rho_{m,m}^{(0)}}{E_{n'} - E_m + \hbar\omega_k} \tag{537}$$
$$- W_{n,m}(\omega_k, t) W_{m,n'}(\omega_l, t) \frac{\rho_{m,m}^{(0)} - \rho_{n,n}^{(0)}}{E_m - E_n + \hbar\omega_l}$$

and the second order polarization

$$
\begin{aligned}
P^{(2)}(\omega_j, t) = N \sum_{m,n,n'} & \mu_{n',n} \frac{\mu_{n,m} \cdot \epsilon(\omega_k, t)\mu_{m,n'} \cdot \epsilon(\omega_l, t)}{(\hbar\omega_j - E_n + E_{n'})(E_{n'} - E_m + \hbar\omega_l)}(\rho^{(0)}_{n',n'} - \rho^{(0)}_{m,m}) \\
& - \mu_{n',n} \frac{\mu_{n,m} \cdot \epsilon(\omega_l, t)\mu_{m,n'} \cdot \epsilon(\omega_k, t)}{(\hbar\omega_j - E_n + E_{n'})(E_m - E_n + \hbar\omega_k)}(\rho^{(0)}_{m,m} - \rho^{(0)}_{n,n}) \\
& + \mu_{n',n} \frac{\mu_{n,m} \cdot \epsilon(\omega_l, t)\mu_{m,n'} \cdot \epsilon(\omega_k, t)}{(\hbar\omega_j - E_n + E_{n'})(E_{n'} - E_m + \hbar\omega_k)}(\rho^{(0)}_{n',n'} - \rho^{(0)}_{m,m}) \\
& - \mu_{n',n} \frac{\mu_{n,m} \cdot \epsilon(\omega_k, t)\mu_{m,n'} \cdot \epsilon(\omega_l, t)}{(\hbar\omega_j - E_n + E_{n'})(E_m - E_n + \hbar\omega_l)}(\rho^{(0)}_{m,m} - \rho^{(0)}_{n,n})
\end{aligned}
\tag{538}
$$

Distributing terms, we obtain:

$$
\begin{aligned}
P^{(2)}(\omega_j, t) = N \sum_{m,n,n'} & \frac{\mu_{n',n}\mu_{n,m} \cdot \epsilon(\omega_k, t)\mu_{m,n'} \cdot \epsilon(\omega_l, t)}{(\hbar\omega_j - E_n + E_{n'})(E_{n'} - E_m + \hbar\omega_l)}\rho^{(0)}_{n',n'} \\
& - \frac{\mu_{n',n}\mu_{n,m} \cdot \epsilon(\omega_k, t)\mu_{m,n'} \cdot \epsilon(\omega_l, t)}{(\hbar\omega_j - E_n + E_{n'})(E_{n'} - E_m + \hbar\omega_l)}\rho^{(0)}_{m,m} \\
& - \frac{\mu_{n',n}\mu_{n,m} \cdot \epsilon(\omega_l, t)\mu_{m,n'} \cdot \epsilon(\omega_k, t)}{(\hbar\omega_j - E_n + E_{n'})(E_m - E_n + \hbar\omega_k)}\rho^{(0)}_{m,m} \\
& + \frac{\mu_{n',n}\mu_{n,m} \cdot \epsilon(\omega_l, t)\mu_{m,n'} \cdot \epsilon(\omega_k, t)}{(\hbar\omega_j - E_n + E_{n'})(E_m - E_n + \hbar\omega_k)}\rho^{(0)}_{n,n} \\
& + \frac{\mu_{n',n}\mu_{n,m} \cdot \epsilon(\omega_l, t)\mu_{m,n'} \cdot \epsilon(\omega_k, t)}{(\hbar\omega_j - E_n + E_{n'})(E_{n'} - E_m + \hbar\omega_k)}\rho^{(0)}_{n',n'} \\
& - \frac{\mu_{n',n}\mu_{n,m} \cdot \epsilon(\omega_l, t)\mu_{m,n'} \cdot \epsilon(\omega_k, t)}{(\hbar\omega_j - E_n + E_{n'})(E_{n'} - E_m + \hbar\omega_k)}\rho^{(0)}_{m,m} \\
& - \frac{\mu_{n',n}\mu_{n,m} \cdot \epsilon(\omega_k, t)\mu_{m,n'} \cdot \epsilon(\omega_l, t)}{(\hbar\omega_j - E_n + E_{n'})(E_m - E_n + \hbar\omega_l)}\rho^{(0)}_{m,m} \\
& + \frac{\mu_{n',n}\mu_{n,m} \cdot \epsilon(\omega_k, t)\mu_{m,n'} \cdot \epsilon(\omega_l, t)}{(\hbar\omega_j - E_n + E_{n'})(E_m - E_n + \hbar\omega_l)}\rho^{(0)}_{n,n}
\end{aligned}
\tag{539}
$$

and

$$\chi_{p,q,r}^{(2)}(\omega_j) = N \sum_{m,n,n'} \left[\rho_{n',n'}^{(0)} \frac{\mu_{n',n}^p \mu_{n,m}^q \mu_{m,n'}^r}{(\hbar\omega_j - E_n + E_{n'})(E_{n'} - E_m + \hbar\omega_l)} \right.$$

$$-\rho_{m,m}^{(0)} \frac{\mu_{n',n}^p \mu_{n,m}^q \mu_{m,n'}^r}{(\hbar\omega_j - E_n + E_{n'})(E_{n'} - E_m + \hbar\omega_l)}$$

$$-\rho_{m,m}^{(0)} \frac{\mu_{n',n}^p \mu_{n,m}^q \mu_{m,n'}^r}{(\hbar\omega_j - E_n + E_{n'})(E_m - E_n + \hbar\omega_k)}$$

$$+\rho_{n,n}^{(0)} \frac{\mu_{n',n}^p \mu_{n,m}^q \mu_{m,n'}^r}{(\hbar\omega_j - E_n + E_{n'})(E_m - E_n + \hbar\omega_k)}$$

$$+\rho_{n',n'}^{(0)} \frac{\mu_{n',n}^p \mu_{n,m}^r \mu_{m,n'}^q}{(\hbar\omega_j - E_n + E_{n'})(E_{n'} - E_m + \hbar\omega_k)} \qquad (540)$$

$$-\rho_{m,m}^{(0)} \frac{\mu_{n',n}^p \mu_{n,m}^r \mu_{m,n'}^q}{(\hbar\omega_j - E_n + E_{n'})(E_{n'} - E_m + \hbar\omega_k)}$$

$$-\rho_{m,m}^{(0)} \frac{\mu_{n',n}^p \mu_{n,m}^r \mu_{m,n'}^q}{(\hbar\omega_j - E_n + E_{n'})(E_m - E_n + \hbar\omega_l)}$$

$$\left. +\rho_{n,n}^{(0)} \frac{\mu_{n',n}^p \mu_{n,m}^r \mu_{m,n'}^q}{(\hbar\omega_j - E_n + E_{n'})(E_m - E_n + \hbar\omega_l)} \right]$$

Gathering terms, we obtain:

$$\chi_{p,q,r}^{(2)}(\omega_j) = N \sum_{m,n,n'} \rho_{m,m}^{(0)} \left[\frac{\mu_{m,n}^p \mu_{n,n'}^q \mu_{n',m}^r}{(\hbar\omega_j - E_n + E_m)(E_m - E_{n'} + \hbar\omega_l)} \right.$$

$$+\frac{\mu_{m,n'}^q \mu_{n',n}^r \mu_{n,m}^p}{(\hbar\omega_j - E_m + E_n)(E_{n'} - E_m + \hbar\omega_k)}$$

$$+\frac{\mu_{m,n}^p \mu_{n,n'}^r \mu_{n',m}^q}{(\hbar\omega_j - E_n + E_m)(E_m - E_{n'} + \hbar\omega_k)} \qquad (541)$$

$$+\frac{\mu_{n',m}^p \mu_{n,n'}^q \mu_{m,n}^r}{(\hbar\omega_j - E_m + E_{n'})(E_n - E_m + \hbar\omega_l)}$$

$$-\frac{\mu_{m,n'}^r \mu_{n',n}^p \mu_{n,m}^q}{(\hbar\omega_j - E_n + E_{n'})} \left[\frac{1}{(E_{n'} - E_m + \hbar\omega_l)} + \frac{1}{(E_m - E_n + \hbar\omega_k)} \right]$$

$$\left. -\frac{\mu_{m,n'}^q \mu_{n',n}^p \mu_{n,m}^r}{(\hbar\omega_j - E_n + E_{n'})} \left[\frac{1}{(E_m - E_n + \hbar\omega_l)} + \frac{1}{(E_{n'} - E_m + \hbar\omega_k)} \right] \right]$$

and simplifying the last two terms after swapping indices n and n', we obtain:

$$
\chi^{(2)}_{p,q,r}(\omega_j) = N \sum_{m,n,n'} \rho^{(0)}_{m,m} \left[\frac{\mu^p_{m,n}\mu^q_{n,n'}\mu^r_{n',m}}{(\hbar\omega_j - E_n + E_m)(E_m - E_{n'} + \hbar\omega_l)} \right.
$$

$$
+ \frac{\mu^q_{m,n'}\mu^r_{n',n}\mu^p_{n,m}}{(\hbar\omega_j - E_m + E_n)(E_{n'} - E_m + \hbar\omega_k)}
$$

$$
+ \frac{\mu^p_{m,n}\mu^r_{n,n'}\mu^q_{n',m}}{(\hbar\omega_j - E_n + E_m)(E_m - E_{n'} + \hbar\omega_k)}
$$

$$
+ \frac{\mu^p_{n',m}\mu^q_{n,n'}\mu^r_{m,n}}{(\hbar\omega_j - E_m + E_{n'})(E_n - E_m + \hbar\omega_l)}
$$

$$
- \frac{\mu^r_{m,n}\mu^p_{n,n'}\mu^q_{n',m}}{(E_n - E_m + \hbar\omega_l)(E_m - E_{n'} + \hbar\omega_k)}
$$

$$
\left. - \frac{\mu^q_{m,n}\mu^p_{n,n'}\mu^r_{n',m}}{(E_m - E_{n'} + \hbar\omega_l)(E_n - E_m + \hbar\omega_k)} \right] \tag{542}
$$

which can be grouped into two terms that are resonant with ω_l and two non-resonant terms, as follows:

$$
\chi^{(2)}_{p,q,r}(\omega_j) = N \sum_{m,n,n'} \rho^{(0)}_{m,m} \left[\frac{\mu^r_{n',m}}{(E_m - E_{n'} + \hbar\omega_l)} \left[\frac{\mu^p_{m,n}\mu^q_{n,n'}}{(\hbar\omega_j - E_n + E_m)} - \frac{\mu^q_{m,n}\mu^p_{n,n'}}{(E_n - E_m + \hbar\omega_k)} \right] \right.
$$

$$
+ \frac{\mu^r_{m,n}}{(E_n - E_m + \hbar\omega_l)} \left[\frac{\mu^p_{n',m}\mu^q_{n,n'}}{(\hbar\omega_j - E_m + E_{n'})} - \frac{\mu^q_{n,n'}\mu^p_{n',m}}{(E_m - E_{n'} + \hbar\omega_k)} \right]
$$

$$
+ \frac{\mu^q_{m,n'}\mu^r_{n',n}\mu^p_{n,m}}{(\hbar\omega_j - E_m + E_n)(E_{n'} - E_m + \hbar\omega_k)}
$$

$$
\left. + \frac{\mu^p_{m,n}\mu^r_{n,n'}\mu^q_{n',m}}{(\hbar\omega_j - E_n + E_m)(E_m - E_{n'} + \hbar\omega_k)} \right] \tag{543}
$$

or

$$
\chi^{(2)}_{p,q,r}(\omega_j) = N \sum_{g,n,m} \rho^{(0)}_{g,g} \left[\frac{\mu^r_{m,g}}{(E_g - E_m + \hbar\omega_l)} \left[\frac{\mu^p_{g,n}\mu^q_{n,m}}{(\hbar\omega_j - E_n + E_g)} - \frac{\mu^q_{g,n}\mu^p_{n,m}}{(E_n - E_g + \hbar\omega_k)} \right] \right.
$$

$$
+ \frac{\mu^r_{g,n}}{(E_n - E_g + \hbar\omega_l)} \left[\frac{\mu^p_{m,g}\mu^q_{n,m}}{(\hbar\omega_j - E_g + E_m)} - \frac{\mu^p_{n,m}\mu^q_{m,g}}{(E_g - E_m + \hbar\omega_k)} \right]
$$

$$
+ \frac{\mu^q_{g,m}\mu^r_{m,n}\mu^p_{n,g}}{(\hbar\omega_j - E_g + E_n)(E_m - E_g + \hbar\omega_k)}
$$

$$
\left. + \frac{\mu^p_{g,n}\mu^r_{n,m}\mu^q_{m,g}}{(\hbar\omega_j - E_n + E_g)(E_g - E_m + \hbar\omega_k)} \right] \tag{544}
$$

Assuming that $\omega_l << \omega_k$ (e.g., ω_l IR, ω_k Vis) then $\omega_j \approx \omega_k$, so we obtain:

$$\chi_{p,q,r}^{(2)}(\omega_j) = N \sum_{g,n,m} \rho_{g,g}^{(0)} \left[\frac{\mu_{m,g}^r}{(E_g - E_m + \hbar\omega_l)} \left[\frac{\mu_{g,n}^p \mu_{n,m}^q}{(\hbar\omega_k - E_n + E_g)} - \frac{\mu_{g,n}^q \mu_{n,m}^p}{(E_n - E_g + \hbar\omega_k)} \right] \right.$$
$$+ \frac{\mu_{g,n}^r}{(E_n - E_g + \hbar\omega_l)} \left[\frac{\mu_{m,g}^p \mu_{n,m}^q}{(\hbar\omega_k - E_g + E_m)} - \frac{\mu_{n,m}^p \mu_{m,g}^q}{(E_g - E_m + \hbar\omega_k)} \right]$$
$$+ \frac{\mu_{g,m}^q \mu_{m,n}^r \mu_{n,g}^p}{(\hbar\omega_j - E_g + E_n)(E_m - E_g + \hbar\omega_k)}$$
$$\left. + \frac{\mu_{g,n}^p \mu_{n,m}^r \mu_{m,g}^q}{(\hbar\omega_j - E_n + E_g)(E_g - E_m + \hbar\omega_k)} \right] \tag{545}$$

which can be simplified, according to Eq. 533, as follows:

$$\chi_{p,q,r}^{(2)}(\omega_j) = -\frac{1}{\hbar} \sum_{g,m} \rho_{g,g}^{(0)} \frac{\alpha_{g,m}^{pq}(\omega_k) \mu_{m,g}^r}{(E_g - E_m + \hbar\omega_l)} + \frac{1}{\hbar} \sum_{g,n} \rho_{g,g}^{(0)} \frac{\mu_{g,n}^r \alpha_{n,g}^{pq}(\omega_k)}{(E_n - E_g + \hbar\omega_l)}$$
$$+ N \sum_{g,n,m} \rho_{g,g}^{(0)} \left[\frac{\mu_{g,m}^q \mu_{m,n}^r \mu_{n,g}^p}{(\hbar\omega_j - E_g + E_n)(E_m - E_g + \hbar\omega_k)} \right. \tag{546}$$
$$\left. + \frac{\mu_{g,n}^p \mu_{n,m}^r \mu_{m,g}^q}{(\hbar\omega_j - E_n + E_g)(E_g - E_m + \hbar\omega_k)} \right]$$

Assuming that the system is initially in a harmonic well, we expand the dipole and polarizability in powers of normal mode displacements, as follows:

$$\hat{\mu}^r = \hat{\mu}^r(\zeta_0) + \sum_j \left. \frac{\partial \hat{\mu}^r}{\partial \zeta(j)} \right|_{\zeta(j)=\zeta_0(j)} (\zeta(j) - \zeta_0(j)) + \cdots,$$
$$\hat{\alpha}^{pq} = \hat{\alpha}^{pq}(\zeta_0) + \sum_j \left. \frac{\partial \hat{\alpha}^{pq}}{\partial \zeta(j)} \right|_{\zeta(j)=\zeta_0(j)} (\zeta(j) - \zeta_0(j)) + \cdots, \tag{547}$$

with $\langle g|n\rangle = \delta_{gn}$, and $\langle g|(\zeta(j) - \zeta_0(j))|n\rangle = \sqrt{\frac{\hbar}{2m_j\omega_j}}[\delta_{(g-1)n} + \delta_{(g+1)n}]$ for harmonic states $|g\rangle$ and $|n\rangle$. Therefore, we obtain the harmonic approximation:

$$\chi_{p,q,r}^{(2)}(\omega_l) \approx -\sum_j \frac{1}{2m_j\omega_j} \frac{1}{\hbar(\omega_l - \omega_j)} \left. \frac{\partial \hat{\alpha}^{pq}}{\partial \zeta(j)} \right|_{\zeta(j)=\zeta_0(j)} \left. \frac{\partial \hat{\mu}^r}{\partial \zeta(j)} \right|_{\zeta(j)=\zeta_0(j)}$$
$$+ \sum_j \frac{1}{2m_j\omega_j} \frac{1}{\hbar(\omega_l + \omega_j)} \left. \frac{\partial \hat{\alpha}^{pq}}{\partial \zeta(j)} \right|_{\zeta(j)=\zeta_0(j)} \left. \frac{\partial \hat{\mu}^r}{\partial \zeta(j)} \right|_{\zeta(j)=\zeta_0(j)}$$
$$+ N \sum_{g,n,m} \rho_{g,g}^{(0)} \left[\frac{\mu_{g,m}^q \mu_{m,n}^r \mu_{n,g}^p}{(\hbar\omega_j - E_g + E_n)(E_m - E_g + \hbar\omega_k)} \right. \tag{548}$$
$$\left. + \frac{\mu_{g,n}^p \mu_{n,m}^r \mu_{m,g}^q}{(\hbar\omega_j - E_n + E_g)(E_g - E_m + \hbar\omega_k)} \right]$$

32.1 Correlation Functions

The goal of this subsection is to show that first and second order susceptibilities can be evaluated in terms of correlation functions, as follows:

$$
\chi_{p,q,r}^{(2)}(\omega_j) = \frac{2}{\hbar^2} \int_0^\infty dt e^{i\omega_l t} \Im[\langle \mu^r(0)\alpha^{pq}(\omega_k, t)\rangle]
$$
$$
+ N \sum_{g,n,m} \rho_{g,g}^{(0)} \left[\frac{\mu_{g,m}^q \mu_{m,n}^r \mu_{n,g}^p}{(\hbar\omega_j - E_g + E_n)(E_m - E_g + \hbar\omega_k)} \right.
$$
$$
\left. + \frac{\mu_{g,n}^p \mu_{n,m}^r \mu_{m,g}^q}{(\hbar\omega_j - E_n + E_g)(E_g - E_m + \hbar\omega_k)} \right],
$$

(549)

and

$$
\chi_{p,q}^{(1)}(\omega_k) = 2N\hbar^{-1} \int_0^\infty dt e^{-i\omega_k t} \Im[\langle \mu^p(0)\mu^q(t)\rangle]
$$

(550)

To derive Eq. (549) from Eq. (548), we introduce the equality

$$
\frac{i}{\hbar} \int_0^\infty dt e^{-\frac{i}{\hbar}(\hbar\omega_l - (E_m - E_g))t} = \frac{i}{\hbar} \lim_{\gamma \to 0} \int_0^\infty dt e^{-\frac{i}{\hbar}(\hbar\omega_l - (E_m - E_g))t} e^{-\gamma t}
$$
$$
= \frac{i}{\hbar} \lim_{\gamma \to 0} \int_0^\infty dt e^{-\frac{i}{\hbar}(\hbar\omega_l - (E_m - E_g))t} e^{-\gamma t}
$$
$$
= \frac{i}{\hbar} \lim_{\gamma \to 0} \int_0^\infty dt e^{-(\gamma + \frac{i}{\hbar}(\hbar\omega_l - (E_m - E_g)))t}
$$
$$
= -\frac{i}{\hbar} \lim_{\gamma \to 0} \left. \frac{e^{-(\gamma + \frac{i}{\hbar}(\hbar\omega_l - (E_m - E_g)))t}}{(\gamma + \frac{i}{\hbar}(\hbar\omega_l - (E_m - E_g)))} \right|_0^\infty = \frac{1}{(E_g - E_m + \hbar\omega_l)}
$$

(551)

and substituting Eq. (551) into Eq. (548), we obtain:

$$
\chi_{p,q,r}^{(2)}(\omega_j) = \frac{i}{\hbar^2} \int_0^\infty dt e^{i\omega_l t} \sum_{g,m} \rho_{g,g}^{(0)} e^{\frac{i}{\hbar}E_g t} \alpha_{g,m}^{pq}(\omega_k) e^{-\frac{i}{\hbar}E_m t} \mu_{m,g}^r
$$
$$
- \frac{i}{\hbar^2} \int_0^\infty dt e^{i\omega_l t} \sum_{g,n} \rho_{g,g}^{(0)} \mu_{g,n}^r e^{\frac{i}{\hbar}E_n t} \alpha_{n,g}^{pq}(\omega_k) e^{-\frac{i}{\hbar}E_g t}
$$
$$
+ N \sum_{g,n,m} \rho_{g,g}^{(0)} \left[\frac{\mu_{g,m}^q \mu_{m,n}^r \mu_{n,g}^p}{(\hbar\omega_j - E_g + E_n)(E_m - E_g + \hbar\omega_k)} \right.
$$
$$
\left. + \frac{\mu_{g,n}^p \mu_{n,m}^r \mu_{m,g}^q}{(\hbar\omega_j - E_n + E_g)(E_g - E_m + \hbar\omega_k)} \right]
$$

(552)

127

Introducing the correlation function

$$\langle \mu^r(0)\alpha^{pq}(\omega_k,t)\rangle = \sum_{g,n} \rho_{g,g}^{(0)} \mu_{g,n}^r e^{\frac{i}{\hbar}E_n t} \alpha_{n,g}^{pq}(\omega_k) e^{-\frac{i}{\hbar}E_g t},$$

$$= \sum_{g,n} \rho_{g,g}^{(0)} \mu_{g,n}^r \alpha_{n,g}^{pq}(\omega_k,t),$$

(553)

with $\hat{\alpha}^{pq}(\omega_k,t) = e^{\frac{i}{\hbar}\hat{H}t}\hat{\alpha}^{pq}(\omega_k)e^{-\frac{i}{\hbar}\hat{H}t}$, we note that according to Eq. (534), $\alpha_{g,n}^{pq*} = \alpha_{n,g}^{pq}$, so

$$\langle \mu^r(0)\alpha^{pq}(\omega_k,t)\rangle^* = \sum_{g,n} \rho_{g,g}^{(0)} \mu_{n,g}^r e^{-\frac{i}{\hbar}E_n t} \alpha_{g,n}^{pq}(\omega_k) e^{\frac{i}{\hbar}E_g t}$$

$$= \sum_{g,n} \rho_{g,g}^{(0)} e^{\frac{i}{\hbar}E_g t} \alpha_{g,n}^{pq}(\omega_k) e^{-\frac{i}{\hbar}E_n t} \mu_{n,g}^r$$

(554)

$$= \langle \alpha^{pq}(\omega_k,t)\mu^r(0)\rangle$$

so we can simplify Eq. (552), as follows:

$$\chi_{p,q,r}^{(2)}(\omega_j) = -\frac{i}{\hbar^2} \int_0^\infty dt e^{i\omega_l t}[-\langle \mu^r(0)\alpha^{pq}(\omega_k,t)\rangle^* + \langle \mu^r(0)\alpha^{pq}(\omega_k,t)\rangle]$$

$$+ N \sum_{g,n,m} \rho_{g,g}^{(0)} \left[\frac{\mu_{g,m}^q \mu_{m,n}^r \mu_{n,g}^p}{(\hbar\omega_j - E_g + E_n)(E_m - E_g + \hbar\omega_k)} \right.$$

(555)

$$\left. + \frac{\mu_{g,n}^p \mu_{n,m}^r \mu_{m,g}^q}{(\hbar\omega_j - E_n + E_g)(E_g - E_m + \hbar\omega_k)} \right]$$

so we obtain:

$$\chi_{p,q,r}^{(2)}(\omega_j) = \frac{2}{\hbar^2} \int_0^\infty dt e^{i\omega_l t} \Im[\langle \mu^r(0)\alpha^{pq}(\omega_k,t)\rangle]$$

$$+ N \sum_{g,n,m} \rho_{g,g}^{(0)} \left[\frac{\mu_{g,m}^q \mu_{m,n}^r \mu_{n,g}^p}{(\hbar\omega_j - E_g + E_n)(E_m - E_g + \hbar\omega_k)} \right.$$

(556)

$$\left. + \frac{\mu_{g,n}^p \mu_{n,m}^r \mu_{m,g}^q}{(\hbar\omega_j - E_n + E_g)(E_g - E_m + \hbar\omega_k)} \right]$$

Equation (550) can be derived from Eq. (533) analogously, as follows:

$$\chi_{p,q}^{(1)}(\omega_k) = N\hbar^{-1} \sum_{g,n} \rho_{g,g}^{(0)} \left[\frac{\mu_{n,g}^p \mu_{g,n}^q}{\omega_{ng} + \omega_k} + \frac{\mu_{n,g}^q \mu_{g,n}^p}{\omega_{ng} - \omega_k} \right]$$

$$= iN\hbar^{-1} \sum_{g,n} \rho_{g,g}^{(0)} \int_0^\infty dt e^{-i(\omega_{ng}+\omega_k)t} \mu_{n,g}^p \mu_{g,n}^q$$

(557)

$$- iN\hbar^{-1} \sum_{g,n} \rho_{g,g}^{(0)} \int_0^\infty dt e^{i(\omega_{ng}-\omega_k)t} \mu_{n,g}^q \mu_{g,n}^p$$

128

$$\chi_{p,q}^{(1)}(\omega_k) = iN\hbar^{-1}\sum_{g,n}\rho_{g,g}^{(0)}\int_0^\infty dt e^{-i\omega_k t}\mu_{n,g}^p(0)\mu_{g,n}^q(t)$$

$$- iN\hbar^{-1}\sum_{g,n}\rho_{g,g}^{(0)}\int_0^\infty dt e^{-i\omega_k}\mu_{g,n}^p(0)\mu_{n,g}^q(t)$$

$$= iN\hbar^{-1}\sum_g p_g \int_0^\infty dt e^{-i\omega_k t}\langle\phi_g|e^{\frac{i}{\hbar}\hat{H}t}\hat{\mu}^q e^{-\frac{i}{\hbar}\hat{H}t}\hat{\mu}^p|\phi_g\rangle \qquad (558)$$

$$- iN\hbar^{-1}\sum_g p_g \int_0^\infty dt e^{-i\omega_k}\langle\phi_g|\hat{\mu}^p e^{\frac{i}{\hbar}\hat{H}t}\hat{\mu}^q e^{-\frac{i}{\hbar}\hat{H}t}|\phi_g\rangle$$

$$= iN\hbar^{-1}\int_0^\infty dt e^{-i\omega_k t}\sum_g p_g\langle\phi_g|[\hat{\mu}^q(t),\hat{\mu}^p(0)]|\phi_g\rangle$$

$$\chi_{p,q}^{(1)}(\omega_k) = iN\hbar^{-1}\sum_{g,n}\rho_{g,g}^{(0)}\int_0^\infty dt e^{-i\omega_k t}\mu_{g,n}^p(0)^*\mu_{n,g}^q(t)^*$$

$$- iN\hbar^{-1}\sum_{g,n}\rho_{g,g}^{(0)}\int_0^\infty dt e^{-i\omega_k}\mu_{g,n}^p(0)\mu_{n,g}^q(t)$$

$$= iN\hbar^{-1}\int_0^\infty dt e^{-i\omega_k t}[\langle\mu^p(0)\mu^q(t)\rangle^* - \langle\mu^p(0)\mu^q(t)\rangle] \qquad (559)$$

$$= -iN\hbar^{-1}\int_0^\infty dt e^{-i\omega_k t}[-\langle\mu^p(0)\mu^q(t)\rangle^* + \langle\mu^p(0)\mu^q(t)\rangle]$$

$$= 2N\hbar^{-1}\int_0^\infty dt e^{-i\omega_k t}\Im[\langle\mu^p(0)\mu^q(t)\rangle]$$

Another equivalent expression for the second order susceptibility can be obtained from Eq. (544), as follows:

$$\chi_{p,q,r}^{(2)}(\omega_j) = N\sum_{g,n,m}\rho_{g,g}^{(0)}\left[\frac{\mu_{m,g}^r}{(E_g - E_m + \hbar\omega_l)}\left[\frac{\mu_{g,n}^p\mu_{n,m}^q}{(\hbar\omega_j - E_n + E_g)} - \frac{\mu_{g,n}^q\mu_{n,m}^p}{(E_n - E_g + \hbar\omega_k)}\right]\right.$$

$$+ \frac{\mu_{g,n}^r}{(E_n - E_g + \hbar\omega_l)}\left[\frac{\mu_{m,g}^p\mu_{n,m}^q}{(\hbar\omega_j - E_g + E_m)} - \frac{\mu_{n,m}^p\mu_{m,g}^q}{(E_g - E_m + \hbar\omega_k)}\right]$$

$$+ \frac{\mu_{g,m}^q\mu_{m,n}^r\mu_{n,g}^p}{(\hbar\omega_j - E_g + E_n)(E_m - E_g + \hbar\omega_k)}$$

$$\left.+ \frac{\mu_{g,n}^p\mu_{n,m}^r\mu_{m,g}^q}{(\hbar\omega_j - E_n + E_g)(E_g - E_m + \hbar\omega_k)}\right] \qquad (560)$$

$$\chi_{p,q,r}^{(2)}(\omega_j) = N \sum_{g,n,m} \rho_{g,g}^{(0)} \left[\frac{\mu_{m,g}^r}{(E_g - E_m + \hbar\omega_l)} \left[\frac{\mu_{g,n}^p \mu_{n,m}^q}{(\hbar\omega_j - E_n + E_g)} - \frac{\mu_{g,n}^q \mu_{n,m}^p}{(E_n - E_g + \hbar\omega_k)} \right] \right.$$

$$+ \frac{\mu_{g,n}^r}{(E_n - E_g + \hbar\omega_l)} \left[\frac{\mu_{m,g}^p \mu_{n,m}^q}{(\hbar\omega_j - E_g + E_m)} - \frac{\mu_{n,m}^p \mu_{m,g}^q}{(E_g - E_m + \hbar\omega_k)} \right] \qquad (561)$$

$$+ \frac{\mu_{g,m}^q \mu_{m,n}^r \mu_{n,g}^p}{(\hbar\omega_j - E_g + E_n)(E_m - E_g + \hbar\omega_k)}$$

$$\left. + \frac{\mu_{g,n}^p \mu_{n,m}^r \mu_{m,g}^q}{(\hbar\omega_j - E_n + E_g)(E_g - E_m + \hbar\omega_k)} \right]$$

33   The Reaction Surface Hamiltonian Method

The goal of this section is to describe the reaction surface Hamiltonian method, introduced by Carrington and Miller (*J. Chem. Phys.* (1986) **84**:4364-4370) to construct *ab initio* Hamiltonians for quantum dynamics simulations, as recently reported in several studies of hydrogen transfer (e.g., [*Phys. Rep.* (2006) **430**:211-276], [*J. Chem. Phys.* (1999) **110**:9922-9936] and [*J. Chem. Phys.* (2000) **113**:9510-9522]).

Methods for simulations of quantum dynamics in polyatomic systems (*i.e.*, N atom systems) require multidimensional potential energy surfaces to describe the energy of the system as a function of the $3N-6$ independent coordinates ξ_j, with $j=1,...,3N-6$. When the system remains near its equilibrium configuration, one can assume that the motion results from small amplitude displacements of its normal mode coordinates relative to their equilibrium configurations ξ_j^0. The energy of those configurations can be described by an *ab initio* potential energy surface constructed as a quadratic expansion in powers of the normal mode displacements $(\xi_j - \xi_j^0)$:

$$V(\xi_1, \xi_2, ..., \xi_{3N-6}) = V(\xi_1^0, \xi_2^0, ..., \xi_{3N-6}^0) + \frac{1}{2} \sum_{j=1}^{3N-6} \mu_j \omega_j^2 \left(\xi_j - \xi_j^0 \right)^2, \qquad (562)$$

with μ_j and ω_j the reduced masses and frequencies of the normal modes, obtained from accurate *ab initio* quantum chemistry calculations. Similar quadratic expansions in terms of internal coordinates (*i.e.*, bond-lengths, bond-angles, etc.) can also properly describe multidimensional systems near their equilibrium configuration as follows:

$$V(\mathbf{r}, \theta, \phi, \mathbf{d}) = V_{str.}(\mathbf{r}) + V_{bend}(\theta) + V_{torsion}(\phi) + V_{non-bond}(\mathbf{d}), \qquad (563)$$

in terms of quadratic expansions in powers of bond-lengths $\mathbf{r}$, bending angles θ, torsion angles ϕ, and distances $\mathbf{d}$ for non-bonding interactions. The model potential introduced by Eq. (827) is inspired in molecular mechanics models where atoms and bonds are described as spheres and spings. The expression of the energy is these systems is called molecular mechanics force field. The parametrization of the individual terms in the quadratic expansions can be based on *ab initio* calculations, or empirically based on the properties of the system as compared to experimental data.

Quadratic expansions are useful for describing the dynamics of the system near equilibrium configurations, however, they are limited on their capabilities to describe chemical reactivity as determined by bond-breaking and bond-forming processes (*e.g.*, hydrogen transfer). An example is the proton transfer between species between an acid AH and a base B^- during the titration:

$$B^- + HA \rightarrow BH + AY^-. \qquad (564)$$

The potential energy profile for this proton transfer reaction is often modulated by displacements along normal modes that affect the distance $q_1 = d$ and the relative orientation $q_2 = \theta$ between the proton donor and acceptor, while the remaining degrees of freedom in the system remain fluctuating as small amplitude harmonic oscillators. For studying this kind of reactive processes in polyatomic systems, we need to generalize the quadratic expansions introduced above by modeling a small

set of large amplitude coordinates (*e.g.*, q_1 and q_2) as coupled to a quadratic expansion for the remaining degrees of freedom in the system. As an example, the complete set of $3N - 6$ normal mode coordinates can be partitioned into two sets of coordinates, including 2 "large amplitude" *reaction coordinates* ($r_1 = q_1$ and $r_2 = q_2$) and the remaining $(3N - 8)$ degrees of freedom $\{q_j\} with j = 3, ..., 3N - 6$.

To construct the complete potential energy surface, we first compute the 2-dimensional *reaction surface* $V_0(r_1, r_2)$, defined as the minimum energy of the molecular system with respect to relaxation of the remaining $3N - 8$ coordinates $\mathbf{q}$, subject to the constraints of fixed values for r_1 and r_2:

$$\frac{\partial V(r_1, r_2, ..., q_i, ...)}{\partial q_i} = 0, \tag{565}$$

with $i = 3, ..., (3N - 6)$. The values of the coordinates $q_3, ..., q_{3N-6}$ as functions of r_1 and r_2, determined by Eq. (565), are the equilibrium positions q_j^0 when the system is *on* the reaction surface $V_0(r_1, r_2) = V(r_1, r_2, ..., q_i^0, ...)$. The complete potential energy $V(r_1, r_2, ..., q_i, ...)$ is then expanded, for the description of configurations where the coordinates $q_3, ..., q_{3N-6}$ are not too much displaced from their equilibrium positions, by expanding the $q_3, ..., q_{3N-6}$ dependence to second order around their equilibrium positions:

$$V(r_1, r_2, q_3, ..., q_{3N-6}) = V_0(r_1, r_2) + \sum_{j,k=3}^{3N-6} \frac{1}{2} \left(q_j - q_j^0\right) \left(\frac{\partial^2 V}{\partial q_j \partial q_k}\right)_{q_{j/k}=q_{j/k}^0} \left(q_k - q_k^0\right). \tag{566}$$

The potential has no linear term in $(q_k - q_k^0)$ because the first order derivatives are equal to zero (see Eq. (565)) by definition of $V_0(\mathbf{r}) = V_0(r_1, r_2)$.

The complete reaction surface Hamiltonian that can be used for mutidimensional quantum dynamics simulations is

$$\begin{aligned} H(\mathbf{r}, \mathbf{P}_r, \mathbf{q}, \mathbf{P}_q) = \sum_{j=1}^{2} \frac{\mathbf{P}_r^2}{2\mu_j} + \sum_{j=3}^{3N-6} \frac{\mathbf{P}_q^2}{2\mu_j} \\ + V_0(\mathbf{r}) + \frac{1}{2} \sum_{j=3}^{3N-6} \sum_{k=3}^{3N-6} (q_j - q_j^0(\mathbf{r})) \frac{\partial^2 V(\mathbf{r}, \mathbf{q})}{\partial q_j \partial q_k} (q_k - q_k^0(\mathbf{r})). \end{aligned} \tag{567}$$

34   Wigner Transform Formulation of Quantum Dynamics

The goal of this section is to describe the Wigner-transform function, $\rho_t^W(p, q)$, introduced by Wigner ([*Phys. Rev.* (1932) **40**:749-759]), and its time-propagation in terms of the evolution of trajectories $\{q(t), p(t)\}$ in phase-space, according to a Lagrangian formulation introduced by Wong (see [*Phys. Rev. C* (1982) **25**:1460-1475] and [*J. Opt. B: Quantum Semiclass. Opt.* (2003) **5**:S420-S428]).

34.1 Definition and Properties:

Given a wavefunction $\Psi_t(x)$, the Wigner transform $\rho_t^W(p, q)$ is defined as follows:

$$\rho^W(p, q; t) \equiv \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} ds \; e^{\frac{i}{\hbar}ps} \Psi^*(q + s/2; t)\Psi(q - s/2; t). \tag{568}$$

This quantity is similar to the phase-space probability density, since it gives the probability density $|\Psi(q; t)|^2$ when integrated with respect to p:

$$
\begin{aligned}
\int_{-\infty}^{\infty} dp \, \rho^W(p, q; t) &= \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} dp \int_{-\infty}^{\infty} ds \; e^{ips/\hbar} \Psi^*(q + s/2; t)\Psi(q - s/2; t), \\
&= \int_{-\infty}^{\infty} ds \, \delta(s)\Psi^*(q + s/2; t)\Psi(q - s/2; t), \\
&= \Psi^*(q; t)\Psi(q; t), \\
&= |\Psi(q; t)|^2.
\end{aligned} \tag{569}
$$

In addition, $\rho^W(p, q; t)$ gives the Fourier transform probability density $|\widetilde{\Psi}(p; t)|^2$ when integrated with respect to q (where $\widetilde{\Psi}$ is the Fourier transform of Ψ):

$$
\begin{aligned}
\int_{-\infty}^{\infty} dq \rho^W(p, q; t) &= \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} dq \int_{-\infty}^{\infty} ds \; e^{ips/\hbar} \Psi^*(q + s/2; t)\Psi(q - s/2; t), \\
&= \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} dx' \int_{-\infty}^{\infty} dx e^{ip(x-x')/\hbar} \Psi^*(x; t)\Psi(x'; t), \\
&= \left| \frac{1}{\sqrt{2\pi\hbar}} \int_{-\infty}^{\infty} dx e^{-ipx/\hbar} \Psi(x; t) \right|^2, \\
&= |\widetilde{\Psi}(p; t)|^2.
\end{aligned} \tag{570}
$$

where in the second line of Eq. (570) we introduced the variable transformation $x \equiv q + s/2$ and $x' \equiv q - s/2$, with Jacobian $\left| det[\frac{\partial(q,s)}{\partial(x,x')}] \right| = 1$. In addition, expectation values of any function of coordinates and momenta (*e.g.*, $H(q, p) = p^2/(2m) + V(q)$) can be computed according to the normal probability calculation:

$$E_t = \frac{\int dq \int dp \rho^W(p, q; t) H(q, p)}{\int dq \int dp \rho^W(p, q; t)}. \tag{571}$$

Therefore, $\rho^W(q, p; t)$ has properties of a normal probability function. However, it can take negative values! Therefore, it *cannot* be interpreted as the simultaneous probability for coordinates and momenta (*i.e.*, as the probability density). Nevertheless, it is a useful function that can be used to compute probabilities and expectation values.

34.2 Time Evolution:

The equation of motion of $\rho_t^W(p,q)$ can be obtained by computing the time-derivative of both sides of Eq. (590), and substituting the time-derivative to the wavefunctions by using the time-dependent Schrödinger equation

$$\frac{\partial \Psi(q \pm s/2; t)}{\partial t} = -\frac{1}{i\hbar}\frac{\hbar^2}{2m}\frac{\partial^2 \Psi(q \pm s/2; t)}{\partial q^2} + \frac{1}{i\hbar}V(q \pm s/2)\Psi(q \pm s/2; t),$$

$$= -\frac{1}{i\hbar}\frac{\hbar^2}{2m}4\frac{\partial^2 \Psi(q \pm s/2; t)}{\partial s^2} + \frac{1}{i\hbar}V(q \pm s/2)\Psi(q \pm s/2; t),$$

(572)

where in the second line of Eq. (573) we have introduced the substitution $\frac{\partial \Psi^*(q \pm s/2; t)}{\partial q} = \pm 2 \frac{\partial \Psi^*(q \pm s/2; t)}{\partial s}$. Thus, the time-derivative of the Wigner transform is

$$\frac{\partial \rho^W(p,q;t)}{\partial t} = \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\left[\frac{\partial \Psi^*(q+s/2; t)}{\partial t}\Psi(q-s/2; t) + \Psi^*(q+s/2; t)\frac{\partial \Psi(q-s/2; t)}{\partial t}\right],$$

$$= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\{\frac{i}{\hbar}\left[V(q+s/2) - V(q-s/2)\right]\Psi^*(q+s/2; t)\Psi(q-s/2; t)$$

$$-\frac{i}{\hbar}\frac{\hbar^2}{2m}4\left[\Psi(q-s/2; t)\frac{\partial^2}{\partial s^2}\Psi^*(q+s/2; t) - \Psi^*(q+s/2; t)\frac{\partial^2}{\partial s^2}\Psi(q-s/2; t)\right]\},$$

$$= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\{\frac{i}{\hbar}\left[V(q+s/2) - V(q-s/2)\right]\Psi^*(q+s/2; t)\Psi(q-s/2; t)$$

$$-\frac{i}{\hbar}\frac{\hbar^2}{2m}4\frac{\partial}{\partial s}\left[\Psi(q-s/2; t)\frac{\partial}{\partial s}\Psi^*(q+s/2; t) - \Psi^*(q+s/2; t)\frac{\partial}{\partial s}\Psi(q-s/2; t)\right]\},$$

$$= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\{\frac{i}{\hbar}\left[V(q+s/2) - V(q-s/2)\right]\Psi^*(q+s/2; t)\Psi(q-s/2; t)$$

$$-\frac{i}{\hbar}\frac{\hbar^2}{2m}2\frac{\partial}{\partial s}\left[\Psi(q-s/2; t)\frac{\partial}{\partial q}\Psi^*(q+s/2; t) + \Psi^*(q+s/2; t)\frac{\partial}{\partial q}\Psi(q-s/2; t)\right]\}.$$

(573)

Therefore,

$$\frac{\partial \rho^W(p,q;t)}{\partial t} = \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\{\frac{i}{\hbar}\left[V(q+s/2) - V(q-s/2)\right]\Psi^*(q+s/2; t)\Psi(q-s/2; t)$$

$$+ \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\frac{\hbar}{mi}\frac{\partial}{\partial s}\frac{\partial}{\partial q}\left[\Psi(q-s/2; t)\Psi^*(q+s/2; t)\right]\}.$$

(574)

Integrating by parts the second line of Eq. (574), using $\int_a^b u\,dv = uv|_a^b - \int_a^b v\,du$ with $u \equiv e^{\frac{i}{\hbar}ps}$ and $v \equiv \frac{\hbar}{mi}\frac{\partial}{\partial q}\left[\Psi(q-s/2;t)\Psi^*(q+s/2;t)\right]$, we obtain:

$$\frac{\partial \rho^W(p,q;t)}{\partial t} = \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\frac{i}{\hbar}\left[V(q+s/2)-V(q-s/2)\right]\Psi^*(q+s/2;t)\Psi(q-s/2;t)$$

$$-\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\frac{p}{m}\frac{\partial}{\partial q}\left[\Psi(q-s/2;t)\Psi^*(q+s/2;t)\right],$$

$$= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\frac{i}{\hbar}\left[V(q+s/2)-V(q-s/2)\right]\Psi^*(q+s/2;t)\Psi(q-s/2;t)$$

$$-\frac{p}{m}\frac{\partial\rho^W(p,q;t)}{\partial q}.$$

$$(575)$$

34.3 Wigner Transform Propagation:

The equation of motion,

$$\frac{\partial \rho^W(p,q;t)}{\partial t} = \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\frac{i}{\hbar}\left[V(q+s/2)-V(q-s/2)\right]\Psi^*(q+s/2;t)\Psi(q-s/2;t)$$

$$-\frac{p}{m}\frac{\partial\rho^W(p,q;t)}{\partial q},$$

$$(576)$$

can be re-written by expanding the potential energy difference in powers of s, as follows:

$$[V(q+s/2)-V(q-s/2)] = \sum_{j=0}^{\infty} 2\left(\frac{s}{2}\right)^{2j+1}\frac{\partial^{2j+1}V(q)}{\partial q^{2j+1}}\frac{1}{(2j+1)!}, \qquad (577)$$

giving

$$\frac{\partial \rho^W(p,q;t)}{\partial t} = \frac{1}{2^{2j}}\frac{1}{2\pi\hbar}\sum_{j=0}^{\infty}\frac{1}{(2j+1)!}\frac{i}{\hbar}\frac{\partial^{2j+1}V(q)}{\partial q^{2j+1}}\int_{-\infty}^{\infty} ds\, s^{2j+1}e^{\frac{i}{\hbar}ps}\Psi^*(q+s/2;t)\Psi(q-s/2;t)$$

$$-\frac{p}{m}\frac{\partial\rho^W(p,q;t)}{\partial q},$$

$$= \frac{1}{2^{2j}}\frac{1}{2\pi\hbar}\sum_{j=0}^{\infty}\frac{1}{(2j+1)!}\frac{i}{\hbar}\frac{\partial^{2j+1}V(q)}{\partial q^{2j+1}}\frac{\partial^{2j+1}}{\partial p^{2j+1}}\left(\frac{\hbar}{i}\right)^{2j+1}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\Psi^*(q+s/2;t)\Psi(q-s/2;t)$$

$$-\frac{p}{m}\frac{\partial\rho^W(p,q;t)}{\partial q},$$

$$= \sum_{j=0}^{\infty}\frac{1}{(2j+1)!}\left(\frac{\hbar}{2i}\right)^{2j}\frac{\partial^{2j+1}V(q)}{\partial q^{2j+1}}\frac{\partial^{2j+1}\rho^W}{\partial p^{2j+1}} - \frac{p}{m}\frac{\partial\rho^W}{\partial q},$$

$$(578)$$

135

Introducing the operators $\hat{P}_q = -i\hbar\partial/\partial q$ and $\hat{P}_p = -i\hbar\partial/\partial p$, we obtain:

$$\frac{\partial}{\partial t}\rho^W = \sum_{j=0}^{\infty} \frac{1}{(2j+1)!} \left(\frac{\hbar}{2i}\right)^{2j} \frac{\partial^{2j+1}V(q)}{\partial q^{2j+1}} \frac{\partial^{2j+1}\rho^W}{\partial p^{2j+1}} - \frac{p}{m}\frac{\partial\rho^W}{\partial q},$$

$$= -\sum_{j=0}^{\infty} \frac{1}{(2j+1)!} \left(\frac{\hbar}{2i}\right)^{2j} \frac{\partial^{2j+1}V(q)}{\partial q^{2j+1}} \frac{1}{(i\hbar)(-i\hbar)^{2j}} \hat{P}_p^{2j+1}\rho^W + \frac{p}{i\hbar m}\hat{P}_q\rho^W, \qquad (579)$$

$$i\hbar\frac{\partial}{\partial t}\rho^W = \left[-\sum_{j=0}^{\infty} \frac{1}{(2j+1)!} \left(\frac{1}{2}\right)^{2j} \frac{\partial^{2j+1}V(q)}{\partial q^{2j+1}} \hat{P}_p^{2j+1} + \frac{p}{m}\hat{P}_q\right]\rho^W = \hbar\hat{\mathbf{L}}\rho^W.$$

According to Eq. (579), ρ^W can be propagated as follows:

$$\rho^W(t) = e^{-\frac{i}{\hbar}\hbar\hat{\mathbf{L}}t}\rho^W(0), \qquad (580)$$

where $\hbar\mathbf{L}$ can be written in a DVR, as follows:

$$\hbar L_{l,l'} = -\sum_{j=0}^{\infty} \frac{1}{(2j+1)!} \left(\frac{1}{2}\right)^{2j} \frac{\partial^{2j+1}V(q_l)}{\partial q_l^{2j+1}} \delta_{q_l q_{l'}} \frac{\Delta_p \Delta_{Pp}}{2\pi\hbar} \sum_{k=1}^{n_{Pp}} e^{\frac{i}{\hbar}(p_{l'}-p_l)Pp_k} Pp_k^{2j+1}$$

$$+ \frac{p_l}{m}\delta_{p_l p_{l'}} \frac{\Delta_q \Delta_{Pq}}{2\pi\hbar} \sum_{k=1}^{n_{Pq}} e^{\frac{i}{\hbar}(q_{l'}-q_l)Pq_k} Pq_k \qquad (581)$$

or

$$i\hbar L_{l,l'} = -\sum_{j=0}^{\infty} \frac{1}{(2j+1)!} \left(\frac{1}{2}\right)^{2j} \frac{\partial^{2j+1}V(q_l)}{\partial q_l^{2j+1}} \delta_{q_l q_{l'}} \frac{\Delta_p \Delta_{Pp}}{2\pi\hbar} \sum_{k=1}^{n_{Pp}/2} 2\sin[(p_{l'}-p_l)Pp_k/\hbar]Pp_k^{2j+1}$$

$$+ \frac{p_l}{m}\delta_{p_l p_{l'}} \frac{\Delta_q \Delta_{Pq}}{2\pi\hbar} \sum_{k=1}^{n_{Pq}/2} 2\sin[(q_{l'}-q_l)Pq_k/\hbar]Pq_k$$

$$(582)$$

in the representation of equally spaced delta functions $\delta(q-q_j)$, at $q_j = (j-n_q/2)\Delta_q$, with $\Delta_q = (q_{max}-q_{min})/n_q$ and $j = 1-n_q$, where $Pq_k = \Delta_{Pq}(k-n_{Pq}/2)$ with $\Delta_{Pq} = 2\pi/(q_{max}-q_{min})$. Analogously, $\delta(p-p_j)$ are placed at $p_j = (j-n_p/2)\Delta_p$, with $\Delta_p = (p_{max}-p_{min})/n_p$ and $j = 1-n_p$, where $Pp_k = \Delta_{Pp}(k-n_{Pp}/2)$ with $\Delta_{Pp} = 2\pi/(p_{max}-p_{min})$.

We note that, according to Eq. (579), the full quantum mechanical evolution of $\rho^W(t)$ is based on a generalized Liouvillian

$$\hat{\mathbf{L}} = \hat{\mathbf{L}}_c + \hat{\mathbf{A}} \qquad (583)$$

where

$$\hat{\mathbf{A}} = -\frac{1}{\hbar}\sum_{j=1}^{\infty} \frac{1}{(2j+1)!} \left(\frac{1}{2}\right)^{2j} \frac{\partial^{2j+1}V(q)}{\partial q^{2j+1}} \hat{P}_p^{2j+1}, \qquad (584)$$

136

and $\hat{\mathbf{L}}_c$ is the classical Liouvillian

$$i\hat{\mathbf{L}}_c = \frac{p}{m}\frac{\partial}{\partial q} - \frac{\partial V(q)}{\partial q}\frac{\partial}{\partial p}, \tag{585}$$

describing the evolution of ρ^W according to Hamilton's equations. All quantum effects, beyond those already included in the initial conditions defined by $\rho^W(0)$, are introduced by $\hat{\mathbf{A}}$.

Note that $\hat{\mathbf{A}} = 0$ when $V(q)$ is quadratic since it involves third and higher derivatives of V with respect to q. Therefore, it is clear that the classical evolution of the Wigner transform provides a full quantum mechanical description of the system evolving on a quadratic potential.

34.4   Computational Problem: WTP

1. Write a program to write the DVR Liouvillian, introduced by Eq. (582), for the Morse potential $V(\hat{x}) = De(1 - \exp(-a(\hat{x} - x_e)))^2$, with $x_e = 0$, $De = 8$, and $a = \sqrt{k/(2D_e)}$, where $k = m\omega^2$.
2. Propagate the Wigner transform of a state initialized as $|0\rangle$, according to Eq. (978), using the DVR Liouvillian of item 1 and the Lanczos propagation scheme in the Krylov basis as implemented in *Expokit*. Compare the time-evolved Wigner transform to the corresponding propagation based on the classical Liouvillian $\mathbf{L}_c$.
Solution: Sec. 61.25.

34.5   N-Level System:

The goal of this subsection is to introduce the Wigner transform propagation scheme as applied to the description of dynamics of a system described by the following N-level Hamiltonian

$$\hat{H} = \sum_{j=1}^{N}\sum_{k=1}^{N} H_{kj}|k\rangle\langle j|. \tag{586}$$

We consider the time-dependent state

$$|\Psi\rangle = \sum_{l} \phi_l(q)|l\rangle, \tag{587}$$

evolving according to the time-dependent Schrödinger equation,

$$i\hbar\frac{\partial}{\partial t}|\Psi\rangle = \hat{H}|\Psi\rangle,$$

$$= \sum_{j=1}^{N}\sum_{l}\sum_{k=1}^{N} H_{kj}|k\rangle\langle j|l\rangle\phi_l(q), \tag{588}$$

$$i\hbar\sum_{l}\frac{\partial}{\partial t}\phi_l(q)|l\rangle = \sum_{j=1}^{N}\sum_{k=1}^{N} H_{kj}|k\rangle\phi_j(q),$$

or

$$i\hbar\dot{\phi}_l(q) = \sum_{j=1}^{N} H_{lj}\phi_j(q), \tag{589}$$

Substituting the time-dependent wavefunction introduced by Eq. (587) into the expression of the Wigner transform,

$$\rho^W(p,q;t) \equiv \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\Psi^*(q+s/2;t)\Psi(q-s/2;t). \tag{590}$$

we obtain

$$\rho^W(p,q;t) = \sum_{k=1}^{N}\sum_{k'=1}^{N} |k'\rangle\langle k|\frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\phi_k^*(q+s/2;t)\phi_{k'}(q-s/2;t),$$
$$= \sum_{k=1}^{N}\sum_{k'=1}^{N} |k'\rangle\langle k|\rho_{k',k}^W(p,q;t), \tag{591}$$

where

$$\rho_{k',k}^W(p,q;t) = \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\phi_k^*(q+s/2;t)\phi_{k'}(q-s/2;t). \tag{592}$$

Therefore,

$$i\hbar\frac{\partial}{\partial t}\rho_{k',k}^W = \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps} \left[-\left[i\hbar\dot{\phi}_k(q+s/2;t)\right]^* \phi_{k'}(q-s/2;t) \right.$$
$$\left. +\phi_k^*(q+s/2;t)i\hbar\dot{\phi}_{k'}(q-s/2;t) \right],$$
$$= \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps} \left[-\left[\sum_{j=1}^{N} H_{kj}\phi_j(q+s/2;t)\right]^* \phi_{k'}(q-s/2;t) \right.$$
$$\left. +\phi_k^*(q+s/2;t)\sum_{j=1}^{N} H_{k'j}\phi_j(q-s/2;t) \right], \tag{593}$$
$$= \sum_{j=1}^{N} \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps} \left[\phi_k^*(q+s/2;t)H_{k'j}\phi_j(q-s/2;t) \right.$$
$$\left. -\phi_{k'}(q-s/2;t)\left[H_{kj}\phi_j(q+s/2;t)\right]^*\right],$$
$$i\hbar\frac{\partial}{\partial t}\rho_{k',k}^W = \sum_{j=1}^{N} H_{k'j}\rho_{jk}^W - \rho_{k'j}^W H_{jk} = \left[H,\rho^W\right]_{kk'}.$$

As an example, we consider the Hamiltonian,

$$\hat{H} = \sum_j \sum_k |j\rangle\langle k| \left[-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial q^2}\delta_{jk} + V_{jk}(q) \right], \tag{594}$$

138

with matrix elements

$$H_{kj} = \left[-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial q^2}\delta_{kj} + V_{kj}(q) \right]. \tag{595}$$

Substituting Eq. (595) into Eq. (593), when $V_{jk} = V_{kj}$, we obtain:

$$i\hbar\frac{\partial}{\partial t}\rho_{k',k}^W = \sum_{j=1}^{N}\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\left[\phi_k^*(q+s/2;t)H_{k'j}\phi_j(q-s/2;t)\right. \tag{596}$$
$$\left. -\phi_{k'}(q-s/2;t)\left[H_{kj}\phi_j(q+s/2;t)\right]^*\right].$$

Therefore,

$$\frac{\partial\rho_{k'k}^W(p,q;t)}{\partial t} = \delta_{kk'}\left[\sum_{l=0}^{\infty}\frac{1}{(2l+1)!}\left(\frac{\hbar}{2i}\right)^{2l}\left(\sum_{j=1}^{N}\frac{\partial^{2l+1}V_{k'j}(q)}{\partial q^{2l+1}}\right)\frac{\partial^{2l+1}\rho_{jk}^W}{\partial p^{2l+1}} - \frac{p}{m}\frac{\partial\rho_{k'k}^W}{\partial q}\right]$$
$$-\frac{i}{\hbar}(1-\delta_{kk'})\sum_{j=1}^{N}\left[\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, V_{k'j}(q-s/2)e^{\frac{i}{\hbar}ps}\phi_k^*(q+s/2;t)\phi_j(q-s/2;t)\right.$$
$$\left.-\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, V_{jk}(q+s/2)e^{\frac{i}{\hbar}ps}\phi_{k'}(q-s/2;t)\phi_j^*(q+s/2;t)\right], \tag{597}$$

Therefore, the diagonal elements (i.e., $k = k'$) evolve, as follows:

$$\frac{\partial\rho_{kk}^W(p,q;t)}{\partial t} = \sum_{l=0}^{\infty}\frac{1}{(2l+1)!}\left(\frac{\hbar}{2i}\right)^{2l}\sum_{j=1}^{N}\frac{\partial^{2l+1}V_{kj}}{\partial q^{2l+1}}\frac{\partial^{2l+1}\rho_{jk}^W}{\partial p^{2l+1}} - \frac{p}{m}\frac{\partial\rho_{kk}^W}{\partial q}, \tag{598}$$

while the off-diagonal elements ($k \neq k'$) evolve, as follows:

$$\frac{\partial\rho_{k'k}^W(p,q;t)}{\partial t} = -\frac{1}{\hbar}\sum_{l=0}^{\infty}\frac{i^{(l+1)}}{l!}\left(\frac{\hbar}{2}\right)^l\sum_{j=1}^{N}\left[\frac{\partial^l V_{k'j}}{\partial q^l}\frac{\partial^l\rho_{jk}^W}{\partial p^l} - (-1)^l\frac{\partial^l\rho_{k'j}^W}{\partial p^l}\frac{\partial^l V_{jk}}{\partial q^l}\right]. \tag{599}$$

According to Eq. (592), $\rho_{kk'}^W = \rho_{k'k}^{W\,*} = \Re\rho_{k'k}^W + i\Im\rho_{k'k}^W$. Therefore, we can propagate both elements $\rho_{kk'}^W$ and $\rho_{k'k}^W$ by evolving the real-valued real and imaginary parts $\Re\rho_{k'k}^W = (\rho_{k'k}^W + (\rho_{k'k}^W)^*)/2 = (\rho_{k'k}^W + \rho_{kk'}^W)/2$ and $\Im\rho_{k'k}^W = (\rho_{k'k}^W - (\rho_{k'k}^W)^*)/2i = (\rho_{k'k}^W + \rho_{kk'}^W)/2i$, respectively,

which evolve according to the following equations of motion:

$$\frac{\partial \Re \rho_{k'k}^W}{\partial t} = \frac{1}{\hbar} \sum_{l=0}^{\infty} \frac{i^{2l}}{(2l)!} \left(\frac{\hbar}{2}\right)^{2l} \sum_{j=1}^{N} \left[\frac{\partial^{2l} V_{k'j}}{\partial q^{2l}} \frac{\partial^{2l} \Im \rho_{jk}^W}{\partial p^{2l}} - \frac{\partial^{2l} \Im \rho_{k'j}^W}{\partial p^{2l}} \frac{\partial^{2l} V_{jk}}{\partial q^{2l}} \right]$$

$$+ \frac{1}{\hbar} \sum_{l=0}^{\infty} \frac{i^{2l}}{(2l+1)!} \left(\frac{\hbar}{2}\right)^{2l+1} \sum_{j=1}^{N} \left[\frac{\partial^{2l+1} V_{k'j}}{\partial q^{2l+1}} \frac{\partial^{2l+1} \Re \rho_{jk}^W}{\partial p^{2l+1}} + \frac{\partial^{2l+1} \Re \rho_{k'j}^W}{\partial p^{2l+1}} \frac{\partial^{2l+1} V_{jk}}{\partial q^{2l+1}} \right],$$

$$\frac{\partial \Im \rho_{k'k}^W}{\partial t} = \frac{1}{\hbar} \sum_{l=0}^{\infty} \frac{i^{2l}}{(2l)!} \left(\frac{\hbar}{2}\right)^{2l} \sum_{j=1}^{N} \left[\frac{\partial^{2l} V_{k'j}}{\partial q^{2l}} \frac{\partial^{2l} \Re \rho_{jk}^W}{\partial p^{2l}} - \frac{\partial^{2l} \Re \rho_{k'j}^W}{\partial p^{2l}} \frac{\partial^{2l} V_{jk}}{\partial q^{2l}} \right] \quad (600)$$

$$+ \frac{1}{\hbar} \sum_{l=0}^{\infty} \frac{i^{2l}}{(2l+1)!} \left(\frac{\hbar}{2}\right)^{2l+1} \sum_{j=1}^{N} \left[\frac{\partial^{2l+1} V_{k'j}}{\partial q^{2l+1}} \frac{\partial^{2l+1} \Im \rho_{jk}^W}{\partial p^{2l+1}} + \frac{\partial^{2l+1} \Im \rho_{k'j}^W}{\partial p^{2l+1}} \frac{\partial^{2l+1} V_{jk}}{\partial q^{2l+1}} \right].$$

In the particular case of harmonic potentials V_{jk}, the equations of motion are:

$$\frac{\partial \Re \rho_{k'k}^W}{\partial t} = \frac{1}{\hbar} \sum_{j=1}^{N} \left[V_{k'j} \Im \rho_{jk}^W - \Im \rho_{k'j}^W V_{jk} \right] - \frac{\hbar}{8} \sum_{j=1}^{N} \left[\frac{\partial^2 V_{k'j}}{\partial q^2} \frac{\partial^2 \Im \rho_{jk}^W}{\partial p^2} - \frac{\partial^2 \Im \rho_{k'j}^W}{\partial p^2} \frac{\partial^2 V_{jk}}{\partial q^2} \right]$$

$$+ \frac{1}{2} \sum_{j=1}^{N} \left[\frac{\partial V_{k'j}}{\partial q} \frac{\partial \Re \rho_{jk}^W}{\partial p} + \frac{\partial \Re \rho_{k'j}^W}{\partial p} \frac{\partial V_{jk}}{\partial q} \right],$$

$$\frac{\partial \Im \rho_{k'k}^W}{\partial t} = \frac{1}{\hbar} \sum_{j=1}^{N} \left[V_{k'j} \Re \rho_{jk}^W - \Re \rho_{k'j}^W V_{jk} \right] - \frac{\hbar}{8} \sum_{j=1}^{N} \left[\frac{\partial^2 V_{k'j}}{\partial q^2} \frac{\partial^2 \Re \rho_{jk}^W}{\partial p^2} - \frac{\partial^2 \Re \rho_{k'j}^W}{\partial p^2} \frac{\partial^2 V_{jk}}{\partial q^2} \right] \quad (601)$$

$$+ \frac{1}{2} \sum_{j=1}^{N} \left[\frac{\partial V_{k'j}}{\partial q} \frac{\partial \Im \rho_{jk}^W}{\partial p} + \frac{\partial \Im \rho_{k'j}^W}{\partial p} \frac{\partial V_{jk}}{\partial q} \right],$$

$$\frac{\partial \Re \rho_{kk}^W}{\partial t} = \sum_{k'=1}^{N} \frac{\partial V_{kk'}}{\partial q} \frac{\partial \Re \rho_{k'k}^W}{\partial p} - \frac{p}{m} \frac{\partial \Re \rho_{kk}^W}{\partial q},$$

$$\frac{\partial \Im \rho_{kk}^W}{\partial t} = \sum_{k'=1}^{N} \frac{\partial V_{kk'}}{\partial q} \frac{\partial \Im \rho_{k'k}^W}{\partial p} - \frac{p}{m} \frac{\partial \Im \rho_{kk}^W}{\partial q},$$

or in a more compact form

$$\frac{\partial \rho_{kk'}^W}{\partial t} = \sum_{j=1}^{N} \frac{1}{2} \left[\frac{\partial V_{k'j}}{\partial q} \frac{\partial \rho_{jk}^{W*}}{\partial p} + \frac{\partial \rho_{k'j}^{W*}}{\partial p} \frac{\partial V_{jk}}{\partial q} \right]$$

$$- \frac{i}{\hbar} \left[\left(V_{k'j} \rho_{jk}^W - \rho_{k'j}^W V_{jk} \right) + \frac{\hbar^2}{8} \left(\frac{\partial^2 V_{k'j}}{\partial q^2} \frac{\partial^2 \rho_{jk}^W}{\partial p^2} - \frac{\partial^2 \rho_{k'j}^W}{\partial p^2} \frac{\partial^2 V_{jk}}{\partial q^2} \right) \right], \quad (602)$$

$$\frac{\partial \rho_{kk}^W}{\partial t} = \sum_{k'=1}^{N} \frac{\partial V_{kk'}}{\partial q} \frac{\partial \rho_{k'k}^W}{\partial p} - \frac{p}{m} \frac{\partial \rho_{kk}^W}{\partial q}.$$

140

or

$$\frac{\partial \rho_{kk'}^W}{\partial t} = \frac{1}{2}\frac{\partial}{\partial q}(V_{k'k'} + V_{kk})\frac{\partial \rho_{kk'}^W}{\partial p}$$

$$+ \frac{1}{2}\frac{\partial V_{k'k}}{\partial q}\left(\frac{\partial \rho_{kk}^W}{\partial p} + \frac{\partial \rho_{k'k'}^W}{\partial p}\right)$$

$$+ \sum_{j\neq k',k}^{N}\frac{1}{2}\frac{\partial V_{k'j}}{\partial q}\frac{\partial \rho_{jk}^{W\,*}}{\partial p} + \sum_{j\neq k,k'}^{N}\frac{1}{2}\frac{\partial \rho_{k'j}^{W\,*}}{\partial p}\frac{\partial V_{jk}}{\partial q}$$

$$- \frac{i}{\hbar}\left[(V_{k'k'} - V_{kk})\rho_{k'k}^W + \frac{\hbar^2}{8}\frac{\partial^2}{\partial q^2}(V_{k'k'} - V_{kk})\frac{\partial^2 \rho_{k'k}^W}{\partial p^2}\right] \tag{603}$$

$$- \frac{i}{\hbar}\left(\sum_{j\neq k'}^{N}V_{k'j}\rho_{jk}^W - \sum_{j\neq k}^{N}\rho_{k'j}^W V_{jk}\right)$$

$$- \frac{i\hbar}{8}\left(\sum_{j\neq k'}^{N}\frac{\partial^2 V_{k'j}}{\partial q^2}\frac{\partial^2 \rho_{jk}^W}{\partial p^2} - \sum_{j\neq k}^{N}\frac{\partial^2 \rho_{k'j}^W}{\partial p^2}\frac{\partial^2 V_{jk}}{\partial q^2}\right),$$

$$\frac{\partial \rho_{kk}^W}{\partial t} = \frac{\partial V_{kk}}{\partial q}\frac{\partial \rho_{kk}^W}{\partial p} - \frac{p}{m}\frac{\partial \rho_{kk}^W}{\partial q} + \sum_{k'\neq k}^{N}\frac{\partial V_{kk'}}{\partial q}\frac{\partial \rho_{k'k}^W}{\partial p}.$$

34.6 Linearized Wigner Transform:

To find a solution of Eq. (575) in terms of coordinates and momenta {q(t),p(t)} that obey the usual Hamilton's equations, we express $\Psi(q;t)$ in real valued polar coordinates:

$$\Psi(q;t) = \rho(q;t)^{1/2} e^{\frac{i}{\hbar}S(q;t)}, \tag{604}$$

and we introduce the following linearization:

$$\Psi(q \pm s/2; t) \approx \left[\rho(q;t)^{1/2} \pm \frac{s}{2}\frac{\partial}{\partial q}\rho(q;t)^{1/2}\right] e^{\frac{i}{\hbar}\left[S(q;t) \pm \frac{s}{2}mv(q;t)\right]}, \tag{605}$$

where $mv(q;t) = \frac{\partial}{\partial q}S(q;t)$, with

$$\Psi^*(q + s/2; t)\Psi(q - s/2; t) \approx \rho(q;t)e^{-\frac{i}{\hbar}mv(q;t)s}, \tag{606}$$

and

$$V(q \pm s/2) \approx V(q) \pm \frac{s}{2}V'(q). \tag{607}$$

Substituting Eq. (605) into Eq. (590), we obtain the linearized Wigner transform:

$$f(p, q; t) = \rho(q;t)\delta(p - mv(q;t)). \tag{608}$$

In contrast to the Wigner transform $\rho^W(q, p; t)$, the linearized Wigner transform $f(q, p; t)$ is always positive and evolves in time according to the classical Boltzmann equation of motion,

$$\frac{\partial f(p, q; t)}{\partial t} = -\left[\frac{\partial f(p, q; t)}{\partial q}\frac{p}{m} - \frac{\partial f(p, q; t)}{\partial p}V'(q)\right], \tag{609}$$

obtained by substituting Eq. (605) and Eq. (607) into Eq. (575).

Note that Eq. (609) leads to the classical Liouville equation,

$$\frac{\partial f(p, q; t)}{\partial t} = -\left[\frac{\partial f(p, q; t)}{\partial q}\frac{\partial H(q, p)}{\partial p} + \frac{\partial f(p, q; t)}{\partial p}\frac{\partial H(q, p)}{\partial q}\right], \tag{610}$$

for the Hamiltonian

$$H(q, p) = \frac{p^2}{2m} + V(q), \tag{611}$$

when the variables p and q evolve in time according to Hamilton's equations:

$$\frac{\partial p}{\partial t} = -\frac{\partial H(q, p)}{\partial q} = -\frac{\partial V(q)}{\partial q},$$
$$\frac{\partial q}{\partial t} = \frac{\partial H(q, p)}{\partial p}. \tag{612}$$

Furthermore, note that upon integration over p, Eq. (609) yields the classical continuity equation:

$$\frac{\partial \rho(q;t)}{\partial t} = -\frac{\partial}{\partial q}\left[\rho(q;t)v(q;t)\right]. \tag{613}$$

To find the equation of motion of $v(q;t)$, we compute the time derivative of $f(p,q;t)$, after substituting Eq. (606) into Eq. (590):

$$\frac{\partial f(p,q;t)}{\partial t} = f(p,q;t)\frac{1}{\rho(q;t)}\frac{\partial \rho(q;t)}{\partial t} - m\frac{\partial v(q;t)}{\partial t}\frac{\partial f(p,q;t)}{\partial p}, \tag{614}$$

and, according to Eq. (613), we rewrite Eq. (614) as follows:

$$\frac{\partial f(p,q;t)}{\partial t} = -v'(q;t)f(p,q;t) - v(q;t)f(p,q;t)\frac{1}{\rho(q;t)}\frac{\partial \rho(q;t)}{\partial q} \\ - m\frac{\partial v(q;t)}{\partial t}\frac{\partial f(p,q;t)}{\partial p}. \tag{615}$$

Substituting Eqs. (605) and (607) into Eq. (574), we obtain:

$$\frac{\partial f(p,q;t)}{\partial t} = \frac{\partial V(q)}{\partial q}\frac{\partial f(p,q;t)}{\partial p} + \frac{1}{2\pi\hbar}\int dy e^{\frac{i}{\hbar}y(p-mv(q;t))}\sqrt{\rho(q+\frac{y}{2};t)\rho(q-\frac{y}{2};t)}\left[V_Q(q-\frac{y}{2};t)-\right. \\ -V_Q(q+\frac{y}{2};t) - \frac{\partial ln\sqrt{\rho(q+\frac{y}{2};t)}}{\partial q}v(q+\frac{y}{2};t) - \frac{\partial ln\sqrt{\rho(q-\frac{y}{2};t)}}{\partial q}v(q-\frac{y}{2};t)+ \\ \left.+\frac{m}{2}v(q-\frac{y}{2};t)^2 - \frac{m}{2}v(q+\frac{y}{2};t)^2 - \frac{1}{2}\frac{\partial v(q-\frac{y}{2};t)}{\partial q} - \frac{1}{2}\frac{\partial v(q+\frac{y}{2};t)}{\partial q}\right], \tag{616}$$

where

$$V_Q(q;t) = -\frac{\hbar^2}{2m}\frac{1}{\rho^{1/2}(q;t)}\frac{\partial^2\rho^{1/2}(q;t)}{\partial q^2}. \tag{617}$$

Therefore, linearizing Eq. (616), we obtain:

$$\frac{\partial f(p,q;t)}{\partial t} = -v'(q;t)f(p,q;t) - v(q;t)f(p,q;t)\frac{1}{\rho(x;t)}\frac{\partial \rho(q;t)}{\partial q} \\ + \frac{\partial}{\partial q}\left[V(q) + V_Q(q;t) + \frac{1}{2}mv(q;t)^2\right]\frac{\partial f(p,q;t)}{\partial p}, \tag{618}$$

Comparing Eq. (618) and Eq. (615), we obtain the equation of motion for $v(q;t)$:

$$\frac{\partial}{\partial t}v(q;t) + v(q;t)\frac{\partial v(q;t)}{\partial q} = -\frac{1}{m}\frac{\partial}{\partial q}\left[V(q) + V_Q(q;t)\right]. \tag{619}$$

that is a non-homogeneous Burgers' equation with viscosity coefficient equal to zero. Computing the total derivative of $v(q;t)$, from Eq. (619), we obtain:

$$m\frac{d}{dt}v(q;t) = -\frac{\partial}{\partial q}\left[\frac{(mv(q;t)-p(t))^2}{2m} + V(q) + V_Q(q;t)\right], \tag{620}$$

or

$$\frac{d}{dt}(mv(q;t) - p(t)) = -\frac{\partial}{\partial q}\left[\frac{(mv(q;t) - p(t))^2}{2m} + V_Q(q;t)\right]. \tag{621}$$

Integrating Eq. (619) over q, we obtain the Hamilton-Jacobi equation:

$$\frac{\partial}{\partial t}S(q;t) + \frac{1}{2m}\left(\frac{\partial S(q;t)}{\partial q}\right)^2 + V(q) + V_Q(q;t) = 0, \tag{622}$$

for the Hamiltonian

$$H_Q\left(q, \frac{\partial S(q;t)}{\partial q}; t\right) = \frac{1}{2m}\left(\frac{\partial S(q;t)}{\partial q}\right)^2 + V(q) + V_Q(q;t),$$

$$H_Q(q, mv(q;t); t) = \frac{1}{2}mv(q;t)^2 + V(q) + V_Q(q;t), \tag{623}$$

where $S(q;t) = S(q, \tilde{p}; t)$ is the generating function of the canonical transformation

$$mv(q;t) = \frac{\partial S(q, \tilde{p}; t)}{\partial q},$$

$$\tilde{q} = \frac{\partial S(q, \tilde{p}; t)}{\partial \tilde{p}}, \tag{624}$$

with

$$\tilde{H}_Q(\tilde{q}, \tilde{p}; t) = \frac{\partial S(q, \tilde{p}; t)}{\partial t} + H_Q\left(q, \frac{\partial S(q, \tilde{p}; t)}{\partial q}; t\right) = 0. \tag{625}$$

Note that the conjugate variables $\tilde{q}$ and $\tilde{p}$ are constants of motion since

$$\frac{\partial \tilde{p}}{\partial t} = -\frac{\partial \tilde{H}_Q(\tilde{p}, \tilde{q}; t)}{\partial \tilde{q}} = 0,$$

$$\frac{\partial \tilde{q}}{\partial t} = \frac{\partial \tilde{H}_Q(\tilde{p}, \tilde{q}; t)}{\partial \tilde{p}} = 0. \tag{626}$$

Therefore,

$$\frac{dS(q, \tilde{p}; t)}{dt} = \frac{\partial S(q, \tilde{p}; t)}{\partial t} + \frac{\partial S(q, \tilde{p}; t)}{\partial q}\frac{\partial q}{\partial t} + \frac{\partial S(q, \tilde{p}; t)}{\partial \tilde{p}}\frac{\partial \tilde{p}}{\partial t}, \tag{627}$$

and, according to Eq. (625), Eq. (626) and Eq. (612), we obtain:

$$\frac{dS(q;t)}{dt} = v(q;t)p(t) - H_Q(q, mv(q;t); t), \tag{628}$$

or the formal integral solution,

$$S(q;t) = \int_0^t dt' \left[v(q;t')p(t') - H_Q(q, mv(q;t'); t')\right]. \tag{629}$$

144

Exact expectation values can be computed by using $f(p, q; t)$ as a phase-space density, as follows: Case 1: When the operators depend only on coordinates (e.g., $O(\hat{x})$), expectation values are computed as phase space averages using the classical expression of the operator:

$$\langle \Psi_t \mid O(\hat{x}) \mid \Psi_t \rangle = \int dp \int dx f(p, q; t) O(q), \tag{630}$$

since

$$\langle \Psi(t) \mid O(\hat{x}) \mid \Psi(t) \rangle = \int dq \rho^{1/2}(q; t) e^{-\frac{i}{\hbar} S(q;t)} O(q) \rho^{1/2}(q; t) e^{\frac{i}{\hbar} S(q;t)},$$
$$= \int dp \int dq \rho^{1/2}(q; t) O(q) \rho^{1/2}(q; t) \delta(p - mv(q; t)). \tag{631}$$

Case 2: Analogously, when the operators are first order in momenta (e.g., $\hat{p}$), expectation values are also computed as phase space averages using the classical expression of the corresponding operators:

$$\langle \Psi(t) \mid \hat{p} \mid \Psi(t) \rangle = \int dp \int dq f(p, q; t) p, \tag{632}$$

since

$$\langle \Psi(t) \mid \hat{p} \mid \Psi(t) \rangle = -i\hbar \int dq \rho_t^{1/2}(q) e^{-\frac{i}{\hbar} S(q;t)} \frac{\partial}{\partial q} \left[\rho^{1/2}(q; t) e^{\frac{i}{\hbar} S(q;t)} \right],$$
$$= -i\hbar \int dq \rho^{1/2}(q; t) \frac{\partial \rho^{1/2}(q; t)}{\partial q} + \int dq \rho(q; t) \frac{\partial S(q; t)}{\partial q}$$
$$= -i\hbar \frac{1}{2} \int dq \frac{\partial \rho(q; t)}{\partial q} + \int dx \rho(q; t) mv(q; t) \tag{633}$$
$$= \int dp \int dq \rho(q; t) p \delta(p - mv(q; t)).$$

Case 3: Operators that are second order in momenta (e.g., $\hat{p}^2$) are computed as phase space averages using the classical expression of the corresponding operators plus a correction factor of order $\hbar^2$:

$$\langle \Psi(t) \mid \hat{p}^2 \mid \Psi(t) \rangle = \int dp \int dq f(p, q; t) p^2 + \int dp \int dq f(p, q; t) \left[-\frac{\hbar^2}{\rho^{1/2}(q; t)} \frac{\partial^2}{\partial q^2} \rho^{1/2}(q; t) \right], \tag{634}$$

since

$$\langle \Psi(t) \mid \hat{p}^2 \mid \Psi(t) \rangle = -\hbar^2 \int dq \rho^{1/2}(q; t) e^{-\frac{i}{\hbar} S(q;t)} \frac{\partial^2}{\partial q^2} \left[\rho^{1/2}(q; t) e^{\frac{i}{\hbar} S(q;t)} \right],$$
$$= -\hbar^2 \int dq \rho^{1/2}(q; t) \left[\frac{\partial^2}{\partial q^2} \rho^{1/2}(q; t) - \rho^{1/2}(q; t) \left(\frac{\partial S(q; t)}{\partial q} \right)^2 \right] \tag{635}$$
$$- i\hbar \int dq \frac{\partial}{\partial x} \left(\rho(q; t) \frac{\partial S(q; t)}{\partial q} \right).$$

Note that, according to Eq. (613), the last term of Eq. (635) is equal zero.

145

34.7 Motion of Auxiliary Variables:

To find a solution of Eq. (575) in terms of auxiliary "hidden" coordinates and momenta {R(t),P(t)} that obey equations of motion similar to Hamilton's equations, we introduce the phase-space variables {R(t),P(t)} with the following definition:

$$\Psi_t^*(q+s/2)\Psi_t(q-s/2) \equiv \int dR_0 \int dP_0 e^{-\frac{i}{\hbar}P_t s}\delta(R_t - q)\rho_0^W(P_0, R_0). \tag{636}$$

Substituting Eq. (636) into Eq. (575) we obtain:

$$\frac{\partial \rho_t^W(p,q)}{\partial t} = \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds \int dR_0 \int dP_0\, e^{\frac{i}{\hbar}(p-P_t)s}\{\frac{i}{\hbar}\left[V(q+s/2) - V(q-s/2)\right]\delta(R_t - q)$$
$$- \frac{p}{m}\frac{\partial\delta(R_t - q)}{\partial q}\}\rho_0^W(P_0, R_0). \tag{637}$$

Furthermore, substituting Eq. (636) into Eq. (590) gives:

$$\rho_t^W(p,q) = \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds \int dR_0 \int dP_0\, e^{\frac{i}{\hbar}(p-P_t)s}\delta(R_t - q)\rho_0^W(P_0, R_0), \tag{638}$$

and computing the time-derivative of both sides of Eq. (638) we obtain:

$$\frac{\partial \rho_t^W(p,q)}{\partial t} = \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds \int dR_0 \int dP_0\, e^{\frac{i}{\hbar}(p-P_t)s}\left[-\frac{i}{\hbar}\frac{\partial P_t}{\partial t}s\delta(R_t - q) + \frac{\partial\delta(R_t - q)}{\partial t}\right]\rho_0^W(P_0, R_0),$$
$$= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds \int dR_0 \int dP_0\, e^{\frac{i}{\hbar}(p-P_t)s}\left[-\frac{i}{\hbar}\frac{\partial P_t}{\partial t}s\delta(R_t - q) + \frac{\partial\delta(R_t - q)}{\partial R_t}\frac{\partial R_t}{\partial t}\right]\rho_0^W(P_0, R_0),$$
$$= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds \int dR_0 \int dP_0\, e^{\frac{i}{\hbar}(p-P_t)s}\left[-\frac{i}{\hbar}\frac{\partial P_t}{\partial t}s\delta(R_t - q) - \frac{\partial\delta(R_t - q)}{\partial q}\frac{\partial R_t}{\partial t}\right]\rho_0^W(P_0, R_0). \tag{639}$$

Comparing the third line of Eq. (639) with Eq. (637) we obtain the equations of motion of {R(t),P(t)}:

$$\frac{\partial P_t}{\partial t} = F_w(R_t, s),$$
$$\frac{\partial R_t}{\partial t} = \frac{p}{m}, \tag{640}$$

where we have introduced the 'Wigner force':

$$F_w(R_t, s) = -\frac{[V(R_t + s/2) - V(R_t - s/2)]}{s}. \tag{641}$$

Note that, according to Eq. (640), R_t and P_t are *not* conjugate variables since P_t depends on s and therefore the integral over s in Eq. (637) is not equal to $\delta(p - P_t)$. In the classical limit, however, Eq. (640) becomes Hamilton's equations,

$$
\begin{aligned}
\frac{\partial P_t}{\partial t} &= F_c(R_t) = -\frac{\partial V(R_t)}{\partial R_t}, \\
\frac{\partial R_t}{\partial t} &= \frac{P_t}{m},
\end{aligned}
\tag{642}
$$

since, in this limit, P_t *is* independent of s and therefore the integral over s can be replaced by $\delta(P_t - p)$, as follows:

$$
\rho_\tau^W(p, q) = \int dR_0 \int dP_0 \, \delta(p - P_\tau^{(c)}) \delta(R_\tau^{(c)} - q) \rho_0^W(P_0, R_0),
\tag{643}
$$

that is the classically evolved Wigner-transform.

The numerical integration of Eq. (643) can be based on the Velocity-Verlet algorithm:

$$
\begin{aligned}
R_\tau^{(c)} &= R_0 + \tau P_0 + \frac{\tau^2}{2} \frac{F_c(R_0)}{m}, \\
P_\tau^{(c)} &= P_0 + \tau \frac{F_c(R_0) + F_c(R_\tau^{(c)})}{2m}.
\end{aligned}
\tag{644}
$$

34.8 Time-Sliced Wigner Transform (TSWT) Propagation

A time-sliced Wigner transform propagation scheme can be implemented by sampling initial coordinates and momenta according to the distribution $|\rho_0^W(P_0, R_0)|$ to obtain $\rho_\tau^W(p, q)$ by Monte Carlo integration, as follows:

$$
\begin{aligned}
\rho_\tau^W(p, q) &= \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} ds \int dR_0 \int dP_0 \, e^{\frac{i}{\hbar}(p - P_\tau)s} \delta(R_\tau - q) \rho_0^W(P_0, R_0), \\
&= \lim_{\gamma \to \infty} A_{\gamma,N} \sum_{j=1}^{N} \int_{-\infty}^{\infty} ds \, e^{-\gamma(R_\tau(j) - q)^2 + \frac{i}{\hbar} s(p - P_\tau(j))} Sgn[\rho_0^W(P_0(j), R_0(j))],
\end{aligned}
\tag{645}
$$

where $A_{\gamma,N} = \frac{1}{2\pi\hbar N} \sqrt{\frac{\gamma}{\pi}} \int dR_0 \int dP_0 |\rho_0^W(P_0, R_0)|$, $Sgn[x] = |x|/x$, and

$$
\lim_{\gamma \to \infty} \sqrt{\frac{\gamma}{\pi}} e^{-\gamma(R_\tau - q)^2} = \delta(R_\tau - q).
$$

Substituting $\rho_0^W(p, q)$ by $\rho_\tau^W(p, q)$, in Eq. (645) gives $\rho_{2\tau}^W(p, q)$, and repeating the procedure N times we obtain the time evolved Wigner transform at time at time $t = N\tau$, $\rho_t^W(p, q)$.

Analogously to Eq. (644), one can approximate R_τ and P_τ in Eq. (645), as follows:

$$
\begin{aligned}
R_\tau &= R_0 + \tau P_0 + \frac{\tau^2}{2} \frac{F_w(R_0, s)}{m}, \\
P_\tau &= P_0 + \tau \frac{(F_w(R_0, s) + F_w(R_\tau, s))}{2m},
\end{aligned}
\tag{646}
$$

and integrate Eq. (645) by averaging over all possible values of s. Note that the dominant contribution results from the classical trajectory $s = 0$ while quantum corrections result from trajectories with $s \neq 0$. Also, note that due to the oscillatory nature of the integrand as a function of s, the most significant quantum corrections result from values of s close to zero.

34.9 Semiclassical Time-Sliced Wigner Transform (SC-TSWT) Propagation

A more efficient (although more approximate) numerical integration, can be implemented by integrating out s analytically from Eq. (645). To this end, we must obtain first the explicit dependence of R_τ and P_τ on s by expanding $F_w(R_\tau, s)$ to second order in powers of s, an expression that can be obtained by expanding $V(R_t \pm s/2)$ around R_t as follows:

$$
\begin{aligned}
F_w(R_t, s) &= -\frac{1}{s}(V(R_t + s/2) - V(R_t - s/2)) = -\frac{1}{s}\sum_{j=0}^{\infty} 2\left(\frac{s}{2}\right)^{2j+1} \frac{\partial^{2j+1}V(R_t)}{\partial R_t^{2j+1}} \frac{1}{(2j+1)!} \\
&= \sum_{j=0}^{\infty}\left(\frac{s}{2}\right)^{2j} \frac{\partial^{2j}F_c(R_t)}{\partial R_t^{2j}} \frac{1}{(2j+1)!}, \\
&\approx F_c(R_t) + s^2 F_c''(R_t)\frac{1}{24} + \cdots
\end{aligned}
$$

(647)

Note that, according to Eq. (647), we can re-write Eq. (637), as follows:

$$
\frac{\partial \rho_t^W}{\partial t} = \sum_{j=0}^{\infty} \frac{\partial^{2j+1}V(R_t)}{\partial R_t^{2j+1}} \frac{1}{(2j+1)!} \left(\frac{\hbar}{2i}\right)^{2j} \frac{\partial^{2j+1}\rho_t^W}{\partial p^{2j+1}} - \frac{p}{m}\frac{\partial \rho_t^W}{\partial q}.
$$

(648)

Substituting Eq. (647) into (646) and using Eq. (644), we obtain:

$$
\begin{aligned}
R_\tau &= R_\tau^{(c)} + s^2 \frac{\tau^2}{2m}\frac{F_c''(R_0)}{24}, \\
P_\tau &= P_\tau^{(c)} + \tau\frac{s^2}{2m}\frac{(F_c''(R_0) + F_c''(R_\tau))}{24}.
\end{aligned}
$$

(649)

where we can approximate $F_c(R_\tau) \approx F_c(R_\tau^{(c)})$, $F_c'(R_\tau) \approx F_c'(R_\tau^{(c)})$ and $F_c''(R_\tau) \approx F_c''(R_\tau^{(c)})$ to obtain:

$$
\begin{aligned}
R_\tau &= R_\tau^{(c)} + \frac{\tau^2}{2m}\frac{s^2 F_c''(R_0)}{24}, \\
P_\tau &= P_\tau^{(c)} + \tau\left(\frac{s^2}{24}\frac{(F_c''(R_0) + F_c''(R_\tau^{(c)}))}{2m}\right).
\end{aligned}
$$

(650)

Substituting Eq. (650) into (645), we obtain:

$$
\begin{aligned}
\rho_\tau^W(p,q) &= \lim_{\gamma\to\infty} \sqrt{\frac{\gamma}{\pi}} \frac{1}{2\pi\hbar} \int dR_0 \int dP_0 \int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}(p-P_\tau)s} e^{-\gamma(R_\tau-q)^2} \rho_0^W(P_0,R_0), \\
&= \lim_{\gamma\to\infty} \sqrt{\frac{\gamma}{\pi}} \frac{1}{2\pi\hbar} \int dR_0 \int dP_0 \int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}\tau\left(s\frac{(p-P_\tau^{(c)})}{\tau}+s^3\frac{(F_c''(R_0)+F_c''(R_\tau^{(c)}))}{48m}\right)} \\
&\quad \times e^{-\gamma((R_\tau^{(c)}-q)+s^2\frac{\tau^2 F_c''(R_0)}{48m})^2} \rho_0^W(P_0,R_0),
\end{aligned}
\tag{651}
$$

giving

$$
\boxed{
\begin{aligned}
\rho_\tau^W(p,q) &= \lim_{\gamma\to\infty} \sqrt{\frac{\gamma}{\pi}} \frac{1}{2\pi\hbar} \int dR_0 \int dP_0 \int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}\tau\left(s\frac{(p-P_\tau^{(c)})}{\tau}+s^3\frac{(F_c''(R_0)+F_c''(R_\tau^{(c)}))}{48m}\right)} \\
&\quad \times e^{-\gamma(R_\tau^{(c)}-q)^2-\gamma s^4(\frac{\tau^2 F_c''(R_0))}{48m})^2-\gamma s^2(R_\tau^{(c)}-q)\frac{\tau^2 F_c''(R_0)}{24m}} \rho_0^W(P_0,R_0),
\end{aligned}
}
\tag{652}
$$

where the integral over s can be computed numerically.

An approximation can be obtained as follows,

$$
\begin{aligned}
\rho_\tau^W(p,q) &\approx \frac{1}{2\pi\hbar} \int dR_0 \int dP_0 \delta(R_\tau^{(c)}-q)\rho_0^W(P_0,R_0) \int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}\tau\left(s\frac{(p-P_\tau^{(c)})}{\tau}+s^3\frac{(F_c''(R_0)+F_c''(R_\tau^{(c)}))}{48m}\right)}, \\
&\approx \frac{1}{\pi\hbar} \int dR_0 \int dP_0 \delta(R_\tau^{(c)}-q)\rho_0^W(P_0,R_0) \int_0^{\infty} ds\, \cos\left(s\frac{(p-P_\tau^{(c)})}{\hbar}+\frac{(s\xi)^3}{3}\right), \\
&\approx \int dR_0 \int dP_0 \delta(R_\tau^{(c)}-q)\rho_0^W(P_0,R_0)\frac{1}{\xi_{R_0,P_0}}\frac{1}{\pi} \int_0^{\infty} ds\, \cos\left(sZ_{R_0,P_0}(p)+s^3/3\right),
\end{aligned}
\tag{653}
$$

where $\xi = \frac{1}{\hbar}\xi_{R_0,P_0}$ with $\xi_{R_0,P_0} = \left(\tau\hbar^2\frac{(F_c''(R_0)+F_c''(R_\tau^{(c)}))}{16m}\right)^{1/3}$, $Z_{R_0,P_0}(p) = \frac{(p-P_\tau^{(c)})}{\xi_{R_0,P_0}}$, giving:

$$
\boxed{
\rho_\tau^W(p,q) \approx \int dR_0 \int dP_0 \delta(R_\tau^{(c)}-q)\rho_0^W(P_0,R_0)\frac{1}{\xi_{R_0,P_0}} Ai\left(\frac{p-P_\tau^{(c)}}{\xi_{R_0,P_0}}\right),
}
\tag{654}
$$

where $Ai(x) = \frac{1}{\pi}\int_0^{\infty} ds\, \cos\left(sx+s^3/3\right)$ is the Airy function that satisfies the differential equation

149

$Ai'' - xAi = 0$ and the integral $\int_{-\infty}^{\infty} Ai(x - x_1)Ai(x_2 - x)dx = 2^{-1/3}Ai(\frac{x_2-x_1}{2^{1/3}})$. [3]

In the quadratic limit (*i.e.*, $\xi_{R_0,P_0} \to 0$), Eq. (655) becomes,

$$\rho_\tau^W(p, q) \approx \int dR_0 \int dP_0 \delta(R_\tau^{(c)} - q)\delta(p - P_\tau^{(c)})\rho_0^W(P_0, R_0). \tag{656}$$

Another approximation is obtained by keeping in Eq. 652 terms only up to second order in powers of s in the exponential to obtain:

$$\rho_\tau^W(p, q) \approx \lim_{\gamma \to \infty} \sqrt{\frac{\gamma}{\pi}} \frac{1}{2\pi\hbar} \int dR_0 \int dP_0 \int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}s(p - P_\tau^{(c)})}$$
$$\times e^{-\gamma(R_\tau^{(c)} - q)^2 - \gamma s^2(R_\tau^{(c)} - q)\frac{\tau^2 F_c''(R_0)}{24m}} \rho_0^W(P_0, R_0), \tag{657}$$

which gives

$$\boxed{\rho_\tau^W(p, q) \approx \lim_{\gamma \to \infty} \frac{1}{2\pi\hbar} \int dR_0 \int dP_0 \rho_0^W(P_0, R_0)e^{-\gamma(R_\tau^{(c)} - q)^2}\sqrt{\frac{\gamma}{\alpha}}e^{-\frac{(p - P_\tau^{(c)})^2}{4\hbar^2\alpha}}} \tag{658}$$

where

$$\alpha = \gamma(R_\tau^{(c)} - q)\frac{\tau^2 F_c''(R_0)}{24m}, \tag{659}$$

34.10 Time-Sliced Wigner Transform Path Integral

Concatenating Eq. (654), for 2 time-steps, we obtain:

$$\rho_{2\tau}^W(P_2, R_2) \approx \int dR_0 dP_0 \int dR_1 dP_1 \delta(R_1^\tau - R_2)\delta(R_0^\tau - R_1)\rho_0^W(P_0, R_0)$$
$$\times \frac{1}{\xi_{R_1,P_1}} \frac{1}{\xi_{R_0,P_0}} Ai\left(\frac{P_2 - P_1^\tau}{\xi_{R_1,P_1}}\right) Ai\left(\frac{P_1 - P_0^\tau}{\xi_{R_0,P_0}}\right), \tag{660}$$

---

[3]Note that

$$\int_{-\infty}^{\infty} Ai(x_2 - x)Ai(x - x_1)dx = \pi^{-2}\int_0^{\infty}\int_0^{\infty} dsds' \int dx \cos(s(x - x_1) + s^3/3)\cos(s'(x_2 - x) + s'^3/3)$$
$$= \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} dsds' e^{i(-sx_1 + s^3/3)}e^{i(s'x_2 + s'^3/3)}\frac{1}{2\pi}\int dx e^{ix(s - s')}$$
$$= \frac{1}{2\pi}\int_{-\infty}^{\infty} ds e^{i(s(x_2 - x_1) + (2^{1/3}s)^3/3)}$$
$$= 2^{-1/3}Ai\left(\frac{x_2 - x_1}{2^{1/3}}\right) \tag{655}$$

where $R_0^\tau = R_0 + \tau P_0 + \tau^2 F_c(R_0)/(2m)$, $P_0^\tau = P_0 + \tau(F_c(R_0) + F_c(R_0^\tau))/(2m)$, $R_1^\tau = R_1 + \tau P_1 + \tau^2 F_c(R_1)/(2m)$ and $P_1^\tau = P_1 + \tau(F_c(R_1) + F_c(R_1^\tau))/(2m)$. Therefore,

$$\rho_{2\tau}^W(P_2, R_2) \approx \int dR_0 dP_0 dP_1 \delta(R_1^\tau - R_2)\rho_0^W(P_0, R_0)$$
$$\times \frac{1}{\xi_{R_0^\tau, P_1}} \frac{1}{\xi_{R_0, P_0}} Ai\left(\frac{P_2 - P_1^\tau}{\xi_{R_0^\tau, P_1}}\right) Ai\left(\frac{P_1 - P_0^\tau}{\xi_{R_0, P_0}}\right), \tag{661}$$

since $R_1^\tau = R_0^\tau + \tau P_1 + \tau^2 F_c(R_0^\tau)/(2m)$ and $P_1^\tau = P_1 + \tau(F_c(R_0^\tau) + F_c(R_1^\tau))/(2m)$.

Assuming that the dominant contributions to the integral over P_1 result from $P_1 \approx P_0^\tau$, we can approximate Eq. (661) as follows:

$$\rho_{2\tau}^W(P_2, R_2) \approx \int dR_0 dP_0 \delta(R_1^\tau - R_2)\rho_0^W(P_0, R_0)\frac{1}{\xi_{R_0^\tau, P_0^\tau}} \frac{1}{\xi_{R_0, P_0}}$$
$$\times \int dP_1 Ai\left(\frac{P_2 - P_1 - \tau(F_c(R_0^\tau) + F_c(R_1^\tau))/(2m)}{\xi_{R_0^\tau, P_0^\tau}}\right) Ai\left(\frac{P_1 - P_0^\tau}{\xi_{R_0, P_0}}\right), \tag{662}$$

Furthermore, according to Eq. (655), we can rewrite Eq. (662) as follows:

$$\boxed{\rho_{2\tau}^W(P_2, R_2) \approx \int dR_0 dP_0 \delta(R_1^\tau - R_2)\rho_0^W(P_0, R_0)\frac{1}{\xi_{R_0, P_0}} \frac{\widetilde{\xi}_{R_0^\tau}}{\xi_{R_0^\tau, P_0^\tau} 2^{1/3}} Ai\left(\frac{P_2 - P_1^\tau}{\widetilde{\xi}_{R_0^\tau} 2^{1/3}}\right),} \tag{663}$$

where $P_1^\tau = P_0^\tau + \tau(F_c(R_0^\tau) + F_c(R_1^\tau))/(2m)$, $R_1^\tau = R_0^\tau + \tau P_0^\tau + \tau^2 F_c(R_0^\tau)/(2m)$ and $\widetilde{\xi}_{R_0^\tau} = \left(\tau \hbar^2 \frac{F_c''(R_0^\tau)}{8m}\right)^{1/3}$ since both ξ_{R_0, P_0} and $\xi_{R_0^\tau, P_0^\tau}$ are approximately equal to $\widetilde{\xi}_{R_0^\tau}$ and

$$I(P_2, P_0^\tau) = \int dP_1 Ai\left(\frac{P_2 - P_1 - \tau(F_c(R_0^\tau) + F_c(R_1^\tau))/(2m)}{\widetilde{\xi}_{R_0^\tau}}\right) Ai\left(\frac{P_1 - P_0^\tau}{\widetilde{\xi}_{R_0^\tau}}\right),$$
$$= \int dP_1 Ai\left(\alpha_2 - \frac{P_1}{\widetilde{\xi}_{R_0^\tau}}\right) Ai\left(\frac{P_1}{\widetilde{\xi}_{R_0^\tau}} - \alpha_1\right), \tag{664}$$
$$= \widetilde{\xi}_{R_0^\tau} \int dP_1 Ai\left(\alpha_2 - P_1\right) Ai\left(P_1 - \alpha_1\right) = \frac{\widetilde{\xi}_{R_0^\tau}}{2^{1/3}} Ai\left(\frac{\alpha_2 - \alpha_1}{2^{1/3}}\right),$$

with $\alpha_2 = \frac{P_2 - \tau(F_c(R_0^\tau) + F_c(R_1^\tau))/(2m)}{\widetilde{\xi}_{R_0^\tau}}$, and $\alpha_1 = \frac{P_0^\tau}{\widetilde{\xi}_{R_0^\tau}}$.

Repeating the concatenation procedure again we obtain:

$$\rho_{3\tau}^W(P_3, R_3) \approx \int dR_0 dP_0 \delta(R_2^\tau - R_3)\rho_0^W(P_0, R_0)\frac{1}{\xi_{R_0, P_0}} \frac{\widetilde{\xi}_{R_1}}{\xi_{R_1, P_1} 2^{1/3}} \frac{\widetilde{\xi}_{R_2}}{\xi_{R_2, P_2} 2^{1/3}} Ai\left(\frac{P_3 - P_2^\tau}{\widetilde{\xi}_{R_2} 2^{1/3}}\right), \tag{665}$$

151

and for an N-step propagation to time $t = N \times \tau$, we obtain:

$$
\begin{aligned}
\rho_t^W(P, R) &\approx \int dR_0 dP_0 \delta(R_{N-1}^\tau - R)\rho_0^W(P_0, R_0)\frac{\widetilde{\xi}_{R_1}}{\xi_{R_0,P_0}2^{1/3}}\frac{\widetilde{\xi}_{R_2}}{\xi_{R_1,P_1}2^{1/3}}\frac{\widetilde{\xi}_{R_3}}{\xi_{R_2,P_2}2^{1/3}} \cdots \\
&\times \frac{\widetilde{\xi}_{R_N}}{\xi_{R_{N-1},P_{N-1}}2^{1/3}}\frac{2^{1/3}}{\widetilde{\xi}_{R_N}}Ai\left(\frac{P - P_{N-1}^\tau}{\widetilde{\xi}_{R_{N-1}}2^{1/3}}\right), \\
&\approx \int dR_0 dP_0 \delta(R_{N-1}^\tau - R)\rho_0^W(P_0, R_0)\frac{2^{1/3}}{\widetilde{\xi}_{R_{N-1}}}Ai\left(\frac{P - P_{N-1}^\tau}{\widetilde{\xi}_{R_{N-1}}2^{1/3}}\right) \quad (666) \\
&\times \left(\frac{F_c''(R_1)}{F_c''(R_0) + F_c''(R_1)}\frac{F_c''(R_2)}{F_c''(R_1) + F_c''(R_2)} \cdots \frac{F_c''(R_N)}{F_c''(R_{N-1}) + F_c''(R_N)}\right)^{1/3}, \\
&\approx \frac{1}{2^{N/3}}\int dR_0 dP_0 \delta(R_{N-1}^\tau - R)\rho_0^W(P_0, R_0)\frac{2^{1/3}}{\widetilde{\xi}_{R_{N-1}}}Ai\left(\frac{P - P_{N-1}^\tau}{\widetilde{\xi}_{R_{N-1}}2^{1/3}}\right),
\end{aligned}
$$

35   Asymmetric Wigner Transform

We define the *asymmetric Wigner transform*, as follows:

$$
\begin{aligned}
\rho^{AW}(q, p; t) &= \frac{1}{2\pi\hbar}\int dx\, \psi_t^*(q)\psi_t(x)e^{-\frac{i}{\hbar}p(x-q)}, \\
&= \frac{1}{\sqrt{2\pi\hbar}}\int ds\, \psi_t^*(q)\psi_t(q + s)e^{-\frac{i}{\hbar}ps}, \quad (667) \\
&= \psi_t^*(q)e^{\frac{i}{\hbar}pq}\frac{1}{\sqrt{2\pi\hbar}}\int dx\, \psi_t(x)e^{-\frac{i}{\hbar}px},
\end{aligned}
$$

or in Dirac notation,

$$
\rho^{AW}(q, p; t) = 2\pi\hbar\langle\psi_t|q\rangle\langle q|p\rangle\langle p|\psi_t\rangle. \quad (668)
$$

Like the Wigner transform, introduced in the previous sections, $\rho^{AW}(q, p; t)$ could be used to compute expectation values (or ensemble averages) since its integral over momenta gives the probability density $|\psi_t(q)|^2$,

$$
\int dp\, \rho^{AW}(q, p; t) = \psi_t^*(q)\psi_t(q), \quad (669)
$$

while its integral over coordinates gives the Fourier transform probability density $|\widetilde{\psi}_t(p)|^2$,

$$
\int dq\, \rho^{AW}(q, p; t) = \widetilde{\psi}_t^*(p)\widetilde{\psi}_t(p). \quad (670)
$$

35.1 Time Evolution:

The equation of motion of $\rho_t^{AW}(p, q)$ can be obtained by computing the time-derivative of both sides of Eq. (667), and substituting the time-derivative to the wavefunctions by using the time-dependent Schrödinger equation

$$\frac{\partial \Psi(q; t)}{\partial t} = -\frac{1}{i\hbar} \frac{\hbar^2}{2m} \frac{\partial^2 \Psi(q; t)}{\partial q^2} + \frac{1}{i\hbar} V(q) \Psi(q; t),$$

$$\frac{\partial \Psi(x; t)}{\partial t} = -\frac{1}{i\hbar} \frac{\hbar^2}{2m} \frac{\partial^2 \Psi(x; t)}{\partial x^2} + \frac{1}{i\hbar} V(x) \Psi(x; t),$$

(671)

Thus, the time-derivative of the asymmetric Fourier transform is

$$
\begin{aligned}
\frac{\partial \rho^{AW}(p, q; t)}{\partial t} &= \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} dx\, e^{-\frac{i}{\hbar} p(x-q)} \left[\frac{\partial \Psi^*(q; t)}{\partial t} \Psi(x; t) + \Psi^*(q; t) \frac{\partial \Psi(x; t)}{\partial t} \right], \\
&= \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} dx\, e^{-\frac{i}{\hbar} p(x-q)} \{ \frac{i}{\hbar} \left[V(q) - V(x) \right] \Psi^*(q; t) \Psi(x; t) \\
&\quad - \frac{i}{\hbar} \frac{\hbar^2}{2m} \left[\Psi(x; t) \frac{\partial^2}{\partial q^2} \Psi^*(q; t) - \Psi^*(q; t) \frac{\partial^2}{\partial x^2} \Psi(x; t) \right] \}, \\
&= \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} ds\, e^{-\frac{i}{\hbar} ps} \{ \frac{i}{\hbar} \left[V(q) - V(s+q) \right] \Psi^*(q; t) \Psi(s+q; t) \\
&\quad - \frac{i}{\hbar} \frac{\hbar^2}{2m} \left[\Psi(s+q; t) \frac{\partial^2}{\partial q^2} \Psi^*(q; t) - \Psi^*(q; t) \frac{\partial^2}{\partial s^2} \Psi(s+q; t) \right] \}, \\
&= \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} ds\, e^{-\frac{i}{\hbar} ps} \{ \frac{i}{\hbar} \left[V(q) - V(s+q) \right] \Psi^*(q; t) \Psi(s+q; t) \\
&\quad - \frac{i}{\hbar} \frac{\hbar^2}{2m} \left[\Psi(s+q; t) \frac{\partial^2}{\partial q^2} \Psi^*(q; t) - \Psi^*(q; t) \frac{\partial^2}{\partial q^2} \Psi(s+q; t) \right] \}, \\
&= \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} ds\, e^{-\frac{i}{\hbar} ps} \{ \frac{i}{\hbar} \left[V(q) - V(s+q) \right] \Psi^*(q; t) \Psi(s+q; t) \\
&\quad - \frac{i}{\hbar} \frac{\hbar^2}{2m} \frac{\partial}{\partial q} \left[\Psi(s+q; t) \frac{\partial}{\partial q} \Psi^*(q; t) - \Psi^*(q; t) \frac{\partial}{\partial q} \Psi(s+q; t) \right] \}, \\
&= \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} ds\, e^{-\frac{i}{\hbar} ps} \{ \frac{i}{\hbar} \left[V(q) - V(s+q) \right] \Psi^*(q; t) \Psi(s+q; t) \\
&\quad - i\frac{\hbar}{2m} \frac{\partial^2}{\partial q^2} \left[\Psi(s+q; t) \Psi^*(q; t) \right] + i\frac{\hbar}{m} \frac{\partial}{\partial q} \frac{\partial}{\partial s} \left[\Psi^*(q; t) \Psi(s+q; t) \right] \}, \\
&= \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} ds\, e^{-\frac{i}{\hbar} ps} \{ \frac{i}{\hbar} \left[V(q) - V(s+q) \right] \Psi^*(q; t) \Psi(s+q; t) \\
&\quad - i\frac{\hbar}{2m} \frac{\partial^2}{\partial q^2} \left[\Psi(s+q; t) \Psi^*(q; t) \right] - \frac{p}{m} \frac{\partial}{\partial q} \left[\Psi^*(q; t) \Psi(s+q; t) \right] \},
\end{aligned}
$$

(672)

Therefore,

$$
\begin{aligned}
\frac{\partial \rho^{AW}(p,q;t)}{\partial t} &= \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} ds\, e^{-\frac{i}{\hbar}ps} \{ \frac{i}{\hbar} \left[-V'(q)s - V''(q)\frac{s^2}{2} - ... \right] \Psi^*(q;t)\Psi(q+s;t) \\
&\quad - i\frac{\hbar}{2m}\frac{\partial^2}{\partial q^2} \left[\Psi^*(q;t)\Psi(q+s;t) \right] - \frac{p}{m}\frac{\partial}{\partial q} \left[\Psi^*(q;t)\Psi(q+s;t) \right] \},
\end{aligned}
\tag{673}
$$

Equation (673) can also be written, as follows:

$$
\frac{\partial \rho_t^{AW}}{\partial t} = \sum_{j=1}^{\infty} \frac{\partial^j V(q)}{\partial q^j} \frac{1}{j!} (i\hbar)^{j-1} \frac{\partial^j \rho_t^{AW}}{\partial p^{2j+1}} - i\frac{\hbar}{2m}\frac{\partial^2 \rho_t^{AW}}{\partial q^2} - \frac{p}{m}\frac{\partial \rho_t^{AW}}{\partial q}.
\tag{674}
$$

35.2 Motion of Auxiliary Variables:

To integrate the equation of motion of the asymmetric Fourier transform in terms of auxiliary "hidden" coordinates and momenta $\{R(t),P(t)\}$ that obey equations of motion similar to Hamilton's equations, we introduce the phase-space variables $\{R(t),P(t)\}$ with the following definition:

$$
\Psi_t^*(q)\Psi_t(q+s) \equiv \int dR_0 \int dP_0 e^{\frac{i}{\hbar}P_t s}\delta(R_t - q)\rho^{AW}(P_0, R_0; 0).
\tag{675}
$$

Substituting into Eq. (673), we obtain:

$$
\begin{aligned}
\frac{\partial}{\partial t}\rho^{AW}(p,q;t) &= \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} ds \int dR_0 \int dP_0\, e^{-\frac{i}{\hbar}(p-P_t)s} \\
&\quad \times \left\{ \frac{i}{\hbar} \left[-\frac{\hbar^2}{2m}\frac{1}{\delta(R_t - q)}\frac{\partial^2\delta(R_t - q)}{\partial q^2} - V'(q)s - V''(q)\frac{s^2}{2} - ... \right] \delta(R_t - q) \right. \\
&\quad \left. - \frac{p}{m}\frac{\partial}{\partial q}\delta(R_t - q) \right\} \rho^{AW}(P_0, R_0; 0), \\
&= \frac{i}{2\pi\hbar^2} \int_{-\infty}^{\infty} ds \int dR_0 \int dP_0\, e^{-\frac{i}{\hbar}(p-P_t)s}\rho^{AW}(P_0, R_0; 0) \\
&\quad \times \left[-\frac{\hbar^2}{2m}\frac{1}{\delta(R_t - q)}\frac{\partial^2\delta(R_t - q)}{\partial q^2} - V'(q)s - V''(q)\frac{s^2}{2} - ... \right] \delta(R_t - q) \\
&\quad - \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} ds \int dR_0 \int dP_0\, e^{-\frac{i}{\hbar}(p-P_t)s}\rho^{AW}(P_0, R_0; 0)\frac{p}{m}\frac{\partial}{\partial q}\delta(R_t - q),
\end{aligned}
\tag{676}
$$

Substituting Eq. (675) into Eq. (667), we obtain:

$$
\rho^{AW}(p,q;t) = \frac{1}{2\pi\hbar} \int ds \int dR_0 \int dP_0 e^{\frac{i}{\hbar}(P_t-p)s}\delta(R_t - q)\rho^{AW}(P_0, R_0; 0),
\tag{677}
$$

154

and computing the time derivative of both sides of Eq. (677), we obtain:

$$\frac{\partial}{\partial t}\rho^{AW}(p,q;t) = \frac{i}{2\pi\hbar^2}\int ds \int dR_0 \int dP_0 e^{-\frac{i}{\hbar}(p-P_t)s}\rho^{AW}(P_0,R_0;0)\left[\dot{P}_t s\right]\delta(R_t-q)-$$
$$-\frac{1}{2\pi\hbar}\int ds \int dR_0 \int dP_0 e^{-\frac{i}{\hbar}(p-P_t)s}\rho^{AW}(P_0,R_0;0)\dot{R}_t\frac{\partial}{\partial q}\delta(R_t-q),$$

(678)

A Gaussian approach can be introduced with the following definition:

$$\Psi_t^*(q)\Psi_t(q+s) \equiv \int dR_0 \int dP_0 e^{\frac{i}{\hbar}P_t s}G(q-R_t)\tilde{\rho}^{AW}(P_0,R_0;0),$$

(679)

where $G(r) = (\gamma/\pi)^{1/4}exp(-\gamma r^2/2)$ and $\tilde{\rho}^{AW}(P_0,R_0;0) = \int dxG(R_0-x)\rho^{AW}(P_0,x;0)$, giving

$$\rho^{AW}(p,q;t) = \frac{1}{2\pi\hbar}\int ds \int dR_0 \int dP_0 e^{\frac{i}{\hbar}(P_t-p)s}G(R_t-q)\tilde{\rho}^{AW}(P_0,R_0;0),$$

(680)

and

$$\frac{\partial}{\partial t}\rho^{AW}(p,q;t) = \frac{1}{2\pi\hbar}\int ds \int dR_0 \int dP_0 e^{-\frac{i}{\hbar}(p-P_t)s}\tilde{\rho}^{AW}(P_0,R_0;0)\left[\frac{i}{\hbar}\dot{P}_t s\right]G(R_t-q)-$$
$$-\frac{1}{2\pi\hbar}\int ds \int dR_0 \int dP_0 e^{-\frac{i}{\hbar}(p-P_t)s}\tilde{\rho}^{AW}(P_0,R_0;0)\dot{R}_t\frac{\partial}{\partial q}G(R_t-q).$$

(681)

Also, according to Eq. (676),

$$\frac{\partial}{\partial t}\rho^{AW}(p,q;t) = \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds \int dR_0 \int dP_0 \, e^{-\frac{i}{\hbar}(p-P_t)s}$$
$$\times \left\{\frac{i}{\hbar}\left[-\frac{\hbar^2}{2m}\frac{1}{G(R_t-q)}\frac{\partial^2 G(R_t-q)}{\partial q^2} - V'(q)s - V''(q)\frac{s^2}{2} - ...\right]G(R_t-q)\right.$$
$$\left.-\frac{p}{m}\frac{\partial}{\partial q}G(R_t-q)\right\}\tilde{\rho}^{AW}(P_0,R_0;0),$$
$$= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds \int dR_0 \int dP_0 \, e^{-\frac{i}{\hbar}(p-P_t)s}G(R_t-q)\tilde{\rho}^{AW}(P_0,R_0;0)$$
$$\times \left\{\frac{i}{\hbar}\left[-\frac{\hbar^2}{2m}(\gamma^2(R_t-q)^2-\gamma) - V'(q)s - V''(q)\frac{s^2}{2} - ...\right] - \frac{p}{m}\gamma(R_t-q)\right\},$$

(682)

36   Bohmian Quantum Dynamics

The goal of this section is to introduce the DeBroglie-Bohm formulation of quantum dynamics in terms of the trajectories of auxiliary (*i.e.*, "hidden") coordinates and momenta $q(t)$ and $p(t)$, as presented by David Bohm in [*Phys. Rev.* (1952) **65**:166-179] and [*Phys. Rev.* (1952) **65**:180-193].

To introduce this formulation, we first review the Hamilton-Jacobi equation of classical mechanics and we show how to use it to compute the trajectory of a system in phase-space as defined by the time-dependent coordinates and momenta $q(t)$ and $p(t)$. Then, we find a solution of the time-dependent Schrödinger equation

$$
\begin{aligned}
i\hbar\frac{\partial \Psi_t(q)}{\partial t} &= \hat{H}\Psi_t(q), \\
&= -\frac{\hbar^2}{2m}\frac{\partial^2 \Psi_t(q)}{\partial q^2} + V(q)\Psi_t(q),
\end{aligned}
\tag{683}
$$

in terms of auxiliary variables $q(t)$ and $p(t)$ that obey the classical Hamilton-Jacobi equation in the limit when $\hbar \to 0$. Furthermore, we show that when $\hbar \neq 0$, the equations of motion of $q(t)$ and $p(t)$ satisfy the same Hamilton-Jacobi equation but with a potential that includes not only the "classical" potential $V(q)$ but also a "quantum" potential $V_Q(q)$ determined by the "quantum field" $\Psi_t(q)$ that is the solution of the time-dependent Schrödinger equation introduced by Eq. (683).

36.1 Hamilton Jacobi Equation

We consider the Hamiltonian

$$
H(q,p) = \frac{p^2}{2m} + V(q),
\tag{684}
$$

and we define a canonical transformation in terms of the generating function $S = S(q, p_0, t)$ as follows:

$$
\begin{aligned}
p &= \frac{\partial S(q, p_0, t)}{\partial q}, \\
q_0 &= \frac{\partial S(q, p_0, t)}{\partial p_0},
\end{aligned}
\tag{685}
$$

with

$$
\boxed{\frac{\partial S(q, p_0, t)}{\partial t} + H(q, \frac{\partial S(q, p_0, t)}{\partial q}) = 0.}
\tag{686}
$$

and

$$
\widetilde{H}(q_0, p_0, t) \equiv \frac{\partial S(q, p_0, t)}{\partial t} + H(q, \frac{\partial S(q, p_0, t)}{\partial q}),
\tag{687}
$$

the Hamiltonian of the system for the transform variables q_0 and p_0. We note that the new conjugate variables q_0 and p_0 are constant in time since according to Eqs. (686) and (687), $\widetilde{H}(q_0, p_0, t) = 0$. Therefore,

$$
\begin{aligned}
\dot{q}_0 &= \frac{\partial \widetilde{H}(q_0, p_0, t)}{\partial p_0} = 0, \\
\dot{p}_0 &= -\frac{\partial \widetilde{H}(q_0, p_0, t)}{\partial q_0} = 0.
\end{aligned}
\tag{688}
$$

156

Equation (686) is the *Hamilton-Jacobi* equation and can be used to find te classical trajectory of coordinates and momenta $q(t)$ and $p(t)$, as follows: First, solve Eq. (686) for $S(q, p_0, t)$. Then, compute $p(t) = \partial S/\partial q$ by partial differentiation as defined in Eq. (1031). Finally, obtain $q(t)$ by first computing $q_0 = \partial S/\partial p_0$ and then solving for $q(t)$ as a function of q_0 and p_0 that are constant in time (Eq. (688)).

36.2 Method of Characteristics

Equation (686) is a partial differential equation (PDE) that can be solved by transforming it into a set of ordinary differential equations (ODE's), according to the *method of characteristics* introduced by Hamilton in 1833 [*Dublin University Review and Quarterly Magazine* (1833) **1**:795-826]. For a first-order PDE like Eq. (686), the method of characteristics finds the curves (called characteristic curves or just characteristics) along which the PDE becomes an ordinary differential equation (ODE). Once the ODE is found, it can be solved along the characteristics and transformed into a solution for the original PDE.

We rewrite Eq. 686 as

$$F(S, \nabla S, q) = 0, \tag{689}$$

with

$$\begin{aligned}
F(S, \nabla S, q) &= \frac{\partial S(q, P, t)}{\partial t} + H\left(q, \frac{\partial S(q, P, t)}{\partial q}\right), \\
&= \frac{\partial S(q, P, t)}{\partial t} + \frac{(\nabla S)^2}{2m} + V(q),
\end{aligned} \tag{690}$$

and

$$\frac{\partial F}{\partial S} \nabla S + \frac{\partial F}{\partial \nabla S} \nabla^2 S + \frac{\partial F}{\partial q} = 0. \tag{691}$$

We now consider $q(t)$ to be a parametric representation of a characteristic curve (that will be obtained later), giving:

$$z(t) = S(q(t)), \tag{692}$$

and

$$\begin{aligned}
p(t) &= \left.\frac{\partial S}{\partial q}\right|_{q=q(t)}, \\
\dot{p}(t) &= \left.\frac{\partial^2 S}{\partial q^2}\right|_{q=q(t)} \dot{q}(t).
\end{aligned} \tag{693}$$

Evaluating Eq. (691) at $q = q(t)$, we obtain:

$$\begin{aligned}
\frac{\partial F(z, p, q)}{\partial z} p(t) + \frac{\partial F(z, p, q)}{\partial p} \frac{\dot{p}}{\dot{q}} + \frac{\partial F(z, p, q)}{\partial q} &= 0, \\
\frac{\partial F(z, p, q)}{\partial p} \frac{\dot{p}}{\dot{q}} + \frac{\partial F(z, p, q)}{\partial q} &= 0,
\end{aligned} \tag{694}$$

since, according to Eq. (690), $\frac{\partial F(z,p,q)}{\partial z} = 0$.

Defining the characteristic curve, as follows:

$$\dot{q}(t) = \frac{\partial F(z,p,q)}{\partial p},$$
$$= \frac{p(t)}{m}, \tag{695}$$

we obtain

$$\dot{p}(t) = -\frac{\partial F(z,p,q)}{\partial q},$$
$$= -\frac{\partial V(q)}{\partial q}. \tag{696}$$

Finally, differentiating $z(t)$, we obtain:

$$\dot{z}(t) = \frac{\partial S}{\partial q}\Big|_{q=q(t)}\dot{q}(t) + \frac{\partial S}{\partial t}\Big|_{q=q(t)},$$
$$= p(t)\dot{q}(t) - H(q(t), p(t)), \tag{697}$$

giving

$$z(t) = S(q(t)) = \int_0^t dt' p(t')\dot{q}(t') - H(q(t'), p(t')), \tag{698}$$

with $q(0) = q_0$.

The solutions of the ODE's (695), and (696) –known as Hamilton's equations, allow for calculation of $S(q(t))$ according to Eq. (698).

Note: The method of characteristics, implemented above, is useful for solving first-order PDE's, in general. As a simple illustrative example, we consider the following PDE:

$$G(x, f', f) = \frac{\partial f(x,t)}{\partial t} - 2\frac{\partial f(x,t)}{\partial x} - 5f(x,t) - k(x,t) = 0. \tag{699}$$

We introduce the usual functions

$$z(t) = f(x(t)), \tag{700}$$

and

$$p(t) = \frac{\partial f}{\partial x}\Big|_{x=x(t)}. \tag{701}$$

Differentiating $p(t)$, we obtain:

$$\dot{p}(t) = \frac{\partial^2 f}{\partial x^2}\Big|_{x=x(t)}\dot{x}(t), \tag{702}$$

and differentiating G with respect to x, we obtain:

$$\frac{\partial G}{\partial f}\nabla f + \frac{\partial G}{\partial \nabla f}\nabla^2 f + \frac{\partial G}{\partial x} = 0. \tag{703}$$

158

Evaluating Eq. (703) at $x = x(t)$, we obtain:

$$\frac{\partial G(z, p, x)}{\partial z} p(t) + \frac{\partial G(z, p, x)}{\partial p} \frac{\dot{p}}{\dot{x}} + \frac{\partial G(z, p, x)}{\partial x} = 0. \tag{704}$$

Defining the characteristic curve, as follows:

$$\dot{x}(t) = \frac{\partial G(z, p, x)}{\partial p}, \tag{705}$$
$$= -2,$$

we obtain $x(t) = x_0 - 2t$ and

$$\dot{p}(t) = -\frac{\partial G(z, p, x)}{\partial z} p(t) - \frac{\partial G(z, p, x)}{\partial x}, \tag{706}$$
$$= 5p(t) + \frac{\partial k}{\partial x}.$$

Finally, differentiating $z(t)$, we obtain:

$$\dot{z}(t) = \frac{\partial f}{\partial x}\bigg|_{x=x(t)} \dot{x}(t) + \frac{\partial f}{\partial t}\bigg|_{x=x(t)}, \tag{707}$$

giving the ODE,

$$\dot{z}(t) - 5z(x, t) - k(x, t) = 0, \tag{708}$$

that can be solved by multiplying the whole equation by e^{-5t},

$$0 = e^{-5t}\dot{z}(t) - 5e^{-5t}z(x, t) - e^{-5t}k(x, t),$$
$$0 = \frac{\partial}{\partial t}e^{-5t}z - e^{-5t}k(x, t) \tag{709}$$

that gives

$$z(t) = e^{5t} \int_0^t dt' e^{-5t'} k(x_0 - 2t', t'). \tag{710}$$

Perhaps the most pedagogical example of the method of characteristics is its implementation for solving the homogeneous transport equation:

$$\frac{\partial f}{\partial t} + \alpha \frac{\partial f}{\partial x} = 0. \tag{711}$$

along the characteristic curve $x = q(t)$, with $\dot{q}(t) = \alpha$, subject to the initial condition $f(x; 0) = g(x)$. Along the characteristic curve, that initial condition gives $f(q(0); 0) = g(q(0))$. We note that, according to Eq. (713), f is constant along the characteristic curve since

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial q}\dot{q}, \tag{712}$$

159

and $\dot{q}(t) = \alpha$. Therefore, after finding the curve $q(t) = q(0) + \alpha t$, we can obtain f at any point along that curve since $f(q(t);t) = f(q(0);0) = g(q(0)) = g(q(t) - \alpha t)$.

Another example is the inhomogeneous transport equation

$$\frac{\partial f}{\partial t} + \alpha \frac{\partial f}{\partial x} = -\beta f, \tag{713}$$

with the same initial conditions $f(x;0) = g(x)$. Due to the inhomogeneous term on the r.h.s. of Eq. (713), f is no longer constant along the characteristics:

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial q}\dot{q} = -\beta f. \tag{714}$$

In fact, f decays exponentially along the characteristics $q(t)$, as follows

$$\begin{aligned} f(q(t);t)) &= f(q(0);0)e^{-\beta t}, \\ &= g(q(t) - \alpha t)e^{-\beta t}. \end{aligned} \tag{715}$$

36.3 Quantum Dynamics: Motion of Hidden Variables

To solve the time-dependent Schrödinger equation, introduced by Eq. (683), we write $\Psi_t(x)$ in terms of the amplitude $A_t(x)$ and phase $S_t(x)$ functions, as follows:

$$\Psi_t(q) = A_t(q)e^{\frac{i}{\hbar}S_t(q)}, \tag{716}$$

where $A_t(q)$ and $S_t(q)$ are defined as real functions. Substituting Eq. (716) into Eq. (683), we obtain:

$$i\hbar\frac{\partial\Psi_t(q)}{\partial t} = \left[i\hbar\frac{\partial A_t(q)}{\partial t} - A_t(q)\frac{\partial S_t(q)}{\partial t}\right]e^{\frac{i}{\hbar}S_t(q)},$$

$$\hat{H}\Psi_t(q) = \left[-\frac{\hbar^2}{2m}\frac{\partial^2 A}{\partial q^2} - \frac{i\hbar}{m}\frac{\partial A}{\partial q}\frac{\partial S}{\partial q} - \frac{i\hbar}{2m}A\frac{\partial^2 S}{\partial q^2} + \frac{1}{2m}A_t(q)\left(\frac{\partial S}{\partial q}\right)^2 + A(q)V(q)\right]e^{\frac{i}{\hbar}S_t(q)}. \tag{717}$$

Since the first line of Eq. (717) must be equal to the second line, the real parts of the left hand sides of Eq. (717) must be equal:

$$\frac{\partial S_t(q)}{\partial t} + \frac{1}{2m}\left(\frac{\partial S}{\partial q}\right)^2 + V(q) + V_Q(q,t) = 0, \tag{718}$$

where

$$V_Q(q,t) = -\frac{\hbar^2}{2m}\frac{\partial^2 A_t(q)}{\partial q^2}\frac{1}{A_t(q)}, \tag{719}$$

160

is a time-dependent "quantum" potential determined by A_t. Note that Eq. (718) is the Hamilton-Jacobi equation for a system described by the Hamiltonian:

$$H(q,p) = \frac{p^2}{2m} + V(q) + V_Q(q,t), \tag{720}$$

with $p = \partial S / \partial q$, and $V_Q(q,t)$ the time-dependent "external" field potential defined by Eq. (719).

Furthermore, since the imaginary parts of the left hand sides of Eq. (717) must be equal, we obtain:

$$\frac{\partial A_t(q)}{\partial t} + \frac{\partial A_t(q)}{\partial q}\frac{\partial S_t(q)}{\partial q}\frac{1}{m} + A_t(q)\frac{\partial^2 S_t(q)}{\partial q^2}\frac{1}{2m} = 0. \tag{721}$$

Making the substitutions $A_t(q) = \sqrt{\Psi_t^*(q)\Psi_t(q)} = \rho_t^{1/2}$ and $p = \partial S / \partial q$ into Eq. (721), gives:

$$\frac{1}{2}\rho_t^{-1/2}\frac{\partial \rho_t}{\partial t} + \frac{1}{2}\rho_t^{-1/2}\frac{\partial \rho_t}{\partial q}\frac{\partial S_t(q)}{\partial q}\frac{1}{m} + \rho_t^{1/2}\frac{\partial^2 S_t(q)}{\partial q^2}\frac{1}{2m} = 0,$$

$$\frac{\partial \rho_t}{\partial t} + \frac{\partial \rho_t}{\partial q}\frac{\partial S_t(q)}{\partial q}\frac{1}{m} + \rho_t\frac{\partial^2 S_t(q)}{\partial q^2}\frac{1}{m} = 0, \tag{722}$$

$$\frac{\partial \rho_t}{\partial t} + \frac{\partial j_t}{\partial q} = 0$$

that is the continuity equation for the classical current $j_t = \rho_t v$, with $v = \frac{p}{m}$.

36.4 Discussion of Bohmian Trajectories

The equations of motion of the auxiliary variables $q(t)$ and $p(t)$ are Hamilton's equations

$$\dot{q}(t) = \frac{\partial H(q,p)}{\partial p},$$

$$\dot{p}(t) = -\frac{\partial H(q,p)}{\partial q}, \tag{723}$$

with $H(q,p)$ defined according to Eq. (720). The variables $q(t)$ and $p(t)$ define the actual coordinates and momenta of the quantum system with unlimited precision. However, they are not observable but "hidden" quantities since measurements can only determine ensemble averages over all possible trajectories, as determined by the initial conditions. Measuring devices interact with the system by means of indivisible quanta that introduce irreducible disturbances during the measurement process, or preparation of the initial state. Only if the precise effects of those disturbances could be corrected for, one could determine $q(t)$ and $p(t)$ and have simultaneous measurements of momentum and position with unlimited precision.

36.5 EPR Paradox

Gedankenexperiments (*i.e.*, thought experiments) have been proposed to determine "hidden" variables. The most famous of these proposals has been the Einstein-Podolski-Rosen (EPR) *gedankenexperiment* [*Phys. Rev.* (1935) **47**:777-780], where a system of 2 particles is initially prepared

with total momentum p_t. At a later time, when the two particles are far apart from each other, the position x_1 is measured on particle 1 and the momentum p_2 is measured on particle 2. The paradox is that the momentum of particle 1 could be obtained from the difference $p_1 = p_t - p_2$. Therefore, the coordinate x_1 and momentum p_1 of particle 1 could be determined with more precision than established as possible by the uncertainty principle, so long as the separation between the two particles could prevent any kind of interaction or disturbance of one particule due to a measurement on the other.

The origin of the paradox is the erroneous assumption that particles that are far apart from each other cannot maintain instantaneous correlations. However, quantum correlations between the properties of distant noninteracting systems *can* be maintained, as described by Bohm and Aharonov [*Phys. Rev.* (1957) **108**:1070-1076] for the state of polarization of pairs of correlated photons. Within the Bohmian picture of quantum mechanics, these quantum correlations are established by the quantum potential $V_Q(q)$, even when the particles are noninteracting (*i.e.*, $V(q) = 0$).

Quantum correlations between distant noninteracting photons were observed for the first time by Aspect and co-workers in 1982 [*Phys. Rev. Lett.* (1982) **49**:91-94], 47 years after the EPR paradox was presented. These quantum correlations constitute the fundamental physics exploited by *teleportation* (*i.e.*, the transmission and reconstruction of quantum states over arbitrary large distances) [*Nature* (1997) **390**:575-579] and *ghost imaging* (*i.e.*, a technique where the object and the image system are on separate optical paths) [*Am. J. Phys.* (2007) **75**:343-351].

37   Measures of Information

The goal of this section is to show that the *Shannon entropy*,

$$S = -\sum_j P_j \, \log_2 P_j, \tag{724}$$

gives *on average* the *number* of bits needed to store or communicate a 'symbol' of a 'message' composed of symbols x_j with probabilty P_j. This is usually stated as Shannon's source coding theorem: *It is impossible to compress the data of a long stream of a random variable such that the average number of bits per symbol is less than the Shannon entropy of the source without loosing information.*

As an example, we consider that the 'message' is a set of n snapshots reporting the position of a particle. The more snapshots (*i.e.*, 'symbols'), the more information the message has with regards to the particle whereabouts. Thefore, the amount of information S must scale with the number n of symbols. For example, if storing each symbol x_j requires K bits, then $S = Kn$.

We anticipate that the number of bits required to store a symbol (*i.e.*, K) is determined by the total number of possible symbols $\Omega = 2^K$ since each symbol can be mapped into one of the possible states of a register with K bits (*i.e.*, an array of K elements that can be either 1 or 0). We note that $K = -log_2(1/\Omega)$, where $1/\Omega$ is the probability of a symbol when randomly picked out of Ω equally probable choices.

More formally, when considering two messages, one with n_1 symbols each of which chosen from Ω_1 possible states, and the other one with n_2 symbols chosen from Ω_2 possibilities, the amount of information in the two messages is the same when the number of possibilities for the two messages is equal: $\Omega_1^{n_1} = \Omega_2^{n_2}$, as pointed out by Hartley in the *The Bell System Technical Journal*, July 1928, p. 535. This is simply because, in that case, $2^{K_1 n_1} = 2^{K_2 n_2}$ so the total number of bits required to store the two messages is the same: $K_1 n_1 = K_2 n_2$, from which we obtain $K_1/\log_2\Omega_1 = K_2/\log_2\Omega_2$. This relation holds for all values of Ω only when $K = K_0 log\Omega$, where the arbitrary constant K_0 can be omitted since the logarithmic base is also arbitrary (i.e., the particular base of the log fixes the size of the unit of information. When the base 2 is used the resulting units are called binary digits, or more briefly *bits*, as suggested by J.W. Tukey. If the base 10 is used the units may be called decimal digits, with $\log_2\Omega = \log_{10}\Omega/\log_{10}2$).

Therefore, as anticipated, the amount of information (*i.e*, the number of bits) of each symbol is the logarithm of the number of possible symbols, and the amount of information of a message is the number of possible messages (which is the number of symbols times the number of possible symbols: $S = n\log_2\Omega$). It is, therefore, clear that for a message composed of symbols with probability P_j, the average number of bits necessary to store one of its symbols is given by Eq. (724), as discussed by Claude E. Shannon in his seminal work on communication theory published in *The Bell System Technical Journal* **27**: 379-423, 1948, with symbols drawn with probability P_j as randomly picked from $\Omega_j = 1/P_j$ equally probable choices.

Majorization: When comparing the entropies of two sources with probabilities $\mathbf{p} = (p_1, p_2, ..., p_n)$ with $p_{j+1} < p_j$ and $\mathbf{q} = (q_1, q_2, ..., q_n)$ with $q_{j+1} < q_j$, it is possible to show that $S(\mathbf{p}) < S(\mathbf{q})$

when **p** majorizes **q**, as follows:

$$\sum_{i=1}^{j} p_i > \sum_{i=1}^{j} q_i, \tag{725}$$

for any $j < n$ with

$$\sum_{i=1}^{n} q_i = \sum_{i=1}^{n} p_i = 1. \tag{726}$$

This can be shown by substracting Eq. (725) from Eq (726) to obtain:

$$\sum_{i=j+1}^{n} q_i > \sum_{i=j+1}^{n} p_i. \tag{727}$$

Therefore,

$$\sum_{j=1}^{n} \left[\sum_{i=j+1}^{n} q_i \right] \log \left(\frac{q(i)}{q(j+1)} \right) > \sum_{j=1}^{n} \left[\sum_{i=j+1}^{n} p_i \right] \log \left(\frac{q(i)}{q(j+1)} \right). \tag{728}$$

Changing the order of the sums in Eq. (728), we obtain:

$$\sum_{i=1}^{n} \left[\sum_{j=1}^{i-1} q_i \right] \log \left(\frac{q(i)}{q(j+1)} \right) > \sum_{i=1}^{n} \left[\sum_{j=1}^{i-1} p_i \right] \log \left(\frac{q(i)}{q(j+1)} \right), \tag{729}$$

and canceling equal terms within the sums over j, we obtain:

$$\sum_{i=1}^{n} q_i \log \left(\frac{q(1)}{q(i)} \right) > \sum_{i=1}^{n} p_i \log \left(\frac{q(1)}{q(i)} \right),$$

$$\log(q(1)) - \sum_{i=1}^{n} q_i \log(q(i)) > \log(q(1)) - \sum_{i=1}^{n} p_i \log(q(i)), \tag{730}$$

$$-\sum_{i=1}^{n} q_i \log \left(q(i) \right) > -\sum_{i=1}^{n} p_i \log \left(q(i) \right).$$

164

Furthermore, since $\log(x) < x - 1$, we obtain:

$$\log\left(\frac{q(i)}{p(i)}\right) < \frac{q(i)}{p(i)} - 1,$$

$$p(i)\log\left(\frac{q(i)}{p(i)}\right) < q(i) - p(i),$$

$$\sum_{i=1}^{n} p(i)\log\left(\frac{q(i)}{p(i)}\right) < 0, \tag{731}$$

$$\sum_{i=1}^{n} p(i)\log\left(q(i)\right) < \sum_{i=1}^{n} p(i)\log\left(p(i)\right),$$

$$-\sum_{i=1}^{n} p(i)\log\left(q(i)\right) > -\sum_{i=1}^{n} p(i)\log\left(p(i)\right).$$

Therefore, according to Eqs. (730) and (731), we obtain:

$$-\sum_{i=1}^{n} q(i)\log\left(q(i)\right) > -\sum_{i=1}^{n} p(i)\log\left(p(i)\right). \tag{732}$$

Maximum Entropy Image Reconstruction: Maximum entropy reconstruction is a widely applicable technique for generating images with maximum information, as discussed by Skillin and Bryan in the *Mon. Not. R. Astr. Soc.* (1984) **211**, 111-124. The image is regarded as a set of positive numbers $f_1, \cdots, f_N$ which are to be determined and give the intensity of pixels. The observational constraints on permitted reconstructions come from data points D_k related to the image in some known way, and subject to some form of noise. In a linear experiment, $D_k = \sum R_{kj} f_j + n_k \sigma_k$, where R is the instrument response , σ_k is the standard error on the data and n_k is a random variable of zero mean and unit variance. Naively one may attempt to recover the image P from the data D by applying R^{-1} but this usually fails whenever the data are incomplete since R^{-1} is not uniquely defined, or when R^{-1} is badly conditioned as in most deconvolution problems.

The reconstruction is set up by comparing the noisy data D_k with the simulated data F_k which would be obtained in the absence of noise: $F_k = \sum R_{kj} f_j$. A reconstruction is said to be *feasible* if the simulated data agree with the actual data to within the noise. The misfit is usually measured by the chi-squared value: $\chi^2 = \sum_k (F_k - D_k)^2 / \sigma_k^2$, where the summation is over the observed data points k.

According to the maximum entropy criteria, one selects the feasible image with greatest entropy: $S = -\sum_k P_k \log_2 P_k$, where $P_k = f_k / \sum_j f_j$, by maximizing S subject to minimum χ^2. Using the Lagrange multiplier method for $Q = S - \lambda \chi^2 - \mu(1 - \sum_k P_k)$, with some value of the Lagrange multipliers λ and μ, we make $\partial Q / \partial f_j = 0$ to obtain f_j.

As pointed out by Skilling and Bryan, the resulting maximum entropy fitting provides a unique approach to reconstruct an image without introducing correlations beyond those which are required by the data. Maximizing the entropy can also be justified in terms of information. Given an image radiating with intensity pattern f_j, the entropy measures the average number of bits needed to

define (or store) the position j of a single radiated photon. Maximizing S, subject to observational constraints, involves seeking a maximally non-biased answer to the fundamental question 'Where would the next photon come from?'. The practical merit of maximizing entropy is that the resulting image has minimum configurational information, so that there must be evidence in the data for any structure which is seen, and the displayed structure is uniquely easy to comprehend.

37.1 Fisher Index

Another measure of information is the Fisher index I, defined as follows:

$$
\begin{aligned}
I &= \int_{-\infty}^{\infty} dx P(x) \left(\frac{1}{P(x)} \frac{\partial P(x)}{\partial x} \right)^2, \\
&= \int_{-\infty}^{\infty} dx P(x) \left(\frac{\partial \log[P(x)]}{\partial x} \right)^2
\end{aligned}
\tag{733}
$$

For an ergodic system, $P(x)$ is determined by the time the variable is found at x over the total observation time, so the total probability is normalized:

$$
\begin{aligned}
1 &= \int_{-\infty}^{\infty} dx P, \\
&= \lim_{t_f \to \infty} \frac{1}{t_f} \int_{-t_f/2}^{t_f/2} dt, \\
&= \lim_{t_f \to \infty} \frac{1}{t_f} \int_{-\infty}^{\infty} dx \left| \frac{\partial t}{\partial x} \right|, \\
&= \lim_{t_f \to \infty} \frac{1}{t_f} \int_{-\infty}^{\infty} dx \frac{1}{\sqrt{v^2}},
\end{aligned}
\tag{734}
$$

from where we identify $1/P = t_f \sqrt{v^2}$, with $t_f \to \infty$. Therefore,

$$
\begin{aligned}
I &= \lim_{t_f \to \infty} t_f \int_{-\infty}^{\infty} dx \sqrt{v^2} \left(\frac{\partial P}{\partial x} \right)^2, \\
&= \lim_{t_f \to \infty} t_f \int_{-\infty}^{\infty} dx v^2 \left| \frac{\partial t}{\partial x} \right| \left(\frac{\partial P}{\partial x} \right)^2, \\
&= \lim_{t_f \to \infty} t_f \int_{-\infty}^{\infty} dx \left| \frac{\partial t}{\partial x} \right| \left(\frac{\partial P}{\partial t} \right)^2, \\
&= \lim_{t_f \to \infty} t_f \int_{-\infty}^{\infty} dt \left(\frac{\partial P}{\partial t} \right)^2,
\end{aligned}
\tag{735}
$$

In the third row of Eq. (735), we have used that $\partial P/\partial x = -1/v \, \partial P/\partial t$, since according to the continuity equation $dP/dt = \partial P/\partial t + v \partial P/\partial x$ and $dP/dt = 0$ for the system at equilibrium.

Furthermore,

$$
\begin{aligned}
\frac{\partial P}{\partial t} &= \frac{1}{t_f} \frac{\partial}{\partial t} (v^2)^{-1/2}, \\
&= \frac{1}{t_f} \left(-\frac{1}{2} \right) (v^2)^{-3/2} 2v \frac{\partial v}{\partial t} \\
&= -\frac{1}{t_f} \frac{1}{v^2} \frac{\partial v}{\partial t} = -\frac{1}{t_f} \frac{a}{v^2} = -\frac{1}{t_f m} \frac{F}{v^2}.
\end{aligned}
\tag{736}
$$

Therefore,

$$
\begin{aligned}
I &= \lim_{t_f \to \infty} \frac{1}{t_f} \int_{-t_f/2}^{t_f/2} dt \frac{F^2}{(mv^2)^2}, \\
&= \left\langle \frac{F^2}{(mv^2)^2} \right\rangle,
\end{aligned}
\tag{737}
$$

37.2 Mutual Information

The goal of this section is to introduce measures of mutual information (or uncertainty) shared by variables x_1 and x_2. Such measures estimate the amount of uncertainty about a variable x_1 reduced by measuring a variable x_2. In particular, when x_1 and x_2 are completely uncorrelated, knowing one of the two variables does not provide any information about the other and the joint probability distribution $P(x_1, x_2)$ is equal to the product of the marginal probability distributions $P_1(x_1) = \int dx_2 P(x_1, x_2)$ and $P_2(x_2) = \int dx_1 P(x_1, x_2)$.

The most common measure of mutual information is the index I_{x_1, x_2}, defined as follows:

$$
I_{x_1, x_2} = \int dx_1 dx_2 P(x_1, x_2) \left(\log[P(x_1, x_2)] - \log[P_1(x_1) P_2(x_2)] \right).
\tag{738}
$$

Note that in the particular case of uncorrelated variables, $I_{x_1, x_2} = 0$.

Analogously, another measure of mutual information can be defined, as follows:

$$
\tilde{I}_{x_1, x_2} = \int_{-\infty}^{\infty} dx_1 \int_{-\infty}^{\infty} dx_2 P \left(\frac{1}{P^2} \left(\frac{\partial P}{\partial x_1} \right)^2 + \frac{1}{P^2} \left(\frac{\partial P}{\partial x_2} \right)^2 - \frac{1}{P_1^2} \left(\frac{\partial P_1}{\partial x_1} \right)^2 - \frac{1}{P_2^2} \left(\frac{\partial P_2}{\partial x_2} \right)^2 \right).
\tag{739}
$$

38   Maxwell's Equations

The goal of this section is show that Maxwell's equations can be written in the form of a Schrödinger equation, as shown by Esposito [*Found. Phys.* **28** (1998) 231]. In MKS units, Maxwell's equations are:

$$\nabla \cdot \mathbf{D} = \rho, \qquad \text{Gauss law for electric field}$$
$$\nabla \cdot \mathbf{B} = 0, \qquad \text{Gauss law for magnetic field}$$
$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}, \qquad \text{Ampere's law} \tag{740}$$
$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t} \qquad \text{Faraday's law}$$

where the units are

$$
\begin{aligned}
\mathbf{E} = \mathbf{F}/q_0 : &\qquad \text{electric field}(V/m) \\
\mathbf{F} : &\qquad \text{electric force}(VC/m) \\
q_0 : &\qquad \text{charge}(C) \\
\mathbf{D} = \epsilon_0 \mathbf{E} + \mathbf{P} = \epsilon \mathbf{E} : &\qquad \text{electric field displacement}(C/m^2) \\
\mathbf{P} = \epsilon_0 (1 + \chi) : &\qquad \text{polarization}(C/m^2) \\
\epsilon_0 = 1/(36\pi) : &\qquad \text{dielectric constant of vacuum}(F/m) \\
\chi : &\qquad \text{electric susceptibility} \\
\epsilon = \epsilon_0 n^2 : &\qquad \text{dielectric constant of the medium}(...) \\
n = \sqrt{\epsilon/\epsilon_0} : &\qquad \text{refractive index} \\
\mathbf{B} = \mu_0 \mathbf{H} + \mathbf{M} : &\qquad \text{magnetic field}(T, \text{or}, Vs/m^2) \\
\mathbf{H} : &\qquad \text{magnetic field intensity}(A/m) \\
\mathbf{M} : &\qquad \text{magnetization}(T, \text{or}, Vs/m^2) \\
\mu_0 = 1/(\epsilon_0 c^2) = 4\pi 10^{-7} : &\qquad \text{permeability of classical vacuum}(H/m) \\
c = 3\ 10^8 : &\qquad \text{speed of light in vacuum}(m/s) \\
\rho : &\qquad \text{charge density}(C/m^3) \\
\mathbf{J} = \sigma \mathbf{E} : &\qquad \text{current density}(A/m^2) \\
\sigma : &\qquad \text{conductivity}(A/(mV))
\end{aligned}
\tag{741}
$$

The wave equation,

$$\nabla^2 \mathbf{E} = \mu_0 \epsilon \frac{\partial^2 \mathbf{E}}{\partial t^2} = \frac{n^2}{c^2} \frac{\partial^2 \mathbf{E}}{\partial t^2}, \tag{742}$$

describes the evolution of the electric field in a medium with $\mathbf{M} = 0$, $\mathbf{J} = 0$ and $\rho = 0$. This equation is obtained from Ampere's law, as follows: $\nabla \times (\nabla \times \mathbf{E}) = -\frac{\partial}{\partial t}(\nabla \times \mathbf{B})$, since $\nabla \times (\nabla \times \mathbf{E}) = \nabla(\nabla \cdot \mathbf{E}) - \nabla^2 \mathbf{E}$, with $\nabla \cdot \mathbf{E} = 0$.

Analogously, we obtain

$$\nabla^2 \mathbf{B} = \frac{n^2}{c^2} \frac{\partial^2 \mathbf{B}}{\partial t^2}, \tag{743}$$

Therefore, the linear combination:

$$\Psi = \frac{1}{\sqrt{2\mu_0}} \left(\frac{n}{c} \mathbf{E} - i\mathbf{B} \right), \tag{744}$$

satisfies the following equation of motion

$$
\begin{aligned}
\frac{\partial \Psi}{\partial t} &= \frac{\partial}{\partial t} \frac{1}{\sqrt{2\mu_0}} \left(\frac{n}{c} \mathbf{E} - i\mathbf{B} \right), \\
&= \frac{1}{\sqrt{2\mu_0}} \left(\frac{n}{c} \frac{\partial \mathbf{E}}{\partial t} - i \frac{\partial \mathbf{B}}{\partial t} \right), \\
&= \frac{1}{\sqrt{2\mu_0}} \left(\frac{c}{n} \nabla \times \mathbf{B} + i\nabla \times \mathbf{E} \right), \\
&= i\frac{c}{n} \nabla \times \frac{1}{\sqrt{2\mu_0}} \left(\frac{n}{c} \mathbf{E} - i\mathbf{B} \right),
\end{aligned}
\tag{745}
$$

since $\partial \mathbf{E}/\partial t = \frac{c^2}{n^2} \nabla \times \mathbf{B}$, and $\partial \mathbf{B}/\partial t = -\nabla \times \mathbf{E}$. Therefore,

$$
\begin{aligned}
i\hbar \frac{\partial \Psi}{\partial t} &= -\frac{\hbar c}{n} \nabla \times \Psi, \\
&= \hat{H}\Psi.
\end{aligned}
\tag{746}
$$

Computing the probability density, we obtain:

$$
\begin{aligned}
\Psi^* \Psi &= \frac{1}{2\mu_0} \left(\frac{n^2}{c^2} \mathbf{E}^2 + \mathbf{B}^2 \right) \\
&= \frac{1}{2} \left(\epsilon \mathbf{E}^2 + \frac{\mathbf{B}^2}{\mu_0} \right).
\end{aligned}
\tag{747}
$$

39   Reacting dipoles in an Electric Field

We consider a model bimolecular reaction of species with dipoles $\vec{\mu}$, where electron-deficient functional groups react with electron-rich groups. The reaction is schematically represented in the following diagram, as influenced by a time-dependent external electromagenetic field $E(t)$:
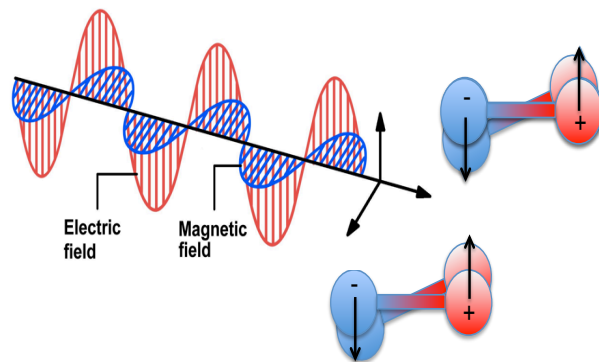
Assuming that the transition state (TS) has half the dipole moment $\tilde{\mu}$ of the reactant species μ, we can compute the speed-up due to the influence of $E(t)$, as given by the ratio of Arrhenius prefactors:

$$f(E) = \frac{A(E)}{A(0)} = e^{(S(TS,E)-S(react,E))/k} e^{-(S(TS,0)-S(react,0))/k}$$

$$f(E) = e^{-\Delta\Delta S(E)/k}$$

(748)

where $\Delta\Delta S = [\Delta S(react, E) - \Delta S(TS, E)]$. We find that $f(E)$ varies, as follows:
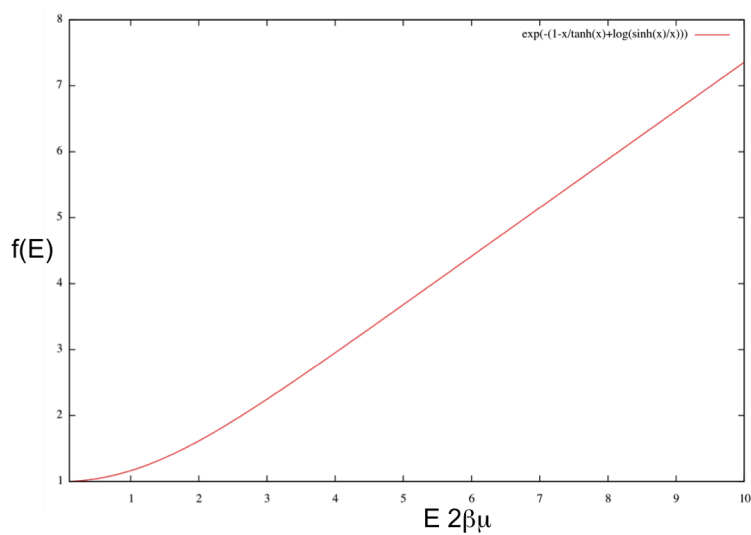
since the change of entropy induced by the field on species with dipole μ depends on $x = 2\beta\mu E(t)$, as follows:

$$\Delta S(react, x) = Nk\left(1 - \frac{x}{2}\coth(x/2) + \ln\left[\frac{\sinh(x/2)}{(x/2)}\right]\right) \tag{749}$$

The rest of this section provides a derivation of Eq. (749). We consider weak dipole moments $\vec{\mu}$ of mass m and moment of inertia I, under room-temperature conditions for which dipole-dipole interactions are much smaller than kT. Under the influence an external Field $\vec{E}(t)$ applied along the z-axis, the system can be described by the following Hamiltonian,

$$H = \sum_{j=1}^{N} H_j \tag{750}$$

with

$$H_j = \frac{p^2}{2m} + \frac{1}{2I}\left(p_\theta^2 + \frac{p_\phi^2}{\sin^2\theta}\right) - E(t)\mu\cos\theta \tag{751}$$

The canonical partition function

$$Z = \prod_{j=1}^{N} Z_j \tag{752}$$

with

$$\begin{aligned}
Z_j(E) &= \int d\phi \int dr \int dp e^{-\frac{\beta p^2}{2m}} \int dp_\theta e^{-\frac{\beta p_\theta^2}{2I}} \int d\theta \int dp_\phi e^{-\frac{\beta p_\phi^2}{\sin^2\theta}} e^{\beta E(t)\mu\cos\theta}, \\
&= 2\pi V[2\pi m\beta^{-1}]^{3/2}[2\pi I\beta^{-1}]^{1/2}\int d\theta[\pi m\beta^{-1}]^{1/2}\sin\theta e^{\beta E(t)\mu\cos\theta} \\
&= 2\pi V[2\pi m\beta^{-1}]^{3/2}[2Im]^{1/2}\pi\beta^{-1}\int_0^\pi d\theta\sin\theta e^{\beta E(t)\mu\cos\theta} \\
&= 2\pi V[2\pi m\beta^{-1}]^{3/2}[2Im]^{1/2}\pi\beta^{-1}\frac{e^{\beta E(t)\mu} - e^{-\beta E(t)\mu}}{\beta E(t)\mu} \\
&= 2\pi^2 V[2\pi m]^{3/2}[2Im]^{1/2}[E(t)\mu]^{-1}\beta^{-7/2}[e^{\beta E(t)\mu} - e^{-\beta E(t)\mu}],
\end{aligned} \tag{753}$$

giving

$$Z_j(E=0) = 4\pi V[2\pi m\beta^{-1}]^{3/2}[2Im]^{1/2}\pi\beta^{-1} \tag{754}$$

and

$$\begin{aligned}
\ln Z_j(E) &= \ln\left(2\pi^2 V[2\pi m]^{3/2}[2Im]^{1/2}[E(t)\mu]^{-1}\right) - \frac{7}{2}\ln\beta + \ln[e^{\beta E(t)\mu} - e^{-\beta E(t)\mu}], \\
&= \ln\left(2\pi^2 V[2\pi m]^{3/2}[2Im]^{1/2}\right) - \frac{5}{2}\ln\beta + \ln[\frac{e^{\beta E(t)\mu} - e^{-\beta E(t)\mu}}{\beta E(t)\mu}], \\
\ln Z_j(0) &= \ln\left(2\pi^2 V[2\pi m]^{3/2}[2Im]^{1/2}\right) - \frac{5}{2}\ln\beta + \ln 2
\end{aligned} \tag{755}$$

171

Therefore,

$$\Delta \ln Z_j = \ln \mathbf{Z}_j(E) - \ln \mathbf{Z}_j(0)$$
$$= \ln\left[\frac{e^{\beta E(t)\mu} - e^{-\beta E(t)\mu}}{2\beta E(t)\mu}\right], \tag{756}$$

which can be used to obtain the entropy change, $\Delta S = S(E) - S(0)$, as follows:

$$\Delta S = \frac{N}{T}\frac{\partial \Delta \ln \mathbf{Z}_j}{\partial(-\beta)} + Nk\Delta \ln \mathbf{Z}_j$$
$$= \frac{N}{T}\left(\frac{x}{2\sinh(x/2)}\left(-E(t)\mu\frac{2\cosh(x/2)}{x} + 2E(t)\mu\frac{2\sinh(x/2)}{x^2}\right)\right) + Nk\ln\left[\frac{2\sinh(x/2)}{x}\right]$$
$$= Nk\left(1 - \frac{x}{2}\coth(x/2) + \ln\left[\frac{\sinh(x/2)}{(x/2)}\right]\right) \tag{757}$$

If the transition state has a dipole moment $\tilde{\mu}$ then

$$\Delta\Delta S = \Delta S(\mu) - \Delta S(\tilde{\mu})$$
$$= Nk\left(\frac{\tilde{x}}{2}\coth(\tilde{x}/2) - \frac{x}{2}\coth(x/2) + \ln\left[\frac{\sinh(x/2)}{\sinh(\tilde{x}/2)}\right] - \ln\left[\frac{x}{\tilde{x}}\right]\right) \tag{758}$$

40   Klein Gordon Model

The relativistic energy of a free particle of mass m and velocity $\mathbf{v}$ is given by Einstein's special theory of relativity, as follows:

$$E = \sqrt{m^2c^2(c^2 + \mathbf{v}^2)},$$
$$= \sqrt{m^2c^4\left(1 + \frac{\mathbf{v}^2}{c^2}\right)}, \tag{759}$$

where c is the speed of light. Note that when $v << c$,

$$E = mc^2\left(1 + \frac{\mathbf{v}^2}{c^2}\right)^{1/2},$$
$$= mc^2\left(1 + \frac{1}{2}\frac{\mathbf{v}^2}{c^2} + \cdots\right), \tag{760}$$
$$\approx mc^2 + \frac{1}{2}m\mathbf{v}^2.$$

Therefore, the rest mass energy is $E = mc^2$.

In the non-relativistic theory, we define the energy relative to the rest mass energy mc^2. In the presence of electric and magnetic fields $\mathbf{E}$ and $\mathbf{B}$, respectively, the non-relativistic Hamiltonian of a particle with mass m, charge q and velocity $\mathbf{v}$ is

$$
\begin{aligned}
H(\mathbf{x}, \mathbf{p}) &= \frac{1}{2}m\mathbf{v}^2 + q\phi(\mathbf{x}), \\
&= \frac{1}{2m}\left(\mathbf{p} - \frac{q}{c}\mathbf{A}(\mathbf{x})\right)^2 + q\phi(\mathbf{x}),
\end{aligned}
\tag{761}
$$

where the vector potential $\mathbf{A}$ and the scalar electrostatic potential ϕ satisfy the following equations:

$$
\begin{aligned}
\mathbf{E}(\mathbf{x}) &\equiv -\nabla\phi - \frac{\partial \mathbf{A}}{\partial t}, \\
\mathbf{B}(\mathbf{x}) &\equiv \nabla \times \mathbf{A}.
\end{aligned}
\tag{762}
$$

Note: The Hamiltonian, introduced by Eq. (761), ensures that the force acting on the particle is the Lorentz force

$$
F(\mathbf{x}) = q\mathbf{E}(\mathbf{x}) + \frac{q}{c}\mathbf{v} \times \mathbf{B}(\mathbf{x}),
\tag{763}
$$

when the coordinates $\mathbf{x}$ and momenta $\mathbf{p}$ are conjugate dynamical variables, as defined by Hamilton's equations:

$$
\begin{aligned}
\dot{x}_j &= \frac{\partial H}{\partial p_j}, \\
\dot{p}_j &= -\frac{\partial H}{\partial x_j}.
\end{aligned}
\tag{764}
$$

To obtain the Lorentz force, we first obtain the velocity, as given by one of Hamilton's equations:

$$
v_j = \frac{dx_j}{dt} = \dot{x}_j = \frac{\partial H}{\partial p_j} = \frac{1}{m}\left(p_j - \frac{q}{c}A_j\right),
\tag{765}
$$

which gives

$$
p_j = mv_j + \frac{q}{c}A_j,
\tag{766}
$$

Therefore,

$$
\begin{aligned}
\frac{dp_j}{dt} = m\frac{dv_j}{dt} + \frac{q}{c}\frac{dA_j}{dt} &= -\frac{\partial H}{\partial x_j}, \\
&= \frac{q}{c}\frac{1}{m}\left(\mathbf{p} - \frac{q}{c}\mathbf{A}\right)\frac{\partial \mathbf{A}}{\partial x_j} - q\frac{\partial \phi}{\partial x_j}
\end{aligned}
\tag{767}
$$

and

$$
F_j \equiv m\frac{dv_j}{dt} = \frac{q}{c}\frac{1}{m}\left(\mathbf{p} - \frac{q}{c}\mathbf{A}\right)\frac{\partial \mathbf{A}}{\partial x_j} - q\frac{\partial \phi}{\partial x_j} - \frac{q}{c}\frac{dA_j}{dt},
\tag{768}
$$

with

$$\frac{dA_j}{dt} = \frac{\partial A_j}{\partial t} + \sum_k \frac{\partial x_k}{\partial t}\frac{\partial A_j}{\partial x_k},$$
$$= \frac{\partial A_j}{\partial t} + (\mathbf{v}\cdot\nabla)A_j,$$

(769)

Substituting Eq. (769) into Eq. (768), we obtain:

$$
\begin{aligned}
F_j &= \frac{q}{c}\frac{1}{m}\left(\mathbf{p} - \frac{q}{c}\mathbf{A}\right)\cdot\frac{\partial\mathbf{A}}{\partial x_j} - q\frac{\partial\phi}{\partial x_j} - \frac{q}{c}\frac{\partial A_j}{\partial t} - \frac{q}{c}(\mathbf{v}\cdot\nabla)A_j \\
&= \frac{q}{c}\frac{1}{m}\left(\mathbf{p} - \frac{q}{c}\mathbf{A}\right)\cdot\frac{\partial\mathbf{A}}{\partial x_j} + q\left(-\frac{\partial\phi}{\partial x_j} - \frac{1}{c}\frac{\partial A_j}{\partial t}\right) - \frac{q}{c}(\mathbf{v}\cdot\nabla)A_j \\
&= \frac{q}{c}\mathbf{v}\cdot\frac{\partial\mathbf{A}}{\partial x_j} + qE_j - \frac{q}{c}(\mathbf{v}\cdot\nabla)A_j \\
&= qE_j - \frac{q}{c}\left((\mathbf{v}\cdot\nabla)A_j - \mathbf{v}\cdot\frac{\partial\mathbf{A}}{\partial x_j}\right) \\
&= qE_j - \frac{q}{c}\left((\mathbf{v}\cdot\nabla)A_j - \mathbf{v}\cdot\frac{\partial\mathbf{A}}{\partial x_j}\right) \\
&= qE_j - \frac{q}{c}\left((\sum_k v_k\frac{\partial}{\partial x_k})A_j - \sum_k v_k\frac{\partial A_k}{\partial x_j}\right) \\
&= qE_j - \frac{q}{c}\sum_k v_k\left(\frac{\partial A_j}{\partial x_k} - \frac{\partial A_k}{\partial x_j}\right) \\
&= qE_j - \frac{q}{c}\sum_{k\neq j} v_k\left(\frac{\partial A_j}{\partial x_k} - \frac{\partial A_k}{\partial x_j}\right) \\
&= qE_j + \frac{q}{c}(\mathbf{v}\times\nabla\times\mathbf{A})_j \\
&= qE_j + \frac{q}{c}(\mathbf{v}\times\mathbf{B})_j
\end{aligned}
$$

(770)

The relativistic energy describing a charged particle of mass m, charge q and velocity $\mathbf{v}$, exposed to electric and magnetic fields $\mathbf{E}(\mathbf{x}) = -\nabla\phi - \frac{\partial\mathbf{A}}{\partial t}$ and $\mathbf{B}(\mathbf{x}) = \nabla\times\mathbf{A}$ is

$$
\begin{aligned}
E + q\phi(\mathbf{x}) &= \sqrt{m^2 c^2(c^2 + \mathbf{v}^2)}, \\
&= \sqrt{m^2 c^4 + c^2\left(\mathbf{p} - \frac{q}{c}\mathbf{A}(\mathbf{x})\right)^2},
\end{aligned}
$$

(771)

To formulate a Schrödinger-like equation, we square Eq. (771) as follows:

$$
\begin{aligned}
(E + q\phi(\mathbf{x}))^2 &= m^2 c^2(c^2 + \mathbf{v}^2), \\
\left(i\hbar\frac{\partial}{\partial t} + q\phi(\mathbf{x})\right)^2\Psi &= m^2 c^4\Psi + c^2\left(\mathbf{p} - \frac{q}{c}\mathbf{A}(\mathbf{x})\right)^2\Psi,
\end{aligned}
$$

(772)

174

to obtain the Klein-Gordon equation, neglecting motion orthogonal to the field polarization:

$$-\frac{\hbar^2}{c^2}\frac{d^2\psi}{dt^2} - (p_3 - \frac{e}{c}A_3(t))^2\psi = m^2c^2\psi, \tag{773}$$

we divide both sides by m^2c^2 and introduce the substitutions $x = \frac{mc^2}{\hbar}t$:

$$-\frac{d^2\psi}{dx^2} - \left(\frac{p_3}{mc} - \frac{e}{mc^2}A_3(x)\right)^2\psi = \psi. \tag{774}$$

As an example, we consider $A_3/c = \frac{\epsilon/\omega}{1+(\omega t)^2}$, where $\epsilon = 0.2\epsilon_c$, with the 'Schwinger critical field' $\epsilon_c = \frac{m^2c^3}{e\hbar} \approx 10^{16}V$/cm and $\omega = 0.1\frac{mc^2}{\hbar}$. Therefore,

$$-\frac{d^2\psi}{dx^2} - \left(\frac{p_3}{mc} - \frac{e}{mc}\frac{0.2\frac{m^2c^3}{e\hbar}/0.1\frac{mc^2}{\hbar}}{1+(\omega t)^2}\right)^2\psi = \psi, \tag{775}$$

or

$$-\frac{d^2\psi}{dx^2} - \left(\frac{p_3}{mc} - \frac{2}{1+0.01x^2}\right)^2\psi = \psi, \tag{776}$$

Introducing the variable $M = 0.5$ and working in atomic units ($\hbar = 1$), we can formulate Eq. (776), as follows:

$$-\frac{\hbar^2}{2M}\frac{d^2\psi}{dx^2} + V(x)\psi = \psi, \tag{777}$$

with $V(x) = -\left(\frac{1.7}{c} - \frac{2}{1+0.01x^2}\right)^2$ in atomic units, where we introduced $1.7 = p_3/m$. Note that $c = 137$ in atomic units.

40.1 The Dirac Equation

To obtain the Dirac equation for a free particle, we square Eq. (771) with $q = 0$ and $c = 1$, as follows:

$$\begin{aligned}E^2 &= m^2c^2(c^2 + \mathbf{v}^2), \\ &= m^2 + \mathbf{p}^2.\end{aligned} \tag{778}$$

We assume that E can be expressed as

$$E = \beta m + \alpha \cdot \mathbf{p}, \tag{779}$$

which is the so-called *Dirac equation*. We look for α and β such that E defined according to Eq. (779) satisfies Eq. (778). In the most general case,

$$\begin{aligned}E^2 &= (\beta m + \alpha^i p_i)(\beta m + \alpha^j p_j), \\ &= \beta^2 m^2 + (\beta\alpha^j + \alpha^j\beta)p_j m + \alpha^i\alpha^j p_i p_j, \\ &= \beta^2 m^2 + (\beta\alpha^j + \alpha^j\beta)p_j m + \frac{1}{2}(\alpha^i\alpha^j + \alpha^j\alpha^i)p_i p_j.\end{aligned} \tag{780}$$

175

We note that according to Eqs. (780) α and β cannot be defined scalars since, according to Eq. (778), $\beta^2 = 1$ and $\beta\alpha^j + \alpha^j\beta = 0$ which could only be satisfied by $\alpha^j = 0$. We show below that even defining α and β as 2×2 (or 3×3) matrices, would not allow the factorization of Eq. (779) to satisfy Eq. (778). The smaller matrices that fulfill the conditions are 4×4 and satisfy the following equations:

$$\beta^2 = I,$$
$$\beta\alpha^j + \alpha^j\beta = 0, \tag{781}$$
$$\alpha^i\alpha^j + \alpha^j\alpha^i = 2\delta^{ij}.$$

Introducing the matrices $\gamma^j = \beta\alpha^j$ and $\gamma^0 = \beta$, we obtain:

$$\alpha^i\beta\beta\alpha^j + \alpha^j\beta\beta\alpha^i = 2\delta^{ij},$$
$$-\beta\alpha^i\beta\alpha^j - \beta\alpha^j\beta\alpha^i = 2\delta^{ij}, \tag{782}$$
$$-\gamma^i\gamma^j - \gamma^j\gamma^i = 2\delta^{ij}.$$

and

$$\beta\beta\alpha^j + \beta\alpha^j\beta = 0,$$
$$\gamma^0\gamma^j + \gamma^j\gamma^0 = 0. \tag{783}$$

Therefore, the γ matrices satisfy the anticommutation relations introduced by Eqs. (782) and (783) that can be summarized, as follows:

$$[\gamma^\mu, \gamma^\nu]_+ = 2g^{\mu\nu},$$

with $g = \text{diag}(1, -1, -1, -1)$ (*i.e.,* a diagonal 4×4 matrix with those elements).

In addition, we note that according to Eq. (781),

$$\beta\alpha^j + \alpha^j\beta = 0,$$
$$\beta\alpha^j\beta = -\alpha^j\beta^2,$$
$$\beta\alpha^j\beta = -\alpha^j,$$
$$(\beta\alpha^j\beta)(\beta\alpha^j\beta) = (\alpha^j)^2,$$
$$\beta\alpha^j\alpha^j\beta = (\alpha^j)^2, \tag{784}$$
$$-\gamma^j\gamma^j = (\alpha^j)^2,$$
$$-\frac{1}{2}(\gamma^j\gamma^j + \gamma^j\gamma^j) = (\alpha^j)^2,$$
$$I = (\alpha^j)^2.$$

Therefore, $(\alpha^j)^{-1} = \alpha^j$ and $\beta = \beta^{-1}$. Furthermore, $\text{Tr}[\beta\alpha^j\beta] = Tr[\alpha^j] = -Tr[\alpha^j]$ which implies that $Tr[\alpha^j] = 0$. Analogously,

$$\beta\alpha^j = -\alpha^j\beta,$$
$$\alpha^j\beta\alpha^j = -\beta, \tag{785}$$

thus, $Tr[\beta] = 0$.

We show that the eigenvalues of β (or α) are $\lambda = \pm 1$, as follows:

$$\begin{aligned}
\beta v &= \lambda v, \\
v &= \lambda \beta v, \\
v &= \lambda^2 v.
\end{aligned} \tag{786}$$

To show that the matrices α and β cannot be defined as 2×2 matrices since they would not fulfill the anticommutation conditions, we first note that E and $\mathbf{p}$ are Hermitian, so α and β must be Hermitian. The most general form of a 2×2 Hermitian matrix is

$$\begin{bmatrix} A & Re[B] - iIm[B] \\ Re[B] + iIm[B] & C \end{bmatrix} = \tfrac{1}{2}A(\sigma_0 + \sigma_3) + \tfrac{1}{2}C(\sigma_0 - \sigma_3) + Re[B]\sigma_1 + Im[B]\sigma_2.$$

where the identity matrix $\sigma_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$,

and the Pauli matrices $\sigma_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, $\sigma_2 = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$, and $\sigma_3 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$,

are linearly independent (*i.e.,* $\sum_{j=0}^{3} c_j \sigma_j = 0$ only when $c_j = 0$ for all $j = 0, \cdots 3$). Therefore, if we define α^j in terms of the Pauli matrices (*i.e.,* $\alpha^j = \sigma_j$, for $j = 1, \cdots 3$, or as linear combinations of Pauli matrices), then β would have to be defined as the identity matrix which would not anticommute with α^j, as required by Eq. (781). Therefore, α and β cannot be 2×2 matrices.

We consider defining α and β as 4×4 matrices, in the so-called *standard representation* with

$$\beta = \begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix}, \text{ and } \alpha^j = \begin{bmatrix} A & B \\ B^\dagger & C \end{bmatrix},$$

where α^j are Hermitian matrices that satisfy the commutation relations $\beta \alpha^j + \alpha^j \beta = 0$,

$$\begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix} \begin{bmatrix} A & B \\ B^\dagger & C \end{bmatrix} + \begin{bmatrix} A & B \\ B^\dagger & C \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix} = \begin{bmatrix} 2A & 0 \\ 0 & -2C \end{bmatrix} = 0,$$

which implies $A = C = 0$. Choosing

$$\alpha^j = \begin{bmatrix} 0 & \sigma_j \\ \sigma_j & 0 \end{bmatrix}, \text{ gives } \gamma^j = \begin{bmatrix} 0 & \sigma_j \\ -\sigma_j & 0 \end{bmatrix}, \text{ and } \gamma^0 = \begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix}.$$

Introducing the substitutions $E = i\partial_0$ and $p^j = -i\partial_j$ into Eq. (779), we obtain:

$$\begin{aligned} i\partial_0 &= -i\alpha^j\partial_j + \beta m, \\ i\gamma^0\partial_0 &= -i\gamma^j\partial_j + m, \\ i\gamma^0\partial_0 + i\gamma^j\partial_j + m &= 0, \\ i\slashed{\partial} + m &= 0. \end{aligned} \tag{787}$$

where we have introduced the so-called 'Feynman slash' notation $\slashed{\partial} = \gamma^\mu\partial_\mu$ for the contraction of the matrices γ^μ and $\partial_\mu = (\partial_0, \partial_1, \partial_2, \partial_3)$. Using this notation, we can write the Dirac equation, as follows:

$$(i\slashed{\partial} + m)\psi(x, t) = 0, \tag{788}$$

where

$$\psi(x, t) = u(p)e^{-ip^\mu x_\mu}, \tag{789}$$

is a 4-component vector, the so-called *Dirac spinor*. It can be shown that the 4 components describe free particles with both positive and negative energy solutions, each with spin $1/2$ either up or down.

41 Fock Dynamics

The goal of this section is to introduce a simple method for propagating the time-dependent wave-function $\Psi(\mathbf{r}; t)$ of a system of n spinless electrons of mass m_e, charge e and coordinates $\mathbf{r} = \mathbf{r}_1, \cdots \mathbf{r}_n$, interacting among themselves and with nuclei with coordinates $\mathbf{R}_1, \cdots \mathbf{R}_N$ and charges $\mathbf{Z}_1, \cdots \mathbf{Z}_N$. Under the Born-Openheimer approximation, the nuclei are assumed to be fixed. Therefore, the Hamiltonian of the system is

$$H(\mathbf{r}; t) = -\frac{\hbar^2}{2m_e}\sum_{j=1}^{n}\nabla_j^2 - \sum_{j=1}^{n}\sum_{J=1}^{N}\frac{eZ_J}{4\pi\epsilon_0|\mathbf{r}_j - \mathbf{R}_J|} + \frac{1}{2}\sum_{j=1}^{n}\sum_{k\neq j}\frac{e^2}{4\pi\epsilon_0|\mathbf{r}_j - \mathbf{r}_k|}. \tag{790}$$

The n-electron wavefunction is approximated as a Fock-product of one-electron wavefunctions,

$$\Psi(\mathbf{r}; t) = \prod_{j=1}^{n}\psi_j(\mathbf{r}_j; t), \tag{791}$$

where the components $\psi_j(\mathbf{r}_j; t)$ are expanded, as follows:

$$\psi_j(\mathbf{r}_j; t) = \sum_k c_k^{(j)}(t)\phi_k^{(j)}(\mathbf{r}_j; t), \tag{792}$$

with $c_k^{(j)}(t) = \langle\phi_k^{(j)}(t)|\psi_j(t)\rangle$. We choose $\phi_k^{(j)}(t)$ to be the solutions of the eigenvalue problem,

$$h(\mathbf{r}_j; t)\phi_k^{(j)}(\mathbf{r}_j; t) = E_k^{(j)}(t)\phi_k^{(j)}(\mathbf{r}_j; t), \tag{793}$$

with

$$h(\mathbf{r}_j; t) = -\frac{1}{2}\nabla_j^2 - \sum_{J=1}^{N}\frac{Z_J}{|\mathbf{r}_j - \mathbf{R}_J|} + \frac{1}{2}\sum_{l\neq j}\int d\mathbf{r}_l\frac{\psi_l(\mathbf{r}_l; t)^*\psi_l(\mathbf{r}_l; t)}{|\mathbf{r}_j - \mathbf{r}_l|}, \tag{794}$$

where we have used atomic units, with $e = m_e = \hbar = 4\pi\epsilon_0 = 1$. Note that the Fock energy

$$E^F = \langle\Psi|H|\Psi\rangle,$$
$$= \sum_{j=1}^{n}\langle\psi_j|h|\psi_j\rangle. \tag{795}$$

Therefore,

$$\Psi(\mathbf{r}; t + \tau) = \prod_{j=1}^{n}\psi_j(\mathbf{r}_j; t + \tau), \tag{796}$$

where

$$\psi_j(\mathbf{r}_j; t + \tau) \approx \sum_k c_k^{(j)}(t + \tau)\phi_k^{(j)}(\mathbf{r}_j; t), \tag{797}$$

with $c_k^{(j)}(t + \tau) = c_k^{(j)}(t)e^{-\frac{i}{\hbar}E_k^{(j)}(t)\tau}$. Further propagation can be performed analogously by re-expanding $\psi_j(\mathbf{r}_j; t + \tau)$ according to Eq. (792), after resetting the time $t + \tau$ to t, and evolving the state for another time-increment τ according to Eq. (797).

41.1 Electron-Hole Pair Dynamics

The correlated dynamics of electrons and holes can also be simulated approximately by using the Fock dynamics methodology introduced in this section. The simplest possible example is the dynamics of one electron and one hole (a missing a electron) described by a time-dependent Fock product,

$$\Psi(\mathbf{r}_e, \mathbf{r}_h; t) = \psi_e(\mathbf{r}_e; t)\psi_h(\mathbf{r}_h; t), \tag{798}$$

where $\psi_e(\mathbf{r}_j; t)$ is expanded, as follows:

$$\psi_e(\mathbf{r}_e; t) = \sum_k c_k^{(e)}(t)\phi_k^{(e)}(\mathbf{r}_e; t), \tag{799}$$

with $c_k^{(e)}(t) = \langle \phi_k^{(e)}(t)|\psi_e(t)\rangle$. Here, $\phi_k^{(e)}(t)$ are solutions of the eigenvalue problem,

$$h_e(\mathbf{r}_e; t)\phi_k^{(e)}(\mathbf{r}_e; t) = E_k^{(e)}(t)\phi_k^{(e)}(\mathbf{r}_e; t), \tag{800}$$

where the electronic Hamiltonian,

$$h_e(\mathbf{r}; t) = -\frac{\hbar^2}{2}\nabla_e^2 - \sum_{J=1}^{N}\frac{Z_J}{4\pi\epsilon_0|\mathbf{r}_e - \mathbf{R}_J|} - \frac{1}{2}\int d\mathbf{r}_h \frac{\psi_h(\mathbf{r}_h; t)^*\psi_h(\mathbf{r}_h; t)}{4\pi\epsilon|\mathbf{r}_e - \mathbf{r}_h|}, \tag{801}$$

includes the electron-hole pair interaction screened by an effective dielectric constant ϵ, due to the other valence electrons. Analogously,

$$\psi_h(\mathbf{r}_h; t) = \sum_k c_k^{(h)}(t)\phi_k^{(h)}(\mathbf{r}_h; t), \tag{802}$$

with $c_k^{(h)}(t) = \langle \phi_k^{(h)}(t)|\psi_h(t)\rangle$. Here, $\phi_k^{(h)}(t)$ are solutions of the eigenvalue problem,

$$h_h(\mathbf{r}_h; t)\phi_k^{(h)}(\mathbf{r}_h; t) = E_k^{(h)}(t)\phi_k^{(h)}(\mathbf{r}_h; t), \tag{803}$$

with

$$h_h(\mathbf{r}; t) = -\frac{\hbar^2}{2}\nabla_h^2 - \sum_{J=1}^{N}\frac{Z_J}{4\pi\epsilon_0|\mathbf{r}_h - \mathbf{R}_J|} - \frac{1}{2}\int d\mathbf{r}_h \frac{\psi_e(\mathbf{r}_e; t)^*\psi_e(\mathbf{r}_e; t)}{4\pi\epsilon|\mathbf{r}_e - \mathbf{r}_h|}. \tag{804}$$

42 Hartree-Fock Dynamics

The propagation scheme introduced in Sec. 41 can be generalized to the propagation of a Hartree-Fock product,

$$\begin{aligned}\Psi(\mathbf{r}; t) &= \hat{A}\prod_{j=1}^{n}\tilde{\chi}_j(\mathbf{x}_j; t), \\ &= |\tilde{\chi}_1(\mathbf{x}_1; t)\tilde{\chi}_2(\mathbf{x}_2; t)\cdots\tilde{\chi}_n(\mathbf{x}_n; t)|,\end{aligned} \tag{805}$$

where $\hat{A}$ is the antisymmetrizing operator and the $|\cdots|$ represents a Slater determinant. The spin-orbitals $\chi_j(\mathbf{x})$ are products of one-electron spatial orbitals $\psi_j(\mathbf{r})$ and either $\alpha(\omega)$ or $\beta(\omega)$ spin-functions, with space-spin coordinates $\mathbf{x} = (\mathbf{r}, \omega)$. The spin-orbitals $\tilde{\chi}_j(\mathbf{x})$ are expanded, as follows:

$$\tilde{\chi}_j(\mathbf{x}_j; t) = \sum_k c_k^{(j)}(t) \chi_k(\mathbf{x}_j; t), \tag{806}$$

with $c_k^{(j)}(t) = \langle \chi_k(t) | \tilde{\chi}_j(t) \rangle$. We choose $\chi_k(\mathbf{x}_j; t)$ to be the solutions of the eigenvalue problem,

$$h(\mathbf{r}_j; t) \chi_k(\mathbf{x}_j; t) - \frac{1}{2} \sum_{l \neq j} \int d\mathbf{x}_l \frac{\chi_l^*(\mathbf{x}_l) \chi_j(\mathbf{x}_l)}{|\mathbf{r}_j - \mathbf{r}_l|} \chi_l(\mathbf{x}_j; t) = E_k(t) \chi_k(\mathbf{x}_j; t), \tag{807}$$

where $h(\mathbf{r}_j; t)$ is defined according to Eq. (794), giving the Hartree-Fock energy:

$$
\begin{aligned}
E_{HF} &= \langle \Psi | H | \Psi \rangle, \\
&= \sum_{j=1}^{n} \langle \chi_j | h | \chi_j \rangle - \frac{1}{2} \sum_{j=1}^{n} \sum_{l \neq j} \int d\mathbf{x}_j \int d\mathbf{x}_l \frac{\chi_l^*(\mathbf{x}_l) \chi_j(\mathbf{x}_l)}{|\mathbf{r}_j - \mathbf{r}_l|} \chi_j^*(\mathbf{x}_j; t) \chi_l(\mathbf{x}_j; t).
\end{aligned}
\tag{808}
$$

Analgously to Sec. 41, we approximate the dynamics as follows:

$$
\begin{aligned}
\Psi(\mathbf{r}; t + \tau) &= \hat{A} \prod_{j=1}^{n} \tilde{\chi}_j(\mathbf{x}_j; t + \tau), \\
&= |\tilde{\chi}_1(\mathbf{x}_1; t + \tau) \tilde{\chi}_2(\mathbf{x}_2; t + \tau) \cdots \tilde{\chi}_n(\mathbf{x}_n; t + \tau)|,
\end{aligned}
\tag{809}
$$

with

$$\tilde{\chi}_j(\mathbf{x}_j; t + \tau) \approx \sum_k c_k^{(j)}(t + \tau) \chi_k(\mathbf{x}_j; t), \tag{810}$$

where $c_k^{(j)}(t + \tau) = c_k^{(j)}(t) e^{-\frac{i}{\hbar} E_k(t) \tau}$. Further propagation can be performed analogously by re-expanding $\tilde{\chi}_j(\mathbf{x}_j; t + \tau)$ according to Eq. (806), after resetting the time $t + \tau$ to t, solving Eq. (807), and evolving the state for another time-increment τ according to Eq. (810).

43 Self-Consistent Field Hartree-Fock Method

The self-consistent field (SCF) Hartree-Fock (HF) method is a variational approach for finding the Slater determinant of a system of n-electrons,

$$|\Phi\rangle = |\chi_1 \chi_2 \cdots \chi_n\rangle \tag{811}$$

that minimizes the expectation value of the energy:

$$\bar{E} = \frac{\langle \Phi | \hat{H} | \Phi \rangle}{\langle \Phi | \Phi \rangle},$$

for a fixed nuclear configuration.

The one-electron basis functions χ_i are typically expressed as linear combinations of spin orbitals ϕ_k, as follows:

$$\chi_i = \sum_k c_{ki}\phi_k, \tag{812}$$

allowing for the variational approach to minimize $\bar{E}$ with respect to the expansion coefficients c_{ki}.

The energy is computed according to the usual Hamiltonian of a system of N nuclei and n electrons, with a fixed nuclear configuration:

$$\hat{H}_{el} = \sum_{i=1}^{n} \hat{h}^{(i)} + \sum_{i}\sum_{k>i}^{n} \frac{e^2}{r_{ik}}, \tag{813}$$

where the spin-orbit coupling interactions are neglected. The first term in Eq. (813) is the sum of 1-electron *core Hamiltonians*,

$$\hat{h}^{(i)} = -\frac{\hbar^2}{2m_e}\nabla_{r_i}^2 - \sum_{j=1}^{N} \frac{z_j e^2}{r_{ji}}, \tag{814}$$

describing a system of n non-interacting electrons in the electrostatic potential of the nuclei. The second term in Eq. (813) is the sum of electron-electron interactions.

As a simple example, we consider the H_2 molecule with $n = 2$, $N = 2$ and $\langle \mathbf{r}_1, \mathbf{r}_2 | \Phi \rangle = 2^{-1/2}\left(\langle\mathbf{r}_1|\chi_1\rangle\langle\mathbf{r}_2|\chi_2\rangle - \langle\mathbf{r}_1|\chi_2\rangle\langle\mathbf{r}_2|\chi_1\rangle\right)$,

$$\begin{aligned}\langle \mathbf{r}_1, \mathbf{r}_2 | \hat{H}_{el} | \Phi \rangle &= 2^{-1/2}\Big[\langle\mathbf{r}_2|\chi_2\rangle\langle\mathbf{r}_1|\hat{h}^{(1)}|\chi_1\rangle - \langle\mathbf{r}_2|\chi_1\rangle\langle\mathbf{r}_1|\hat{h}^{(1)}|\chi_2\rangle + \langle\mathbf{r}_1|\chi_1\rangle\langle\mathbf{r}_2|\hat{h}^{(2)}|\chi_2\rangle \\ &\quad -\langle\mathbf{r}_1|\chi_2\rangle\langle\mathbf{r}_2|\hat{h}^{(2)}|\chi_1\rangle + \tfrac{e^2}{r_{12}}[\langle\mathbf{r}_1|\chi_1\rangle\langle\mathbf{r}_2|\chi_2\rangle - \langle\mathbf{r}_1|\chi_2\rangle\langle\mathbf{r}_2|\chi_1\rangle]\Big]\end{aligned} \tag{815}$$

and the energy expectation value

$$\begin{aligned}\bar{E} &= \langle\Phi|\hat{H}_{el}|\Phi\rangle \\ &= \tfrac{1}{2}\Big[\langle\chi_1|\hat{h}^{(1)}|\chi_1\rangle + \langle\chi_2|\hat{h}^{(1)}|\chi_2\rangle + \langle\chi_2|\hat{h}^{(1)}|\chi_2\rangle + \langle\chi_1|\hat{h}^{(1)}|\chi_1\rangle+ \\ &\quad +[\langle\chi_1\chi_2| - \langle\chi_2\chi_1|]\tfrac{e^2}{r_{12}}[|\chi_1\chi_2\rangle - |\chi_2\chi_1\rangle)]\Big] \\ &= \langle\chi_1|\hat{h}^{(1)}|\chi_1\rangle + \langle\chi_2|\hat{h}^{(1)}|\chi_2\rangle + \tfrac{1}{2}[\langle\chi_1\chi_2|\tfrac{e^2}{r_{12}}|\chi_1\chi_2\rangle \\ &\quad -\langle\chi_1\chi_2|\tfrac{e^2}{r_{12}}|\chi_2\chi_1\rangle - \langle\chi_2\chi_1|\tfrac{e^2}{r_{12}}|\chi_1\chi_2\rangle + \langle\chi_2\chi_1|\tfrac{e^2}{r_{12}}|\chi_2\chi_1\rangle]\end{aligned} \tag{816}$$

since $\hat{h}^{(1)} = \hat{h}^{(2)}$, according to Eq. (814). In general,

$$\bar{E} = \sum_{j=1}^{n}\langle\chi_j|\hat{h}^{(1)}|\chi_j\rangle + \frac{1}{2}\sum_{j=1}^{n}\sum_{k\neq j}\langle\chi_j\chi_k|\frac{e^2}{r_{12}}|\chi_j\chi_k\rangle - \langle\chi_j\chi_k|\frac{e^2}{r_{12}}|\chi_k\chi_j\rangle, \tag{817}$$

or

$$\bar{E} = \sum_{j=1}^{n}\langle\chi_j|\hat{h}^{(1)}|\chi_j\rangle + \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n}\langle\chi_j\chi_k|\frac{e^2}{r_{12}}|\chi_j\chi_k\rangle - \langle\chi_j\chi_k|\frac{e^2}{r_{12}}|\chi_k\chi_j\rangle, \tag{818}$$

182

since $\langle \chi_j \chi_k | \frac{e^2}{r_{12}} | \chi_j \chi_k \rangle - \langle \chi_j \chi_k | \frac{e^2}{r_{12}} | \chi_k \chi_j \rangle = 0$, when $j = k$.

To minimize $\bar{E}$ with respect to χ_j, subject to the constraint of orthonormal orbitals, we apply the Lagrange multiplier method for the following functional:

$$L(\chi_1, ..., \chi_n) = \bar{E} + \sum_j \sum_k \epsilon_{jk}[\langle \chi_j | \chi_k \rangle - \delta_{jk}], \tag{819}$$

where ϵ_{jk} are the Lagrange multipliers. Varying the spin orbitals χ_j in an infinitesimal amount δ_j, with respect to the expansion coefficients c_{kj}, we obtain:

$$\delta L(\chi_1, ..., \chi_n) = \delta \bar{E} + \sum_{j=1}^{n} \sum_{k=1}^{n} \epsilon_{jk}[\langle \delta \chi_j | \chi_k \rangle + \langle \chi_j | \delta \chi_k \rangle], \tag{820}$$

where

$$\begin{aligned}
\delta \bar{E} &= \sum_{j=1}^{n} \langle \delta \chi_j | \hat{h}^{(1)} | \chi_j \rangle + \langle \chi_j | \hat{h}^{(1)} | \delta \chi_j \rangle \\
&+ \frac{1}{2} \sum_{j=1}^{n} \sum_{k=1}^{n} \langle \delta \chi_j \chi_k | \frac{e^2}{r_{12}} | \chi_j \chi_k \rangle + \langle \chi_j \delta \chi_k | \frac{e^2}{r_{12}} | \chi_j \chi_k \rangle \\
&+ \langle \chi_j \chi_k | \frac{e^2}{r_{12}} | \delta \chi_j \chi_k \rangle + \langle \chi_j \chi_k | \frac{e^2}{r_{12}} | \chi_j \delta \chi_k \rangle \\
&- \langle \delta \chi_j \chi_k | \frac{e^2}{r_{12}} | \chi_k \chi_j \rangle - \langle \chi_j \delta \chi_k | \frac{e^2}{r_{12}} | \chi_k \chi_j \rangle \\
&- \langle \chi_j \chi_k | \frac{e^2}{r_{12}} | \delta \chi_k \chi_j \rangle - \langle \chi_j \chi_k | \frac{e^2}{r_{12}} | \chi_k \delta \chi_j \rangle
\end{aligned} \tag{821}$$

Substituting Eq. (821) into Eq. (820) and simplifying, we obtain:

$$\begin{aligned}
\delta L &= \sum_{j=1}^{n} \langle \delta \chi_j | \hat{h}^{(1)} | \chi_j \rangle + \sum_{j=1}^{n} \sum_{k=1}^{n} \langle \delta \chi_j \chi_k | \frac{e^2}{r_{12}} | \chi_j \chi_k \rangle - \langle \delta \chi_j \chi_k | \frac{e^2}{r_{12}} | \chi_k \chi_j \rangle \\
&+ \sum_{j=1}^{n} \sum_{k=1}^{n} \epsilon_{jk} \langle \delta \chi_j | \chi_k \rangle + c.c.
\end{aligned} \tag{822}$$

which gives,

$$\begin{aligned}
\delta L &= \sum_{j=1}^{n} \langle \delta \chi_j(1) | \left[\hat{h}^{(1)} | \chi_j(1) \rangle + \sum_{k=1}^{n} \langle \chi_k(2) | \frac{e^2}{r_{12}} | \chi_k(2) \rangle | \chi_j(1) \rangle \right. \\
&\left. - \langle \chi_k(2) | \frac{e^2}{r_{12}} | \chi_j(2) \rangle | \chi_k(1) \rangle + \sum_{j=1}^{n} \sum_{k=1}^{n} \epsilon_{jk} | \chi_k \rangle \right] + c.c. = 0
\end{aligned} \tag{823}$$

Since $\delta \chi_j$ is arbitrary, it must be that the expression in square brackets is equal to zero for all j:

$$\left[\hat{h}^{(1)} + \sum_{k=1}^{n} \langle \chi_k(2) | \frac{e^2}{r_{12}} [1 - \hat{P}_{12}] | \chi_k(2) \rangle \right] | \chi_j(1) \rangle = \sum_{j=1}^{n} \sum_{k=1}^{n} \epsilon_{jk} | \chi_k \rangle \tag{824}$$

where the operator $\hat{P}_{12}$ permutes the states of electrons 1 and 2.

To write Eq. (824) in the canonical eigenvalue form, we change the basis set according to the unitary transformation,

$$|\chi_j'\rangle = \sum_k |\chi_k\rangle \Gamma_{kj}, \tag{825}$$

with $\boldsymbol{\Gamma}\boldsymbol{\Gamma}^\dagger = 1$. Considering that the Lagrange multipliers matrix ϵ is Hermitian (since the functional L is real), it is always possible to find a $\boldsymbol{\Gamma}$ that diagonalizes ϵ according to the similarity transformation:

$$\epsilon' = \boldsymbol{\Gamma}^\dagger \epsilon \boldsymbol{\Gamma} \tag{826}$$

183

Such a transformation defines the set of *canonical spin orbitals* $|\chi'_j\rangle$ for which

$$\hat{f}^{(1)}|\chi'_j\rangle = \epsilon'_{jj}|\chi'_j\rangle. \tag{827}$$

for $j = 1 - n$, where $\hat{f}^{(1)}$ is the *Fock* operator,

$$\hat{f}^{(1)} = \hat{h}^{(1)} + \hat{V}_1^{HF}, \tag{828}$$

where $\hat{V}_1^{HF}$ is the *Hartree-Fock potential* describing the electron-electron interactions, as follows:

$$\hat{V}_1^{HF} = J_1(r_1) - \hat{X}_1(r_1), \tag{829}$$

where $J_1(r_1)$ is the Coulomb mean-field potential,

$$J_1(r_1) = \sum_{k=1}^{n} \langle \chi'_k | \frac{e^2}{r_{12}} | \chi'_k \rangle. \tag{830}$$

The matrix elements in Eq. (830) are integrals over the spatial and spin coordinates of electron 2. Analogously, $\hat{X}_1(r_1)$ is defined as the *exchange operator*,

$$\hat{X}_1(r_1) = \sum_{k\neq j}^{n} \langle \chi'_k(2) | \frac{e^2}{r_{12}} \hat{P}_{12} | \chi'_k(2) \rangle. \tag{831}$$

Equation (827) defines a self-consistent field (SCF) problem since the operator $\hat{f}^{(1)}$, required to find the solutions χ'_j, depends on those functions through J_1 and $\hat{X}_1$. To solve this SCF problem, we first obtain approximate solutions χ'_j by approximating $\hat{f}^{(1)}$ by $\hat{h}^{(1)}$ (i.e., neglecting the electron-electron interactions introduced by J_1 and $\hat{X}_1$), or by using a semiempirical method (like the Hueckel method described later in these lectures). These approximate functions χ'_j are then used to compute J_1 and $\hat{X}_1$, giving an approximate $\hat{f}^{(1)}$ that can be used to obtain improved functions χ'_j. The process is repeated until convergence.

To solve the Hartree-Fock Eq. (827) by solving a set of matrix equations, we substitute χ'_j by a linear combination of atomic orbitals, analogous Eq. (812): $|\chi'_j\rangle = \sum_k C_{kj}|\psi_k\rangle$. Making the substitution and multiplying from the left with $\langle \psi_{k'}|$, we obtain:

$$\sum_k \langle \psi_{k'}|\hat{f}(j)|\psi_k\rangle C_{kj} = \epsilon_{jj} \sum_k \langle \psi_{k'}|\psi_j\rangle C_{kj}, \tag{832}$$

or, in matrix form,

$$\mathbf{FC} = \mathbf{SC}\epsilon' \tag{833}$$

where we have introduced the overlap matrix $S_{jk} = \langle \psi_j|\psi_k\rangle$, the Fock matrix $F_{jk} = \langle \psi_j|\hat{f}|\psi_k\rangle$, and the matrix of column eigenvectors C_{kj} defining the canonical molecular orbitals χ'_j, with energies ϵ_{jj}, expressed in the basis of atomic orbitals $|\psi_k\rangle$.

43.1 Restricted Closed Shell Hartree-Fock

The so-called *closed-shell restricted* Hartree-Fock method is essentially the Hartree-Fock approach implemented for systems with an *even* number n of electrons, with each orbital j populated by 2 electrons with opposite spins (one with spin α, described by ψ_j, and the other one with spin β, described by $\bar{\psi}_j$). It is called *restricted* to indicate that the spin states are restricted to be either α or β, and *closed shell* to indicate that each shell ψ_j is full with 2 electrons. The system is described by the Slater determinant,

$$
\begin{aligned}
|\Phi\rangle &= |\chi_1\chi_2...\chi_n\rangle \\
&= |\psi_1\bar{\psi}_1...\psi_{n/2}\bar{\psi}_{n/2}\rangle
\end{aligned}
\tag{834}
$$

where $\chi_1 = \psi_1$, $\chi_2 = \bar{\psi}_1$, ..., $\chi_{n-1} = \psi_{n/2}$, $\chi_n = \bar{\psi}_{n/2}$.

The energy of this *closed-shell restricted* Hartree-Fock wave function is computed, according to Eq. (818), by replacing the sums over n spin-orbitals χ_j by sums over $n/2$ spin-orbitals with spin α, ψ_j and $n/2$ spin-orbitals with spin β, $\bar{\psi}_j$, as follows:

$$
\begin{aligned}
\bar{E} &= \sum_{j=1}^{n/2}\langle\psi_j|\hat{h}^{(1)}|\psi_j\rangle + \frac{1}{2}\sum_{j=1}^{n/2}\sum_{k=1}^{n/2}\langle\psi_j\psi_k|\tfrac{e^2}{r_{12}}|\psi_j\psi_k\rangle - \langle\psi_j\psi_k|\tfrac{e^2}{r_{12}}|\psi_k\psi_j\rangle \\
&+ \frac{1}{2}\sum_{j=1}^{n/2}\sum_{k=1}^{n/2}\langle\psi_j\bar{\psi}_k|\tfrac{e^2}{r_{12}}|\psi_j\bar{\psi}_k\rangle - \cancel{\langle\psi_j\bar{\psi}_k|\tfrac{e^2}{r_{12}}|\bar{\psi}_k\psi_j\rangle} \\
&+ \sum_{j=1}^{n/2}\langle\bar{\psi}_j|\hat{h}^{(1)}|\bar{\psi}_j\rangle + \frac{1}{2}\sum_{j=1}^{n/2}\sum_{k=1}^{n}\langle\bar{\psi}_j\psi_k|\tfrac{e^2}{r_{12}}|\bar{\psi}_j\psi_k\rangle - \cancel{\langle\bar{\psi}_j\psi_k|\tfrac{e^2}{r_{12}}|\psi_k\bar{\psi}_j\rangle} \\
&+ \frac{1}{2}\sum_{j=1}^{n/2}\sum_{k=1}^{n/2}\langle\bar{\psi}_j\bar{\psi}_k|\tfrac{e^2}{r_{12}}|\bar{\psi}_j\bar{\psi}_k\rangle - \langle\bar{\psi}_j\bar{\psi}_k|\tfrac{e^2}{r_{12}}|\bar{\psi}_k\bar{\psi}_j\rangle,
\end{aligned}
\tag{835}
$$

where we can cross out the terms that cancel upon integration over the spin variable to obtain:

$$
\bar{E} = 2\sum_{j=1}^{n/2}\langle\psi_j|\hat{h}^{(1)}|\psi_j\rangle + \sum_{j=1}^{n/2}\sum_{k=1}^{n/2}2\langle\psi_j\psi_k|\tfrac{e^2}{r_{12}}|\psi_j\psi_k\rangle - \langle\psi_j\psi_k|\tfrac{e^2}{r_{12}}|\psi_k\psi_j\rangle.
\tag{836}
$$

Analogously to the general case, we minimize $\bar{E}$ with respect to ψ_j, subject to the constraint of orthonormal orbitals by applying the Lagrange multiplier method for the following functional:

$$
L(\psi_1, ..., \psi_n) = \bar{E} + \sum_j\sum_k\epsilon_{jk}[\langle\psi_j|\psi_k\rangle - \delta_{jk}],
\tag{837}
$$

where ϵ_{jk} are the Lagrange multipliers. Varying the spin orbitals ψ_j in an infinitesimal amount δ_j, with respect to expansion coefficients c_{kj}, we obtain:

$$
\delta L(\psi_1, ..., \psi_n) = \delta\bar{E} + \sum_{j=1}^{n}\sum_{k=1}^{n}\epsilon_{jk}[\langle\delta\psi_j|\psi_k\rangle + \langle\psi_j|\delta\psi_k\rangle].
\tag{838}
$$

Varying the spatial orbitals ψ_j in an infinitesimal amount δ_j with respect to the expansion coefficients c_j, we obtain:

$$
\begin{aligned}
\delta\bar{E} &= 2\sum_{j=1}^{n/2}\langle\delta\psi_j|\hat{h}^{(1)}|\psi_j\rangle + \sum_{j=1}^{n/2}\sum_{k=1}^{n/2}2\langle\delta\psi_j\psi_k|\tfrac{e^2}{r_{12}}|\psi_j\psi_k\rangle - \langle\delta\psi_j\psi_k|\tfrac{e^2}{r_{12}}|\psi_k\psi_j\rangle \\
&+ \sum_{j=1}^{n/2}\sum_{k=1}^{n/2}2\langle\psi_j\delta\psi_k|\tfrac{e^2}{r_{12}}|\psi_j\psi_k\rangle - \langle\psi_j\delta\psi_k|\tfrac{e^2}{r_{12}}|\psi_k\psi_j\rangle \\
&+ 2\sum_{j=1}^{n/2}\langle\psi_j|\hat{h}^{(1)}|\delta\psi_j\rangle + \sum_{j=1}^{n/2}\sum_{k=1}^{n/2}2\langle\psi_j\psi_k|\tfrac{e^2}{r_{12}}|\delta\psi_j\psi_k\rangle - \langle\psi_j\psi_k|\tfrac{e^2}{r_{12}}|\delta\psi_k\psi_j\rangle \\
&+ \sum_{j=1}^{n/2}\sum_{k=1}^{n/2}2\langle\psi_j\psi_k|\tfrac{e^2}{r_{12}}|\psi_j\delta\psi_k\rangle - \langle\psi_j\psi_k|\tfrac{e^2}{r_{12}}|\psi_k\delta\psi_j\rangle
\end{aligned}
\tag{839}
$$

which gives

$$\delta \bar{E} = 2\sum_{j=1}^{n/2}\langle\delta\psi_j|\hat{h}^{(1)}|\psi_j\rangle + 2\sum_{j=1}^{n/2}\sum_{k=1}^{n/2}2\langle\delta\psi_j\psi_k|\frac{e^2}{r_{12}}|\psi_j\psi_k\rangle - \langle\delta\psi_j\psi_k|\frac{e^2}{r_{12}}|\psi_k\psi_j\rangle + c.c. \tag{840}$$

Substituting Eq. (840) into Eq. (838), we obtain:

$$\delta L = 2\sum_{j=1}^{n/2}\langle\delta\psi_j|\left[\hat{h}^{(1)}|\psi_j\rangle + \sum_{k=1}^{n/2}2\langle\psi_k|\frac{e^2}{r_{12}}|\psi_k\rangle|\psi_j\rangle - \langle\psi_k|\frac{e^2}{r_{12}}\hat{P}_{12}|\psi_k\rangle|\psi_j\rangle + \epsilon_{jk}|\psi_k\rangle\right] + c.c. = 0, \tag{841}$$

which is satisfied when

$$\left[\hat{h}^{(1)} + \sum_{k=1}^{n/2}\langle\psi_k|\frac{e^2}{r_{12}}[2-\hat{P}_{12}]|\psi_k\rangle\right]|\psi_j\rangle = \epsilon_{jk}|\psi_k\rangle. \tag{842}$$

To write Eq. (842) in the canonical eigenvalue form, we change the basis set according to the unitary transformation,

$$|\psi_j'\rangle = \sum_k |\psi_k\rangle\Gamma_{kj}, \tag{843}$$

with $\Gamma\Gamma^\dagger = 1$. Considering that the Lagrange multipliers matrix ϵ is Hermitian (since the functional L is real), it is always possible to find a Γ that diagonalizes ϵ according to the similarity transformation:

$$\epsilon' = \Gamma^\dagger\epsilon\Gamma \tag{844}$$

Such a transformation defines the set of *canonical orbitals* $|\psi_j'\rangle$ for which

$$\hat{f}_{rhf}^{(1)}|\psi_j'\rangle = \epsilon_{jj}'|\psi_j'\rangle, \tag{845}$$

for $j = 1 - n/2$, where $\hat{f}_{rhf}^{(1)}$ is the *closed-shell restricted Hartree-Fock* operator,

$$\hat{f}_{rhf}^{(1)} = \hat{h}^{(1)} + \hat{V}_1^{rhf}, \tag{846}$$

where $\hat{V}_j^{rhf}$ is the *restricted Hartree-Fock potential* describing the interactions between electrons of the same spin, as follows:

$$\hat{V}_1^{rhf} = 2J_1(r_1) - \hat{X}_1(r_1), \tag{847}$$

where $J_1(r_1)$ is the Coulombic mean-field potential due to the presence of other electrons of the same spin,

$$J_1(r_1) = \sum_{k=1}^{n/2}\langle\psi_k'(2)|\frac{e^2}{r_{12}}|\psi_k'(2)\rangle. \tag{848}$$

Analogously, $\hat{X}_1(r_1)$ is defined as the *exchange operator*,

$$\hat{X}_1(r_1) = \sum_{k=1}^{n/2}\langle\psi_k'(2)|\frac{e^2}{r_{12}}\hat{P}_{12}|\psi_k'(2)\rangle, \tag{849}$$

where the permutation operator $\hat{P}_{12}$ interchanges electrons 1 and 2.

Equation (845) defines a self-consistent field (SCF) problem since $\hat{f}^{(1)}_{rhf}$ depends on ψ_j through $\hat{V}^{rhf}_1$. To solve this SCF problem, we first obtain approximate solutions ψ_j by approximating $\hat{f}^{(1)}_{rhf}$ by $\hat{h}_j$ (i.e., neglecting the electron-electron interactions introduced by J_1 and $\hat{X}_1$), or by using a semiempirical method (like the Hückel method described in these lectures). The resulting approximate solutions ψ_j are then used to compute J_1 and $\hat{X}_1$, giving a better approximation to $\hat{f}^{(1)}_{rhf}$ that can be used to obtain improved functions ψ_j. The process is repeated until convergence.

In practice, the restricted Hartree-Fock eigenvalue problem, introduced by Eq. (845), is solved with a set of matrix equations, obtained by substituting ψ'_j by a linear combination of atomic orbitals, analogous Eq. (812): $|\psi'_j\rangle = \sum_{k=1}^{n/2} C_{kj}|\zeta_k\rangle$. Making the substitution and multiplying from the left with $\langle\zeta_{k'}|$, we obtain:

$$\sum_{k=1}^{n/2}\langle\zeta_{k'}|\hat{f}^{(1)}_{rhf}|\zeta_k\rangle C_{kj} = \epsilon_{jj}\sum_{k=1}^{n/2}\langle\zeta_{k'}|\zeta_j\rangle C_{kj}, \tag{850}$$

or, in matrix form,

$$\mathbf{F}^{rhf}\mathbf{C} = \mathbf{S}\mathbf{C}\epsilon' \tag{851}$$

where we have introduced the overlap matrix $S_{jk} = \langle\zeta_j|\zeta_k\rangle$, the restricted Hartree Fock matrix

$$F^{rhf}_{jk} = \langle\zeta_j|\hat{f}^{(1)}_{rhf}|\zeta_k\rangle, \tag{852}$$

and the matrix of column eigenvectors C_{kj} defining the canonical orbitals ψ'_j, with energies ϵ_{jj}, in the basis of atomic orbitals $|\zeta_k\rangle$. Equations (851 are analogues of the Roothaan-Hall equations.

The electronic density $\rho(\mathbf{r})$ of the system with 2 electrons per orbital, populating the lowest $n/2$ states (*i.e.*, closed-shell Hartree-Fock density) can be computed, as follows:

$$\begin{aligned}\rho(\mathbf{r}) &= 2\sum_{k=1}^{n/2}\psi'^*_k(\mathbf{r})\psi'_k(\mathbf{r})\\ &= 2\sum_{lm}\zeta^*_l(\mathbf{r})\zeta_m(\mathbf{r})\sum_{k=1}^{n/2}C^*_{kl}C_{mk}\\ &= 2\sum_{lm}\zeta^*_l(\mathbf{r})\zeta_m(\mathbf{r})P_{lm}\end{aligned} \tag{853}$$

where

$$P_{lm} = \sum_{k=1}^{n/2}C^*_{kl}C_{mk} \tag{854}$$

define the elements of the electronic *density matrix* $\mathbf{P}$.

The elements of the density matrix, P_{lm}, are thus computed from the solution of the eigenvalue problem, introduced by Eq. (851). The resulting elements of the density matrix P_{lm} can be used to compute not only the density, according to Eq. (853), but also the restricted Hartree-Fock matrix, according to Eq. (852) since $\hat{f}^{(1)}_{rhf} = \hat{h}^{(1)} + \hat{V}^{rhf}_1$, with $\hat{V}^{rhf}_1 = 2J_1(\mathbf{r}_1) - \hat{X}(\mathbf{r}_1)$, where

$$\begin{aligned}J_1(\mathbf{r}_1) &= \sum_{k=1}^{n/2}\int d\mathbf{r}_2\psi'^*_k(\mathbf{r}_2)\frac{e^2}{r_{12}}\psi'_k(\mathbf{r}_2)\\ &= \sum_{k=1}^{n/2}\int d\mathbf{r}_2\sum_{l=1}^{n/2}\sum_{m=1}^{n/2}C^*_{km}\zeta^*_m(\mathbf{r}_2)\frac{e^2}{r_{12}}C_{lk}\zeta_l(\mathbf{r}_2)\\ &= \sum_{l=1}^{n/2}\sum_{m=1}^{n/2}\sum_{k=1}^{n/2}C^*_{km}C_{lk}\int d\mathbf{r}_2\zeta^*_m(\mathbf{r}_2)\frac{e^2}{r_{12}}\zeta_l(\mathbf{r}_2)\\ &= \sum_{l=1}^{n/2}\sum_{m=1}^{n/2}P_{ml}\int d\mathbf{r}_2\zeta^*_m(\mathbf{r}_2)\frac{e^2}{r_{12}}\zeta_l(\mathbf{r}_2)\end{aligned} \tag{855}$$

and

$$\begin{aligned}
\hat{X}_1(r_1) &= \sum_{k=1}^{n/2} \langle \psi_k' | \frac{e^2}{r_{jk}} \hat{P}_{kj} | \psi_k' \rangle, \\
&= \sum_{l=1}^{n/2} \sum_{m=1}^{n/2} P_{ml} \int d\mathbf{r}_2 \zeta_m^*(\mathbf{r}_2) \frac{e^2}{r_{12}} \hat{P}_{12} \zeta_l(\mathbf{r}_2).
\end{aligned} \tag{856}$$

Therefore,

$$\hat{f}_{rhf}^{(1)}(\mathbf{r}_1) = \hat{h}^{(1)} + \sum_{l=1}^{n/2} \sum_{m=1}^{n/2} P_{ml} \left[2\langle \zeta_m(\mathbf{r}_2) | \frac{e^2}{r_{12}} | \zeta_l(\mathbf{r}_2) \rangle - \langle \zeta_m(\mathbf{r}_2) | \frac{e^2}{r_{12}} \hat{P}_{12} | \zeta_l(\mathbf{r}_2) \rangle \right], \tag{857}$$

and

$$\begin{aligned}
F_{jk}^{rhf} &= \langle \zeta_j | \hat{f}_{rhf}^{(1)} | \zeta_k \rangle, \\
&= H_{jk}^{core} + G_{jk},
\end{aligned} \tag{858}$$

with

$$H_{jk}^{core} = \langle \zeta_j(1) | \hat{h}^{(1)} | \zeta_k(1) \rangle, \tag{859}$$

and

$$G_{jk} = \sum_{l=1}^{n/2} \sum_{m=1}^{n/2} P_{ml} \left[2\langle \zeta_j(\mathbf{r}_1) \zeta_m(\mathbf{r}_2) | \frac{e^2}{r_{12}} | \zeta_l(\mathbf{r}_2) \zeta_k(\mathbf{r}_1) \rangle - \langle \zeta_j(\mathbf{r}_1) \zeta_m(\mathbf{r}_2) | \frac{e^2}{r_{12}} \hat{P}_{12} | \zeta_l(\mathbf{r}_2) \zeta_k(\mathbf{r}_1) \rangle \right]. \tag{860}$$

To solve Eq. (851), we first diagonalize the overlap matrix by computing the matrix $\mathbf{X} = \mathbf{S}^{-1/2}$ that transforms the overlap matrix into the identity matrix, as follows: $\mathbf{X}^\dagger \mathbf{S} \mathbf{X} = \mathbf{1}$. Then, we introduce the matrix $\tilde{\mathbf{C}}$, as follows:

$$\mathbf{C} = \mathbf{X}\tilde{\mathbf{C}}, \tag{861}$$

that, according to Eq. (851), satisfies the eigenvalue problem:

$$\mathbf{F}^{rhf} \mathbf{X} \tilde{\mathbf{C}} = \mathbf{S} \mathbf{X} \tilde{\mathbf{C}} \epsilon' \tag{862}$$

or

$$\tilde{\mathbf{F}}^{rhf} \tilde{\mathbf{C}} = \tilde{\mathbf{C}} \epsilon' \tag{863}$$

with

$$\tilde{\mathbf{F}}^{rhf} = \mathbf{X}^\dagger \mathbf{F}^{rhf} \mathbf{X}. \tag{864}$$

These equations allow for the implementation of the self-consistent-field restricted Hartree-Fock (SCF RHF) method, for a fixed nuclear configuration, as follows:

1. Calculate the matrix elements S_{jk}, H_{jk}^{core} and the 2-electron integrals $\langle \zeta_j(\mathbf{r}_1) \zeta_m(\mathbf{r}_2) | \frac{e^2}{r_{12}} | \zeta_l(\mathbf{r}_2) \zeta_k(\mathbf{r}_1) \rangle$ and $\langle \zeta_j(\mathbf{r}_1) \zeta_m(\mathbf{r}_2) | \frac{e^2}{r_{12}} | \zeta_l(\mathbf{r}_1) \zeta_k(\mathbf{r}_2) \rangle$.

2. Diagonalize S_{jk} to obtain X_{jk}

3. Obtain an approximate density matrix P_{jk}, according to Eq. (854), by solving Eq. (851) with $\hat{f}^{(1)} \approx \hat{h}^{(1)}$, or $\hat{f}^{(1)} \approx \hat{h}_{EH}^{(1)}$, where $\hat{h}_{EH}^{(1)}$ is the semiempirical extended-Hückel Hamiltonian of the system.

4. Compute the matrix elements G_{jk}, according to Eq. (860), using P_{jk} and the 2-electron integrals.

5. Compute the Fock matrix F_{jk}^{rhf}, according to Eq. (858), by using H_{jk}^{core}, P_{jk} and the 2-electron integrals.

6. Compute the transformed Fock matrix $\tilde{F}_{jk}^{rhf}$ by using F_{jk}^{rhf} and X_{jk}, according to Eq. (864).

7. Obtain $\tilde{C}$ and ϵ' by solving Eq. (863).

8. Calculate C by using Eq. (861).

9. Compute a new density matrix P, according to Eq. (854), based on C obtained in (8).

10. If P has changed more than a given tolerance, relative to the previous iteration, go to (4). Otherwise, the SCF calculation has converged and the solution is given by the current eigenvectors C and eigenvalues ϵ.

The total energy provided by Hartree-Fock theory is usually satisfactory since it is dominated by high-density inner-shell electrons that are well described by HF. However, the description of low-density valence electrons provided by HF theory is usually unsatisfactoy since it neglects important correlation energy terms. While correlation effects can be addressed through configuration-interaction corrections, the complexity of the corrections, their sensitivity to the choice of basis functions, and the increase in effort required with the decrease in spacing between energy levels, preclude application to large systems. In addition, the long range of the Coulomb interaction produces unrealistic features in the HF energy eigenvalues, such as vanishing density of states at the Fermi level in metals, and unphysically large band gaps in insulators. The density-functional theory, described in the following section, provides an alternative approach.

44   Density Functional Theory

In the Kohn-Sham formulation of density functional theory (DFT), the total electronic energy is expressed as a sum of electronic kinetic energy T, electron-nuclear interaction V, Coulomb self-interaction U of the electron density ρ and the remaining part of the electron-electron repulsion energy due to exchange-correlation E^{XC}, all expressed as functionals of the density ρ, as follows:

$$E = T + V + U + E^{XC}. \tag{865}$$

Considering a spin-unrestricted format, as described by Pople, α and β electrons are assigned to sets of orthonormal orbitals ψ_i^α with $i = 1, .., n_\alpha$ and ψ_i^β with $i = 1, .., n_\beta$, respectively, defining a single Slater determinant. The corresponding total density is then obtained as the sum of the α and β densities,

$$\rho = \rho^\alpha + \rho^\beta, \tag{866}$$

with

$$\rho_\alpha = \sum_{i=1}^{n_\alpha} |\psi_i^\alpha|^2,$$
$$\rho_\beta = \sum_{j=1}^{n_\beta} |\psi_j^\beta|^2. \tag{867}$$

The energies T, V and U are defined, as follows:

$$T = -\frac{1}{2} \sum_{i=1}^{n_\alpha} \langle \psi_i^\alpha | \nabla^2 | \psi_i^\alpha \rangle - \frac{1}{2} \sum_{j=1}^{n_\beta} \langle \psi_j^\beta | \nabla^2 | \psi_j^\beta \rangle, \tag{868}$$

$$V = -\sum_{j=1}^{N_{nucl}} Z_j \int \frac{\rho(\mathbf{r})}{|\mathbf{r} - \mathbf{r}_j|} d\mathbf{r}, \tag{869}$$

$$U = \frac{1}{2} \int \int \frac{\rho(\mathbf{r}_1)\rho(\mathbf{r}_2)}{|\mathbf{r}_2 - \mathbf{r}_1|} d\mathbf{r}_1 d\mathbf{r}_2. \tag{870}$$

The exchange-correlation energy E^{XC} is typically approximated by a functional f of the densities and their gradients, as follows:

$$E^{XC} = \int f(\rho_\alpha, \rho_\beta, \gamma_{\alpha\alpha}, \gamma_{\alpha\beta}, \gamma_{\beta\beta}) d\mathbf{r}, \tag{871}$$

where

$$\gamma_{jk} = \nabla \rho_j \cdot \nabla \rho_k. \tag{872}$$

Expanding the orthonormal orbitals ψ_i^α in the basis of atomic orbitals (AOs) ϕ_μ,

$$\psi_i^\alpha = \sum_\mu c_{\mu,i}^\alpha \phi_\mu, \tag{873}$$

we obtain:

$$\rho_\alpha = \sum_{i=1}^{n_\alpha} \sum_\mu \sum_\nu (c_{\mu i}^\alpha)^* c_{\nu i}^\alpha \phi_\mu^* \phi_\nu,$$
$$= \sum_\mu \sum_\nu P_{\mu,\nu}^\alpha \phi_\mu^* \phi_\nu, \tag{874}$$

190

and similarly for ρ^β, where $P^\alpha_{\mu,\nu}$ in Eq. (874) is the AO density matrix of α electrons. Substituting these expressions into Eq. (865) and then minimizing with respect to the expansion coefficients, as done for the Hartree-Fock method, we obtain the equations,

$$\sum_\nu (F^\alpha_{\mu\nu} - \epsilon^\alpha_i S_{\mu\nu})c^\alpha_{\nu i} = 0, \tag{875}$$

analogous to the Roothaan-Hall Eqs. (851) of Hartree-Fock theory. The only difference is that the Fock matrix is replaced by the Fock-type matrices,

$$\begin{aligned} F^\alpha_{\mu\nu} &= H^{core}_{\mu\nu} + J_{\mu\nu} + F^{XC\alpha}_{\mu\nu}, \\ F^\beta_{\mu\nu} &= H^{core}_{\mu\nu} + J_{\mu\nu} + F^{XC\beta}_{\mu\nu}. \end{aligned} \tag{876}$$

Here, $S_{\mu\nu}$ and $H^{core}_{\mu\nu}$ are the overlap and bare nucleus Hamiltonian matrices, $J_{\mu\nu}$ is the Coulomb matrix

$$J_{\mu\nu} = \sum_{\lambda\sigma} \left(P^\alpha_{\lambda\sigma} + P^\beta_{\lambda\sigma} \right) \langle \mu\nu | \lambda\sigma \rangle. \tag{877}$$

and $F^{XC\alpha}_{\mu\nu}$ are given, as follows:

$$F^{XC\alpha}_{\mu\nu} = \int d\mathbf{r} \left[\frac{\partial f}{\partial \rho_\alpha} \phi_\mu \phi_\nu + \left(2\frac{\partial f}{\partial \gamma_{\alpha\alpha}} \nabla \rho_\alpha + \frac{\partial f}{\partial \gamma_{\alpha\beta}} \nabla \rho_\beta \right) \cdot \nabla(\phi_\mu \phi_\nu) \right]. \tag{878}$$

Expressions and Fortran codes for various versions of proposed f, defining popular functionals such as LDA, PBE, B3LYP, etc., and the corresponding derivatives $\frac{\partial f}{\partial \rho_\alpha}$, $\frac{\partial f}{\partial \gamma_{\alpha\alpha}}$ and $\frac{\partial f}{\partial \gamma_{\alpha\beta}}$ are available at the CCLRC DFT repository . The Fortran codes were automatically generated with dfauto from Maple expressions.

Once the equations have been solved to find the expansion coefficients $c^\alpha_{\nu i}$ through an iterative self-consistent procedure, the Kohn-Sham energy is obtained from Eq. (865), as follows:

$$E = \sum_{\mu\nu} P_{\mu\nu} H^{core}_{\mu\nu} + \frac{1}{2} \sum_{\mu\nu} \sum_{\lambda\sigma} P_{\mu\nu} P_{\lambda\sigma} \langle \mu\nu | \lambda\sigma \rangle + E^{XC}. \tag{879}$$

44.1   Hohenberg and Kohn Theorems

The underlying concept, introduced by Eq. (865), is that the minimum energy is a unique functional of the ground state density (*i.e.*, the minimum energy of a non-degenerate state is uniquely determined by the ground state density). Such concept is a consequence of the Hohenberg and Kohn first theorem: '*For any system of interacting particles in an external potential V, the ground state density is uniquely determined (*i.e., *the potential is a unique functional of the density, to within an additive constant)*'.

The theorem is demonstrated by considering two external potentials V and V', differing by more than a constant, giving the same ground state density ρ:

$$\begin{aligned} E' &= \langle \psi' | H' | \psi' \rangle < \langle \psi | H' | \psi \rangle = \langle \psi | H + (V' - V) | \psi \rangle, \\ E' &< E + \int d\mathbf{r} \rho(\mathbf{r})(V'(\mathbf{r}) - V(\mathbf{r})), \end{aligned} \tag{880}$$

where the inequality is strict because ψ and ψ' are different (*i.e.*, eigenstates of different Hamiltonians). Exchanging the primed and unprimed quantities, and assuming $\rho = \rho'$, we obtain:

$$E = \langle \psi | H | \psi \rangle < \langle \psi' | H | \psi' \rangle = \langle \psi' | H' + (V - V') | \psi' \rangle,$$

$$E < E' + \int d\mathbf{r} \rho(\mathbf{r})(V(\mathbf{r}) - V'(\mathbf{r})), \tag{881}$$

Summing Eqs. (880) and (881), we obtain: $E' + E < E + E'$ which is an absurd. Therefore, two potentials that differ in more than a constant value cannot define the same density, so the density defines a unique potential (except for a constant).

As mentioned above, a straightforward consequence of the first theorem is Hohenberg and Kohn second theorem stating that *the ground state energy E_0 is uniquely determined by a functional of the density*, as follows:

$$E = F[\rho] + \int d\mathbf{r} \rho(\mathbf{r}) V(\mathbf{r}), \tag{882}$$

where $F[\rho]$ is a a universal functional valid for any number of particles and any external potential V. The second theorem can be proved by using the variational theorem with a trial state ψ, as follows:

$$\langle \psi | \hat{H} | \psi \rangle = F[\rho] + \int d\mathbf{r} \rho(\mathbf{r}) V(\mathbf{r}) \geqslant E_0. \tag{883}$$

where the equality holds only when ψ is the ground state for $V(\mathbf{r})$. Note that one can write the energy as a functional of the density because the external potential is uniquely determined by the density and since the potential in turn uniquely (except in degenerate situations) determines the ground state wavefunction, all the other observables of the system such as kinetic energy are uniquely determined.

44.2   Kohn Sham Equations

Kohn and Sham mapped the system of interacting electrons onto a fictitious system of non-interacting 'electrons', and wrote the variational problem for the Hohenberg-Kohn density-functional by introducing a Lagrange multiplier μ that constrains the number of electrons to N, as follows:

$$\delta \left[F[\rho] + \int d\mathbf{r} \rho(\mathbf{r}) V(\mathbf{r}) - \mu \left(\int d\mathbf{r} \rho(\mathbf{r}) - N \right) \right] = 0. \tag{884}$$

$F[\rho]$ is separated into the kinetic energy $T[\rho]$ of the non-interacting electron gas with density ρ, the classical electrostatic potential, and the non-classical term due to exchange-correlation accounting for the difference between the kinetic energies of the interacting and non-interacting electrons, as follows:

$$F[\rho] = T[\rho] + \frac{1}{2} \int d\mathbf{r} \int d\mathbf{r}' \frac{\rho(\mathbf{r})\rho(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} + E_{XC}[\rho]. \tag{885}$$

The first two terms are simple and 'classical' while the third term describes the complex behaviour of correlated electrons and is usually approximated by using proper interpolation between asymptotic limits.

Substituting Eq. (885) into Eq. (884), we obtain:

$$\frac{\delta T[\rho]}{\delta \rho(\mathbf{r})} + V_{KS}(\mathbf{r}) = \mu, \tag{886}$$

where the Kohn-Sham potential $V_{KS}(\mathbf{r})$ is defined, as follows:

$$V_{KS}(\mathbf{r}) = \int d\mathbf{r}' \frac{\rho(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} + V_{XC}(\mathbf{r}) + V(\mathbf{r}). \tag{887}$$

Here, we have introduced the exchange-correlation potential $V_{XC}(\mathbf{r}) = \frac{\delta E_{XC}[\rho]}{\delta \rho(\mathbf{r})}$.

Equation (886) is equivalent to the description of a system of non-interacting electrons in an external potential $V_{KS}(\mathbf{r})$ for which the ground state can be found by solving the one-electron Schrödinger equation:

$$H_{KS}\psi_j(\mathbf{r}) = E_j\psi_j(\mathbf{r}), \tag{888}$$

where we have introduced the *Kohn-Sham orbitals* ψ_j that are eigenstates of the DFT Hamiltonian,

$$H_{KS} = -\frac{1}{2}\nabla^2 + V_{KS}(\mathbf{r}). \tag{889}$$

The density can be computed, as follows:

$$\rho(\mathbf{r}) = 2\sum_{j=1}^{N/2} \psi_j(\mathbf{r})^*\psi_j(\mathbf{r}), \tag{890}$$

after obtaining the *Kohn-Sham orbitals*. However, to obtain the *Kohn-Sham orbitals* we need the density since, according to Eq. (887), $V_{KS}(\mathbf{r})$ depends on the density $\rho(\mathbf{r})$. So, it is necessary to solve Eq. (888) self-consistently. Having an initial guess for the density, approximate Kohn-Sham orbitals ψ_j are found by solving Eq. (888) and the density $\rho(\mathbf{r})$ is updated according to Eq. (890). The procedure is repeated multiple times until the input and output densities are the same.

The total energy of the system of interacting electrons,

$$E = T[\rho] + \frac{1}{2}\int d\mathbf{r}\int d\mathbf{r}' \frac{\rho(\mathbf{r})\rho(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} + \int d\mathbf{r}\rho(\mathbf{r})V(\mathbf{r}) + E_{XC}[\rho], \tag{891}$$

can be computed from the energy $E^{non-int}$ of the non-interacting system,

$$E^{non-int} = 2\sum_{j=1}^{N/2} E_j = T[\rho] + \int d\mathbf{r}\rho(\mathbf{r})V_{KS}(\mathbf{r}),$$

$$= T[\rho] + \int d\mathbf{r}\int d\mathbf{r}' \frac{\rho(\mathbf{r})\rho(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} + \int d\mathbf{r}\rho(\mathbf{r})[V_{XC}(\mathbf{r}) + V(\mathbf{r})], \tag{892}$$

as follows:

$$E = E^{non-int} - \frac{1}{2}\int d\mathbf{r}\int d\mathbf{r}' \frac{\rho(\mathbf{r})\rho(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} - \int d\mathbf{r}\rho(\mathbf{r})V_{XC}(\mathbf{r}) + E_{XC}[\rho]. \tag{893}$$

44.3 Thomas-Fermi Functional

The Thomas-Fermi model functional assumes a uniform distribution of electrons in phase space, with 2 electrons per element of phase-space volume h^3. According to such uniform phase-space distribution, the number of electrons $\Delta N(\mathbf{r})$ in a volume ΔV at $\mathbf{r}$ is

$$\Delta N(\mathbf{r}) = \frac{2}{h^3}\Delta V \frac{4}{3}\pi p_F(\mathbf{r})^3, \tag{894}$$

where $p_F(\mathbf{r})$ is the maximum value of momentum for electrons in ΔV, as determined by the local density $\rho(\mathbf{r})$, as follows:

$$\rho(\mathbf{r}) = \frac{2}{h^3}\frac{4}{3}\pi p_F(\mathbf{r})^3. \tag{895}$$

The local density $\rho(\mathbf{r})$ also determines the *Wigner-Seitz radius* $r_s(\mathbf{r})$, defined as the radius of a sphere with the mean volume per electron (or, mean volume per atom in metals where each atom contributes with a single electron to the electronic structure of interest), as follows:

$$\frac{4}{3}\pi r_s^3 = \rho(\mathbf{r}). \tag{896}$$

The fraction of electrons with momentum between p and $p + dp$ is $4\pi p^2 dp/(\frac{4}{3}\pi p_F(\mathbf{r})^3)$, since $4\pi p^2$ is the surface of a sphere of momentum p, while $4\pi p^2 dp$ is the volume between the surfaces of momentum p and $p + dp$ and $4/3\pi p_F^3$ is the total volume of mometum in the element of phase-space volume h^3. Therefore, the kinetic energy per unit volume is

$$\begin{aligned}
T(\mathbf{r}) &= \rho(\mathbf{r})\int_0^{p_F} dp \frac{p^2}{2m_e}\frac{4\pi p^2}{\frac{4}{3}\pi p_F(\mathbf{r})^3}, \\
&= \int_0^{p_F} dp \frac{p^2}{2m_e}\frac{8\pi p^2}{h^3}, \\
&= \frac{p_F^5}{10m_e}\frac{8\pi}{h^3}, \\
&= \left(\frac{3h^3\rho(\mathbf{r})}{8\pi}\right)^{5/3}\frac{1}{10m_e}\frac{8\pi}{h^3}, \\
&= \left(\frac{3}{8\pi}\right)^{2/3}\frac{3h^2}{10m_e}\rho(\mathbf{r})^{5/3}.
\end{aligned} \tag{897}$$

giving the total kinetic energy as a functional of the density,

$$T = \left(\frac{3}{8\pi}\right)^{2/3}\frac{3h^2}{10m_e}\int d\mathbf{r}\rho(\mathbf{r})^{5/3}. \tag{898}$$

The complete energy, can now be written as a functional of the density, as follows:

$$E[\rho] = \left(\frac{3}{8\pi}\right)^{2/3}\frac{3h^2}{10m_e}\int d\mathbf{r}\rho(\mathbf{r})^{5/3} + \frac{1}{2}\int d\mathbf{r}\int d\mathbf{r}'\frac{\rho(\mathbf{r})\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} - \sum_{j=1}^{N_{nucl}} Z_j\int \frac{\rho(\mathbf{r})}{|\mathbf{r}-\mathbf{r}_j|}d\mathbf{r} + E^{XC}[\rho]. \tag{899}$$

The *Thomas-Fermi model functional* neglects the E_{XC} term in Eq (899). However, such an approximation usually introduces significant errors. For example, in the one-electron limit (i.e., for one-electron systems), $E_C = 0$. Therefore, $E_{XC} = E_X$, with $E_X = -U$ according to Hartree-Fock theory (see, Eq. (808)). Neglecting E_{XC} would thus introduce error since the self-interaction energy U would no longer cancel out with the exchange term. *Self-interaction corrections* (SIC) can be introduced to avoid that error. The earliest SIC was proposed by E. Fermi and E. Amaldi [Accad. Ital. Rome 6, 119 (1934)], who replaced $U[\rho]$ by $U[\rho] - NU[\rho/N]$, where N is the number of electrons in the system, giving the so-called *Thomas-Fermi-Amaldi* (TFA) functional:

$$E^{TFA} = \left(\frac{3}{8\pi}\right)^{2/3} \frac{3h^2}{10m_e} \int d\mathbf{r}\rho(\mathbf{r})^{5/3} + \frac{1}{2}\frac{N-1}{N} \int d\mathbf{r} \int d\mathbf{r}' \frac{\rho(\mathbf{r})\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} - \sum_{j=1}^{N_{nucl}} Z_j \int \frac{\rho(\mathbf{r})}{|\mathbf{r}-\mathbf{r}_j|}d\mathbf{r}. \tag{900}$$

44.4 Local Density Approximation

The local density approximation (LDA) states that if the charge density is sufficiently slowly varying, the exchange correlation energy is approximately

$$E_{XC}[\rho] = \int d\mathbf{r}\rho(\mathbf{r})E_{XC}^{HEG}(\rho(\mathbf{r})), \tag{901}$$

where $E_{XC}^{HEG}(\rho(\mathbf{r}))$ is the exchange-correlation energy of a homogeneous electron gas (HEG) with the same local charge density $\rho(\mathbf{r})$.

45   Imaginary Time Propagation

Consider the problem of solving the time dependent Schrödinger equation,

$$i\hbar\frac{\partial\Psi}{\partial t} = \hat{H}\Psi, \tag{902}$$

with $\hat{H} = \hat{p}^2/(2m) + V(x)$ and

$$\Psi = \Psi_R + i\Psi_I, \tag{903}$$

with real valued Ψ_R and Ψ_I. Substituting Eq. (903) into Eq. (902), we obtain:

$$\hbar\frac{\partial\Psi_R}{\partial t} = \hat{H}\Psi_I,$$
$$-\hbar\frac{\partial\Psi_I}{\partial t} = \hat{H}\Psi_R, \tag{904}$$

or,

$$\hbar\frac{\partial}{\partial t}\begin{pmatrix} \Psi_R \\ \Psi_I \end{pmatrix} = \begin{pmatrix} 0 & H \\ -H & 0 \end{pmatrix}\begin{pmatrix} \Psi_R \\ \Psi_I \end{pmatrix}. \tag{905}$$

Therefore,

$$\begin{pmatrix} \Psi_R(t) \\ \Psi_I(t) \end{pmatrix} = \exp\left[-\frac{i}{\hbar}\begin{pmatrix} 0 & H \\ -H & 0 \end{pmatrix}\tilde{t}\right]\begin{pmatrix} \Psi_R(0) \\ \Psi_I(0) \end{pmatrix}, \tag{906}$$

with $\tilde{t} = it$.

$$\begin{pmatrix} \Psi_R(t) \\ \Psi_I(t) \end{pmatrix} = \exp\left[-\frac{i}{\hbar}\mathbf{L}^\dagger\begin{pmatrix} -iH & 0 \\ 0 & iH \end{pmatrix}\mathbf{L}\tilde{t}\right]\begin{pmatrix} \Psi_R(0) \\ \Psi_I(0) \end{pmatrix}, \tag{907}$$

with

$$\mathbf{L} = \begin{pmatrix} i/\sqrt{2} & -1/\sqrt{2} \\ i/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}, \tag{908}$$

and $\mathbf{L}^\dagger_{kj} = \mathbf{L}^*_{jk}$.

$$\begin{pmatrix} \Psi_R(t) \\ \Psi_I(t) \end{pmatrix} = \mathbf{L}^\dagger\begin{pmatrix} e^{-\frac{i}{\hbar}Ht} & 0 \\ 0 & e^{\frac{i}{\hbar}Ht} \end{pmatrix}\mathbf{L}\begin{pmatrix} \Psi_R(0) \\ \Psi_I(0) \end{pmatrix}, \tag{909}$$

46   Lagrangian Formulation of Quantum Mechanics

Equations (904), and thus Eq. (909), can also be derived from the Euler-Lagrange equation:

$$\frac{\partial\mathcal{L}}{\partial\Psi_j} - \frac{\partial}{\partial x}\frac{\partial\mathcal{L}}{\partial\Psi'_j} - \frac{\partial}{\partial t}\frac{\partial\mathcal{L}}{\partial\dot{\Psi}_j} = 0, \tag{910}$$

with $j = (R, I)$ and

$$
\begin{aligned}
\mathcal{L} &= \hbar(\dot{\Psi}_R \Psi_I - \dot{\Psi}_I \Psi_R) - \frac{\hbar^2}{2m}(\Psi_R'^2 + \Psi_I'^2) - V(x)(\Psi_I^2 + \Psi_R^2) \\
&= -\hbar \mathrm{Im}\left[\frac{\dot{\Psi}}{\Psi}\right] \Psi' \Psi'^* - \frac{\hbar^2}{2m}\Psi'\Psi'^* - V\Psi\Psi^* \\
&= \Psi\Psi^*\left[-\frac{\hbar^2}{2m}\frac{\Psi'\Psi'^*}{\Psi\Psi^*} - \left(V(x) + \hbar\mathrm{Im}\left[\frac{\dot{\Psi}}{\Psi}\right]\right)\right].
\end{aligned}
\tag{911}
$$

Introducing the substitution $\Psi = Re^{\frac{i}{\hbar}S}$, with real valued R and S, we obtain:

$$
\mathcal{L} = -\frac{\hbar^2 R'^2}{2m} - R^2\frac{S'^2}{2m} - R^2\left(V + \dot{S}\right).
\tag{912}
$$

Substituting Eq. (912) into the following Euler-Lagrange equations,

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial R} - \frac{\partial}{\partial x}\frac{\partial \mathcal{L}}{\partial R'} - \frac{\partial}{\partial t}\frac{\partial \mathcal{L}}{\partial \dot{R}} &= 0, \\
\frac{\partial \mathcal{L}}{\partial S} - \frac{\partial}{\partial x}\frac{\partial \mathcal{L}}{\partial S'} - \frac{\partial}{\partial t}\frac{\partial \mathcal{L}}{\partial \dot{S}} &= 0,
\end{aligned}
\tag{913}
$$

we obtain the quantum Hamilton-Jacobi equation:

$$
\dot{S} + \frac{S'^2}{2m} + V - \frac{\hbar^2}{2m}\frac{R''}{R} = 0,
\tag{914}
$$

and the continuity equation

$$
\frac{\partial R^2}{\partial t} + \nabla\left(R^2\frac{S'}{m}\right) = 0,
\tag{915}
$$

or, in terms of the density $\rho = R^2$, we obtain:

$$
\mathcal{L} = \rho\left[-\frac{\hbar^2}{2m}\left(\frac{\rho'}{2\rho}\right)^2 - \left[\frac{S'^2}{2m} + (V + \dot{S})\right]\right],
\tag{916}
$$

and

$$
\begin{aligned}
\dot{S} + \frac{S'^2}{2m} + V - \frac{\hbar^2}{4m}\left(\frac{\rho''}{\rho} - \frac{\rho'^2}{2\rho^2}\right) &= 0, \\
\frac{\partial \rho}{\partial t} + \nabla\left(\rho^2\frac{S'}{m}\right) &= 0.
\end{aligned}
\tag{917}
$$

Using the Lagrangian formulation, we seek for the equations of motion for the variables $p(t)$, $q(t)$, $Q(t)$, $P(t)$ and $\tilde{S}(x, t)$ that define the following ansatz:

$$
\Psi(x, t) = \pi^{-1/4}\hbar^{-1/4}Q^{-1/2}e^{-PQ^{-1}(x-q)^2/(2\hbar)}e^{\frac{i}{\hbar}S(x,t)},
\tag{918}
$$

197

with $S(x,t) = S_a(x,t) + p(t)[x - q(t)]$.

   We note that

$$\Psi\Psi^* = \pi^{-1/2}\hbar^{-1/2}Q^{-1/2}\bar{Q}^{-1/2}e^{-(PQ^{-1}+\bar{P}\bar{Q}^{-1})(x-q)^2/(2\hbar)}e^{\frac{i}{\hbar}(S-\bar{S})}$$

$$\Psi'(x,t) = \Psi(x,t)(-PQ^{-1}(x-q)/\hbar + \frac{i}{\hbar}S'),$$

$$\hbar\frac{\dot{\Psi}(x,t)}{\Psi(x,t)} = -\frac{\hbar}{2}\dot{Q}Q^{-1} - \frac{1}{2}\left[\dot{P}Q^{-1} - P\dot{Q}Q^{-2}\right](x-q)^2 - PQ^{-1}(x-q)\dot{q} + i\dot{S},$$

$$\hbar\text{Im}\left[\frac{\dot{\Psi}(x,t)}{\Psi(x,t)}\right] = -\frac{\hbar}{4i}(\dot{Q}Q^{-1} - \dot{\bar{Q}}\bar{Q}^{-1}) - \frac{1}{4i}\left[\dot{P}Q^{-1} - \dot{\bar{P}}\bar{Q}^{-1} - P\dot{Q}Q^{-2}\right. \tag{919}$$

$$\left. +\bar{P}\dot{\bar{Q}}Q^{-2}\right](x-q)^2 - \frac{1}{2i}(PQ^{-1} - \bar{P}\bar{Q}^{-1})(x-q)\dot{q} + \frac{1}{2}(\dot{S} + \dot{\bar{S}}),$$

$$-\frac{\hbar^2}{2m}\frac{\Psi'\Psi'^*}{\Psi\Psi^*} = \frac{1}{2m}(-PQ^{-1}(x-q) + iS')(\bar{P}\bar{Q}^{-1}(x-q) + i\bar{S}'),$$

Therefore,

$$\mathcal{L} = \Psi\Psi^*\left[-\frac{\hbar^2}{2m}\frac{\Psi'\Psi'^*}{\Psi\Psi^*} - \left(V(x) + \hbar\text{Im}\left[\frac{\dot{\Psi}}{\Psi}\right]\right)\right],$$

$$= \pi^{-1/2}\hbar^{-1/2}Q^{-1/2}\bar{Q}^{-1/2}e^{-(PQ^{-1}+\bar{P}\bar{Q}^{-1})(x-q)^2/(2\hbar)}e^{\frac{i}{\hbar}(S_a - \bar{S}_a)}$$

$$\times\left[\frac{1}{2m}[-PQ^{-1}(x-q) + i(S'_a + p)][\bar{P}\bar{Q}^{-1}(x-q) + i(\bar{S}'_a + p)] - V(x) + \right. \tag{920}$$

$$+\frac{\hbar}{4i}(\dot{Q}Q^{-1} - \dot{\bar{Q}}\bar{Q}^{-1}) + \frac{1}{4i}\left[\dot{P}Q^{-1} - \dot{\bar{P}}\bar{Q}^{-1} - P\dot{Q}Q^{-2} + \bar{P}\dot{\bar{Q}}Q^{-2}\right](x-q)^2$$

$$\left. +\frac{1}{2i}(PQ^{-1} - \bar{P}\bar{Q}^{-1})(x-q)\dot{q} - \frac{1}{2}(\dot{S}_a + \dot{\bar{S}}_a) - \dot{p}(x-q) + p\dot{q}\right].$$

47   Second Quantization Mapping

The goal of this section is to introduce the single-particle basis $\{\psi_{\nu_1}(\mathbf{r}), \psi_{\nu_2}(\mathbf{r}), \psi_{\nu_3}(\mathbf{r}), \cdots\}$ for representation of the N-particle state $\Psi(\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_N)$ in terms of symmetrized product states $\hat{S}_\pm \prod_{j=1}^N \psi_{\nu_j}(\mathbf{r}_j)$, and its correspondence to the occupation number representation $|n_{\nu_1}, n_{\nu_2}, n_{\nu_3}, \cdots\rangle$, where n_{ν_j} is the number of particles in state $\psi_{\nu_j}(\mathbf{r})$ in the product state representation. Furthermore, we introduce the creation $\hat{a}_j^\dagger$ and anihilation $\hat{a}_j$ operators (i.e., operators that raise or lower the occupation numbers n_{ν_j} by one unit) and we show that any single particle operator $\hat{A}$ can be expressed in terms of $\hat{a}_j^\dagger$ and $\hat{a}_j$, as follows: $\hat{A} = \sum_{\nu_j, \nu_k} A_{\nu_j, \nu_k} \hat{a}_j^\dagger \hat{a}_k$, with $A_{\nu_j, \nu_k} = \langle \nu_j | \hat{A} | \nu_k \rangle$.

47.1   Single-Particle Basis

The state of the N-particle system $\Psi(\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_N)$ can be represented in a complete orthonormal basis composed of single-particle states $\{\psi_{\nu_j}(\mathbf{r})\}$, satisfying that

$$\sum_{\nu_j} \psi_{\nu_j}(\mathbf{r}')^* \psi_{\nu_j}(\mathbf{r}) = \delta(\mathbf{r}' - \mathbf{r}), \tag{921}$$

and

$$\int d\mathbf{r}\, \psi_{\nu_j}(\mathbf{r})^* \psi_{\nu_k}(\mathbf{r}) = \delta_{\nu_j \nu_k}. \tag{922}$$

To represent $\Psi(\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_N)$, we first project the state along the basis set of $\mathbf{r}_1$, as follows:

$$\Psi(\mathbf{r}_1, \mathbf{r}'_2, \cdots, \mathbf{r}'_N) = \sum_{\nu_1} \psi_{\nu_1}(\mathbf{r}_1) \int d\mathbf{r}'_1 \psi_{\nu_1}(\mathbf{r}'_1)^* \Psi(\mathbf{r}'_1, \mathbf{r}'_2, \cdots, \mathbf{r}'_N), \tag{923}$$

and then we proceed analogously with the other coordinates, so we obtain:

$$\Psi(\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_N) = \sum_{\nu_1, \cdots, \nu_N} c_{\nu_1, \cdots, \nu_N} \prod_{j=1}^N \psi_{\nu_j}(\mathbf{r}_j), \tag{924}$$

with

$$c_{\nu_1, \cdots, \nu_N} = \int d\mathbf{r}'_1 \psi_{\nu_1}(\mathbf{r}'_1)^* \cdots \int d\mathbf{r}'_N \psi_{\nu_N}(\mathbf{r}'_N)^* \Psi(\mathbf{r}'_1, \mathbf{r}'_2, \cdots, \mathbf{r}'_N). \tag{925}$$

While the product states $\prod_{j=1}^N \psi_{\nu_j}(\mathbf{r}_j)$ form a complete basis for the N-particle Hilbert space, they do not necessarily fulfill the indistinguishability requirement of bosons (or fermions) so they need to be symmetrized (or anti-symmetrized). Applying the bosonic symmetrization $\hat{S}_+$ (or the fermionic anti-symmetrization $\hat{S}_-$) operator, we obtain linear combinations of product states with the proper symmetry to describe systems of N-bosons (or fermions), according to the following normalized

permanents (or *Slater determinants*):

$$\hat{S}_{\pm} \prod_{j=1}^{N} \psi_{\nu_j}(\mathbf{r}_j) = \frac{1}{\prod_{\nu} \sqrt{n_{\nu}!}} \frac{1}{\sqrt{N!}} \begin{vmatrix} \psi_{\nu_1}(\mathbf{r}_1) & \psi_{\nu_1}(\mathbf{r}_2) & \cdots & \psi_{\nu_1}(\mathbf{r}_N) \\ \psi_{\nu_2}(\mathbf{r}_1) & \psi_{\nu_2}(\mathbf{r}_2) & \cdots & \psi_{\nu_2}(\mathbf{r}_N) \\ \cdots & \cdots & \cdots & \cdots \\ \psi_{\nu_N}(\mathbf{r}_1) & \psi_{\nu_N}(\mathbf{r}_2) & \cdots & \psi_{\nu_N}(\mathbf{r}_N) \end{vmatrix}_{\pm}, \tag{926}$$

$$= \langle \mathbf{r} | \psi_{\nu_1} \psi_{\nu_2} \cdots \psi_{\nu_N} \rangle,$$

which are linear combinations of product states corresponding to all possible permutation on the set of N coordinates. Each term of the Slater determinant has a sign $(-1)^p$, corresponding to the number of permutations p, while the bosonic permanent terms are all sing-less.

47.2   Occupation Number Basis

The product states, introduced by Eq. (926), are linear combinations of occupied single-particle states. The occupation number representation $|n_{\nu_1}, n_{\nu_2}, n_{\nu_3}, \cdots\rangle$, simply lists the number of particles n_{ν_j} in each occupied state ν_j, with $\sum_j n_{\nu_j} = N$. Such states are eigenstates of the number operators,

$$\hat{n}_{\nu_k} |n_{\nu_1}, n_{\nu_2}, n_{\nu_3}, \cdots\rangle = n_{\nu_k} |n_{\nu_1}, n_{\nu_2}, n_{\nu_3}, \cdots\rangle. \tag{927}$$

For fermions, $n_{\nu_k} = 0, 1$ while for bosons $n_{\nu_k} = 0, 1, 2, \cdots$ is a positive integer.

47.3   Creation and Anihilation Operators

Bosons: The creation and anihilation operators of bosons, $\hat{b}_j^{\dagger}$ and $\hat{b}_j$, are defined to ensure that the number operator $\hat{n}_{\nu_j} = \hat{b}_j^{\dagger} \hat{b}_j$ gives the number of bosons in state ν_j as follows:

$$\hat{n}_{\nu_j} |n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots\rangle = n_{\nu_j} |n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots\rangle, \tag{928}$$

and raise or lower the occupation of that state, as follows:

$$\hat{b}_j^{\dagger} |n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots\rangle = B_+(n_{\nu_j}) |n_{\nu_1}, n_{\nu_2}, \cdots (n_{\nu_j} + 1), \cdots\rangle,$$
$$\hat{b}_j |n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots\rangle = B_-(n_{\nu_j}) |n_{\nu_1}, n_{\nu_2}, \cdots (n_{\nu_j} - 1), \cdots\rangle, \tag{929}$$

where $B_+(n_{\nu_j})$ and $B_-(n_{\nu_j})$ are normalization constants. We further demand that the occupation number of an unoccupied state (e.g., $n_{\nu_j} = 0$) cannot be further reduced, which is equivalent to demand that $\hat{b}_j |n_{\nu_1}, n_{\nu_2}, \cdots 0, \cdots\rangle = 0$. Furthermore, we define the normalization constants $B_+(0) = 1$ and $B_-(1) = 1$ so that

$$\hat{b}_j^{\dagger} |n_{\nu_1}, n_{\nu_2}, \cdots 0, \cdots\rangle = |n_{\nu_1}, n_{\nu_2}, \cdots 1, \cdots\rangle,$$
$$\hat{b}_j |n_{\nu_1}, n_{\nu_2}, \cdots 1, \cdots\rangle = |n_{\nu_1}, n_{\nu_2}, \cdots 0, \cdots\rangle. \tag{930}$$

Therefore,

$$\hat{b}_j \hat{b}_j^{\dagger} |n_{\nu_1}, n_{\nu_2}, \cdots 0, \cdots\rangle = |n_{\nu_1}, n_{\nu_2}, \cdots 0, \cdots\rangle,$$
$$\hat{b}_j^{\dagger} \hat{b}_j |n_{\nu_1}, n_{\nu_2}, \cdots 0, \cdots\rangle = 0, \tag{931}$$

which can be summarized as $\hat{b}_j \hat{b}_j^\dagger = \hat{n}_{\nu_j} + 1$ and $[\hat{b}_j, \hat{b}_j^\dagger] = 1$. When $j \neq k$, however, $[\hat{b}_j, \hat{b}_k^\dagger] = 0$. The normalization constants for other states are found from Eq. (928), as follows:

$$\langle n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots | \hat{b}_j^\dagger \hat{b}_j | n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots \rangle = n_{\nu_j},$$

$$\langle n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots | \hat{b}_j^\dagger \hat{b}_j | n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots \rangle = B_-(n_{\nu_j})^2, \tag{932}$$

so $B_-(n_{\nu_j}) = \sqrt{n_{\nu_j}}$. Analogously, we obtain

$$\langle n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots | \hat{b}_j \hat{b}_j^\dagger | n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots \rangle = B_+(n_{\nu_j})^2,$$

$$(n_{\nu_j} + 1) = B_+(n_{\nu_j})^2, \tag{933}$$

$B_+(n_{\nu_j}) = \sqrt{n_{\nu_j} + 1}$. Therefore,

$$(\hat{b}_j^\dagger)^{n_\nu} | n_{\nu_1}, n_{\nu_2}, \cdots 0, \cdots \rangle = \sqrt{n_\nu!} | n_{\nu_1}, n_{\nu_2}, \cdots n_\nu, \cdots \rangle. \tag{934}$$

or

$$| n_{\nu_1}, n_{\nu_2}, n_{\nu_3}, \cdots \rangle = \prod_j \frac{(\hat{b}_j^\dagger)^{n_{\nu_j}}}{\sqrt{n_\nu!}} | 0, 0, 0, \cdots \rangle. \tag{935}$$

Fermions: The creation and anihilation operators of fermions, $\hat{c}_j^\dagger$ and $\hat{c}_j$, are defined to ensure that the number operator $\hat{n}_{\nu_j} = \hat{c}_j^\dagger \hat{c}_j$ gives the number of fermions $n_{\nu_j} = 0, 1$ in state ν_j. This requires that $\hat{c}_j^\dagger |1\rangle = 0$, $\hat{c}_j^\dagger |0\rangle = |1\rangle$, $\hat{c}_j |0\rangle = 0$, and $\hat{c}_j^\dagger |0\rangle = |1\rangle$. Therefore, $\hat{c}_j \hat{c}_j^\dagger |0\rangle = |0\rangle$ and $\hat{c}_j \hat{c}_j^\dagger |0\rangle = |0\rangle$, or $\hat{c}_j \hat{c}_j^\dagger + \hat{c}_j^\dagger \hat{c}_j = 0$.

47.4 Operators in Second Quantization

In this subsection we show that any single particle operator $\hat{A}$ can be expressed in terms of $\hat{b}_j^\dagger$ and $\hat{b}_j$, as follows: $\hat{A} = \sum_{\nu_j, \nu_k} A_{\nu_j, \nu_k} \hat{b}_j^\dagger \hat{b}_k$, with $A_{\nu_j, \nu_k} = \langle \nu_j | \hat{A} | \nu_k \rangle$. As an example of a single particle operator, we consider the kinetic energy $\hat{T} = \sum_{k=1}^N \hat{T}_k$, with $\hat{T}_k = \frac{\hat{p}_k^2}{2m_k}$:

$$\langle \mathbf{r} | \hat{T} | \psi_{\nu_1} \psi_{\nu_2} \cdots \psi_{\nu_N} \rangle = \sum_{\nu_j} \langle \mathbf{r} | \psi_{\nu_j} \rangle \langle \psi_{\nu_j} | \hat{T} | \psi_{\nu_1} \psi_{\nu_2} \cdots \psi_{\nu_N} \rangle$$

$$= \sum_{\nu_j} \langle \mathbf{r} | \psi_{\nu_j} \rangle \sum_{k=1}^N \langle \psi_{\nu_j} | \hat{T}_k | \psi_{\nu_1} \psi_{\nu_2} \cdots \psi_{\nu_N} \rangle$$

$$= \sum_{\nu_j} \langle \mathbf{r} | \psi_{\nu_j} \rangle \sum_{k=1}^N \langle \psi_{\nu_j} | \hat{T}_k | \psi_{\nu_k} \rangle \langle \mathbf{r} | \hat{b}_{\nu_k} | \psi_{\nu_1} \psi_{\nu_2} \cdots \psi_{\nu_N} \rangle \tag{936}$$

$$= \sum_{k=1}^N \sum_{\nu_j, \nu_l} \langle \mathbf{r} | \psi_{\nu_j} \rangle \delta_{\nu_l, \nu_k} T_{\nu_j, \nu_l} \langle \mathbf{r} | \hat{b}_{\nu_k} | \psi_{\nu_1} \psi_{\nu_2} \cdots \psi_{\nu_N} \rangle$$

$$= \sum_{k=1}^N \sum_{\nu_j, \nu_l} \delta_{\nu_l, \nu_k} T_{\nu_j, \nu_l} \langle \mathbf{r} | \hat{b}_{\nu_j}^\dagger \hat{b}_{\nu_k} | \psi_{\nu_1} \psi_{\nu_2} \cdots \psi_{\nu_N} \rangle$$

Therefore,

$$
\begin{aligned}
\hat{T}\left[\hat{b}^\dagger_{\nu_1} \cdots \hat{b}^\dagger_{\nu_N}|0\rangle\right] &= \sum_{k=1}^{N} \sum_{\nu_j,\nu_l} \delta_{\nu_l,\nu_k} T_{\nu_j,\nu_l} \hat{b}^\dagger_{\nu_j} \hat{b}_{\nu_k} \hat{b}^\dagger_{\nu_1} \cdots \hat{b}^\dagger_{\nu_N}|0\rangle \\
&= \sum_{k=1}^{N} \sum_{\nu_j,\nu_l} \delta_{\nu_l,\nu_k} T_{\nu_j,\nu_l} \hat{b}^\dagger_{\nu_j} \frac{\hat{n}_{\nu_k}}{n_{\nu_k}} \hat{b}_{\nu_k} \hat{b}^\dagger_{\nu_1} \cdots \hat{b}^\dagger_{\nu_N}|0\rangle \\
&= \sum_{k=1}^{N} \sum_{\nu_j,\nu_l} \delta_{\nu_l,\nu_k} T_{\nu_j,\nu_l} \frac{\hat{b}^\dagger_{\nu_j} \hat{b}_{\nu_k}}{n_{\nu_k}} \left[\hat{b}^\dagger_{\nu_k} \hat{b}_{\nu_k} \hat{b}^\dagger_{\nu_1} \cdots \hat{b}^\dagger_{\nu_N}|0\rangle\right] \\
&= \sum_{\nu_j,\nu_l} T_{\nu_j,\nu_l} \sum_{k=1}^{N} \delta_{\nu_l,\nu_k} \frac{\hat{b}^\dagger_{\nu_j} \hat{b}_{\nu_k}}{n_{\nu_k}} \left[\hat{b}^\dagger_{\nu_1} \cdots \hat{b}^\dagger_{\nu_N}|0\rangle\right] \qquad (937) \\
&= \sum_{\nu_j,\nu_l} T_{\nu_j,\nu_l} \hat{b}^\dagger_{\nu_j} \hat{b}_{\nu_l} \sum_{k=1}^{N} \frac{\delta_{\nu_l,\nu_k}}{n_{\nu_k}} \left[\hat{b}^\dagger_{\nu_1} \cdots \hat{b}^\dagger_{\nu_N}|0\rangle\right] \\
&= \sum_{\nu_j,\nu_l} T_{\nu_j,\nu_l} \hat{b}^\dagger_{\nu_j} \hat{b}_{\nu_l} \frac{1}{n_{\nu_l}} \sum_{k=1}^{N} \delta_{\nu_l,\nu_k} \left[\hat{b}^\dagger_{\nu_1} \cdots \hat{b}^\dagger_{\nu_N}|0\rangle\right] \\
&= \sum_{\nu_j,\nu_l} T_{\nu_j,\nu_l} \hat{b}^\dagger_{\nu_j} \hat{b}_{\nu_l} \left[\hat{b}^\dagger_{\nu_1} \cdots \hat{b}^\dagger_{\nu_N}|0\rangle\right]
\end{aligned}
$$

where p is the number of particles in state state ψ_{ν_k} for the N-particle system described by state $|\psi_{\nu_1}\psi_{\nu_2}\cdots\psi_{\nu_N}\rangle = \hat{b}^\dagger_{\nu_1} \cdots \hat{b}^\dagger_{\nu_N}|0\rangle$, so according to Eq. (933), $\langle\psi_{\nu_1}\psi_{\nu_2}\cdots\psi_{\nu_N}|\hat{b}_{\nu_k}\hat{b}^\dagger_{\nu_k}|\psi_{\nu_1}\psi_{\nu_2}\cdots\psi_{\nu_N}\rangle = (n_{\nu_k}+1)$. Therefore,

$$
\hat{T} = \sum_{\nu_j,\nu_l} T_{\nu_j,\nu_l} \hat{b}^\dagger_{\nu_j} \hat{b}_{\nu_l}. \qquad (938)
$$

Analogously, any 2-particle operator $\hat{V}$ such as the pair-wise additive potential,

$$
\hat{V} = \frac{1}{2} \sum_{j=1}^{N} \sum_{k \neq j} V(x_j, x_k), \qquad (939)
$$

can be written in second quantization, as follows:

$$
\hat{V} = \sum_{\nu_j,\nu_i,\nu_l,\nu_k} V_{\nu_j,\nu_i,\nu_l,\nu_k} \hat{b}^\dagger_{\nu_j} \hat{b}^\dagger_{\nu_i} \hat{b}_{\nu_l} \hat{b}_{\nu_k} \qquad (940)
$$

where $V_{\nu_j,\nu_i,\nu_l,\nu_k} = \langle\psi_{\nu_j}\psi_{\nu_i}|V(x_1,x_2)|\psi_{\nu_l}\psi_{\nu_k}\rangle$.

47.5   Change of basis in Second Quantization

We consider two different complete and ordered single-particle basis sets $\{|\psi_{\nu_j}\rangle\}$ and $\{|\psi_{\mu_j}\rangle\}$ with $j = 1$–N. Using the completeness relationship we can write any element of one basis set as a linear combination of elements of the other basis set, as follows:

$$|\psi_{\mu_j}\rangle = \sum_k |\psi_{\nu_k}\rangle\langle\psi_{\nu_k}|\psi_{\mu_j}\rangle, \tag{941}$$

where $\psi_{\nu_k}\rangle = \hat{a}^\dagger_{\nu_k}|0\rangle$ and $\psi_{\mu_j}\rangle = \hat{a}^\dagger_{\mu_j}|0\rangle$. Therefore,

$$\hat{a}_{\mu_j}|0\rangle = \sum_k \langle\psi_{\nu_k}|\psi_{\mu_j}\rangle\hat{a}_{\nu_k}|0\rangle, \tag{942}$$

or

$$\hat{a}_{\mu_j} = \sum_k \langle\psi_{\nu_k}|\psi_{\mu_j}\rangle\hat{a}_{\nu_k}, \tag{943}$$

and

$$\hat{a}^\dagger_{\mu_j} = \sum_k \langle\psi_{\nu_k}|\psi_{\mu_j}\rangle^*\hat{a}^\dagger_{\nu_k}, \tag{944}$$

47.6   Mapping into Cartesian Coordinates

Introducing the Cartesian operators $\tilde{x}_{\nu_j} = \frac{1}{\sqrt{2}}[\hat{b}^\dagger_{\nu_j} + \hat{b}_{\nu_j}]$ and $\tilde{p}_{\nu_j} = \frac{i}{\sqrt{2}}[\hat{b}^\dagger_{\nu_j} - \hat{b}_{\nu_j}]$, with $[\tilde{x}_{\nu_j}, \tilde{p}_{\nu_j}] = i$, since $\tilde{x}_{\nu_j} = \hat{x}\sqrt{\frac{m\omega}{\hbar}}$, $\tilde{p}_{\nu_j} = \hat{p}/\sqrt{m\omega\hbar}$ and $[\hat{x}_{\nu_j}, \hat{p}_{\nu_j}] = i\hbar$, for the harmonic oscillator Hamiltonian

$$\begin{aligned} H &= \frac{\hat{p}_{\nu_j}^2}{2m} + \frac{1}{2}m\omega^2\hat{x}^2, \\ &= \frac{\tilde{p}_{\nu_j}^2}{2m}m\omega\hbar + \frac{1}{2}m\omega^2\frac{\hbar}{m\omega}\tilde{x}_{\nu_j}^2, \\ &= \frac{\hbar\omega}{2}\left[\tilde{p}_{\nu_j}^2 + \tilde{x}_{\nu_j}^2\right]. \end{aligned} \tag{945}$$

Considering that

$$\begin{aligned} \hat{n}_{\nu_j} &= \hat{b}^\dagger_{\nu_j}\hat{b}_{\nu_j}, \\ \hat{b}^\dagger_{\nu_j} &= \frac{1}{\sqrt{2}}\left[\tilde{x}_{\nu_j} - i\tilde{p}_{\nu_j}\right], \\ \hat{b}_{\nu_j} &= \frac{1}{\sqrt{2}}\left[\tilde{x}_{\nu_j} + i\tilde{p}_{\nu_j}\right]. \end{aligned} \tag{946}$$

we obtain,

$$\begin{aligned} \hat{n}_{\nu_j} &= \frac{1}{2}(\tilde{x}_{\nu_j} - i\tilde{p}_{\nu_j})(\tilde{x}_{\nu_j} + i\tilde{p}_{\nu_j}) \\ &= \frac{1}{2}(\tilde{x}_{\nu_j}^2 + i[\tilde{x}_{\nu_j}, \tilde{p}_{\nu_j}] + \tilde{p}_{\nu_j}^2) \\ &= \frac{1}{2}(\tilde{x}_{\nu_j}^2 + \tilde{p}_{\nu_j}^2 - 1) \end{aligned} \tag{947}$$

and

$$H = \hbar\omega\left(\hat{n}_{\nu_j} + \frac{1}{2}\right).$$ (948)

Substituting the Cartesian expressions of b_{ν_j} and $b_{\nu_j}^\dagger$ into Eq. (938), we obtain:

$$
\begin{aligned}
\hat{T} &= \frac{1}{2}\sum_{\nu_j,\nu_l} T_{\nu_j,\nu_l}\left[\tilde{x}_{\nu_j} - i\tilde{p}_{\nu_j}\right]\left[\tilde{x}_{\nu_l} + i\tilde{p}_{\nu_l}\right], \\
&= \frac{1}{2}\sum_{\nu_j} T_{\nu_j,\nu_j}(\tilde{x}_{\nu_j}^2 + \tilde{p}_{\nu_j}^2 - 1) + \frac{1}{2}\sum_{\nu_j}\sum_{\nu_l\neq\nu_j} T_{\nu_j,\nu_l}\left[\tilde{x}_{\nu_j} - i\tilde{p}_{\nu_j}\right]\left[\tilde{x}_{\nu_l} + i\tilde{p}_{\nu_l}\right] \\
&= \frac{1}{2}\sum_{\nu_j} T_{\nu_j,\nu_j}(\tilde{x}_{\nu_j}^2 + \tilde{p}_{\nu_j}^2 - 1) + \frac{1}{2}\sum_{\nu_j}\sum_{\nu_l\neq\nu_j} T_{\nu_j,\nu_l}\left[\tilde{x}_{\nu_j}\tilde{x}_{\nu_l} + \tilde{p}_{\nu_j}\tilde{p}_{\nu_l}\right]
\end{aligned}
$$ (949)

since $[\tilde{x}_{\nu_j}, \tilde{p}_{\nu_l}] = i\delta_{\nu_l,\nu_j}$ while $[\tilde{x}_{\nu_j}, \tilde{x}_{\nu_l}] = 0$ and $[\tilde{p}_{\nu_j}, \tilde{p}_{\nu_l}] = 0$.

48   Spin-Boson Model

Consider the spin-boson Hamiltonian

$$\hat{H} = \frac{2}{\hbar}\left[\epsilon\hat{S}_z + J\hat{S}_x + \hat{S}_z \times f(\mathbf{y})\right] + H_b(\mathbf{y}, \mathbf{p}_y),$$ (950)

with $f(\mathbf{y}) = \sum_{i=1}^N c_i y_i$, and $H_b(\mathbf{y}, \mathbf{p}_y) = \sum_{i=1}^N p_i^2/(2m_i) + 1/2 m_i\omega_i y_i^2$. For example, for a spin $S = 1/2$, $\hat{H}$ can be written as follows:

$$H = \begin{bmatrix} \epsilon & 0 \\ 0 & -\epsilon \end{bmatrix} + \begin{bmatrix} 0 & J \\ J & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \times f(\mathbf{y}) + H_b(\mathbf{y}, \mathbf{p}_y),$$ (951)

which is written in the Zeeman basis set of $2S + 1$ eigenstates of $\hat{S}_z$ with eigenvalues $m_s\hbar$ ranging from $-\hbar S$ to $\hbar S$, as defined by the eigenvalue problem:

$$\hat{S}_z|S, m_s\rangle = \hbar m_s|S, m_s\rangle.$$ (952)

The raising $\hat{S}^+ = (\hat{S}_x + i\hat{S}_y)$, and lowering $\hat{S}^- = (\hat{S}_x - i\hat{S}_y)$ operators satisfy the commutation relations: $[\hat{S}_z, \hat{S}^+] = \hbar\hat{S}^+$, $[\hat{S}_z, \hat{S}^-] = -\hbar\hat{S}^-$, $[\hat{S}^+, \hat{S}^-] = \hbar 2\hat{S}_z$.

Matrix elements in the Zeeman basis are defined, as usual:

$$
\begin{aligned}
\langle m_s'|\hat{S}_x|m_s\rangle &= \hbar(\delta_{m_s',m_s+1} + \delta_{m_s'+1,m_s})\frac{1}{2}\sqrt{S(S+1) - m_s'm_s} \\
\langle m_s'|\hat{S}_y|m_s\rangle &= \hbar(\delta_{m_s',m_s+1} - \delta_{m_s'+1,m_s})\frac{1}{2i}\sqrt{S(S+1) - m_s'm_s} \\
\langle m_s'|\hat{S}_z|m_s\rangle &= \hbar\delta_{m_s',m_s}m_s \\
\langle m_s'|\hat{S}^+|m_s\rangle &= \hbar\delta_{m_s',m_s+1}\sqrt{S(S+1) - m_s'm_s} \\
\langle m_s'|\hat{S}^-|m_s\rangle &= \hbar\delta_{m_s'+1,m_s}\sqrt{S(S+1) - m_s'm_s} \\
\langle m_s'|\hat{S}^2|m_s\rangle &= \hbar\delta_{m_s',m_s}S(S+1)
\end{aligned}
$$ (953)

so the same Hamiltonian introduced by Eq. (950) can be used to model a spin-boson model with an arbitrary large number of states, as for example for 6 energy levels (S=5/2) with

$$
S^+ = \hbar \begin{bmatrix} 0 & \sqrt{5} & 0 & 0 & 0 & 0 \\ 0 & 0 & \sqrt{8} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sqrt{9} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{8} & 0 \\ 0 & 0 & 0 & 0 & 0 & \sqrt{5} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},
$$

$$
S^- = \hbar \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \sqrt{5} & 0 & 0 & 0 & 0 & 0 \\ 0 & \sqrt{8} & 0 & 0 & 0 & 0 \\ 0 & 0 & \sqrt{9} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sqrt{8} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{5} & 0 \end{bmatrix}
$$

(954)

and

$$
S_x = \frac{\hbar}{2} \begin{bmatrix} 0 & \sqrt{5} & 0 & 0 & 0 & 0 \\ \sqrt{5} & 0 & \sqrt{8} & 0 & 0 & 0 \\ 0 & \sqrt{8} & 0 & \sqrt{9} & 0 & 0 \\ 0 & 0 & \sqrt{9} & 0 & \sqrt{8} & 0 \\ 0 & 0 & 0 & \sqrt{8} & 0 & \sqrt{5} \\ 0 & 0 & 0 & 0 & \sqrt{5} & 0 \end{bmatrix},
$$

$$
S_y = \frac{\hbar}{2i} \begin{bmatrix} 0 & \sqrt{5} & 0 & 0 & 0 & 0 \\ -\sqrt{5} & 0 & \sqrt{8} & 0 & 0 & 0 \\ 0 & -\sqrt{8} & 0 & \sqrt{9} & 0 & 0 \\ 0 & 0 & -\sqrt{9} & 0 & \sqrt{8} & 0 \\ 0 & 0 & 0 & -\sqrt{8} & 0 & \sqrt{5} \\ 0 & 0 & 0 & 0 & -\sqrt{5} & 0 \end{bmatrix},
$$

(955)

$$
S_z = \frac{\hbar}{2} \begin{bmatrix} 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -3 & 0 \\ 0 & 0 & 0 & 0 & 0 & -5 \end{bmatrix},
$$

According to the second quantization mapping, the spin-boson Hamiltonian can be written as follows:

$$
\hat{H} = H_b(\mathbf{y}, \mathbf{p}_y) + \sum_{m_s, m_s'} \langle m_s' | (\epsilon + f(\mathbf{y})) \frac{2}{\hbar} \hat{S}_z + J \frac{2}{\hbar} \hat{S}_x | m_s \rangle \hat{a}_{m_s'}^\dagger \hat{a}_{m_s}.
$$

(956)

205

and in Cartesian coordinates,

$$\hat{H} = H_b(\mathbf{y}, \mathbf{p}_y) + \sum_{m_s} \frac{2}{\hbar}(\epsilon + f(\mathbf{y}))\langle m_s|\hat{S}_z|m_s\rangle[\tilde{x}_{m_s}^2 + \tilde{p}_{m_s}^2 - 1]$$
$$+ \sum_{m_s} \sum_{m_s' \neq m_s} \frac{2}{\hbar}\langle m_s|\hat{S}_x|m_s'\rangle J \left[\tilde{x}_{m_s}\tilde{x}_{m_s'} + \tilde{p}_{m_s}\tilde{p}_{m_s'}\right] \tag{957}$$

49   Holstein-Primakoff Mapping

According to the Holstein-Primakoff (HP) transformation, we re-write the spin operators in terms of boson operators, as follows:

$$\hat{S}^+ = \hbar\sqrt{2S - \hat{N}}\hat{a}, \tag{958}$$

and

$$\hat{S}^- = \hat{a}^\dagger \hbar\sqrt{2S - \hat{N}}, \tag{959}$$

where $\hat{N} = \hat{a}^\dagger\hat{a}$ is the usual number operator counting the number of bosons, as defined in terms of the creation $\hat{a}^\dagger$ and annihilation $\hat{a}$ operators satisfying $[\hat{a}, \hat{a}^\dagger] = 1$.

We note that $\hat{S}_z = \frac{1}{2\hbar}[\hat{S}^+, \hat{S}^-] = \frac{\hbar}{2}(\sqrt{2S - \hat{N}}\hat{a}\hat{a}^\dagger\sqrt{2S - \hat{N}} - \hat{a}^\dagger\sqrt{2S - \hat{N}}\sqrt{2S - \hat{N}}\hat{a})$ since $\hat{S}_z = \frac{\hbar}{2}(\sqrt{2S - \hat{N}}(1 + \hat{N})\sqrt{2S - \hat{N}} - \hat{a}^\dagger(2S - \hat{N})\hat{a}) = \frac{\hbar}{2}((2S - \hat{N})(1 + \hat{N}) - \hat{a}^\dagger\hat{a}2S + \hat{a}^\dagger\hat{N}\hat{a}) = \frac{\hbar}{2}(2S - \hat{N} + 2S\hat{N} - \hat{N}^2 - \hat{N}2S + \hat{a}^\dagger\hat{N}\hat{a}) = \frac{\hbar}{2}(2S - \hat{N} + 2S\hat{N} - \hat{N}^2 - \hat{N}2S + \hat{N}^2 - \hat{N}) = \hbar(S - \hat{N})$. Therefore,

$$\hat{S}_z = \hbar(S - \hat{N}), \tag{960}$$

and

$$\hat{S}_x = \frac{1}{2}\left[\hat{S}^+ + \hat{S}^-\right] = \frac{\hbar}{2}\left[\sqrt{2S - \hat{N}}\hat{a} + \hat{a}^\dagger\sqrt{2S - \hat{N}}\right], \tag{961}$$

The HP transformation corresponds to a change in basis, mapping the basis set of eigenstates of S_z into the basis of the number operator $\hat{N}$, limited to a range of eigenvalues determined by S.

Introducing the operators $\tilde{x} = \frac{1}{\sqrt{2}}[\hat{a}^\dagger + \hat{a}]$ and $\tilde{p} = \frac{i}{\sqrt{2}}[\hat{a}^\dagger - \hat{a}]$, we obtain:

$$\tilde{x} - i\tilde{p} = \sqrt{2}\hat{a}^\dagger, \tag{962}$$

and

$$\tilde{x} + i\tilde{p} = \sqrt{2}\hat{a}. \tag{963}$$

Therefore,

$$\hat{N} = \frac{1}{2}(\tilde{x} - i\tilde{p})(\tilde{x} + i\tilde{p})$$
$$= \frac{1}{2}(\tilde{x}^2 + i[\tilde{x}, \tilde{p}] + \tilde{p}^2) \tag{964}$$
$$= \frac{1}{2}(\tilde{x}^2 + \tilde{p}^2 - 1)$$

Therefore, $S_z = \hbar(S - \frac{1}{2}(\tilde{x}^2 + \tilde{p}^2 - 1))$. Analogously, we obtain:

$$\hat{S}_x = \frac{\hbar}{2}\left[\sqrt{S - \frac{1}{4}(\tilde{x}^2 + \tilde{p}^2 - 1)}(\tilde{x} + i\tilde{p}) + (\tilde{x} - i\tilde{p})\sqrt{S - \frac{1}{4}(\tilde{x}^2 + \tilde{p}^2 - 1)}\right]. \tag{965}$$

and

$$\hat{H} = H_b(\mathbf{y}, \mathbf{p}_y) + (\epsilon + f(\mathbf{y}))(2S - (\tilde{x}^2 + \tilde{p}^2 - 1))$$
$$+ J\left[\sqrt{S - \frac{1}{4}(\tilde{x}^2 + \tilde{p}^2 - 1)}(\tilde{x} + i\tilde{p}) + (\tilde{x} - i\tilde{p})\sqrt{S - \frac{1}{4}(\tilde{x}^2 + \tilde{p}^2 - 1)}\right] \tag{966}$$

which in the classical limit (i.e., with $\left[\sqrt{S - \hat{N}/2}, \tilde{p}\right] \approx 0$, and $\left[\sqrt{S - \hat{N}/2}, \tilde{x}\right] \approx 0)$), gives:

$$\hat{H} = H_b(\mathbf{y}, \mathbf{p}_y) + (\epsilon + f(\mathbf{y}))(2S - (\tilde{x}^2 + \tilde{p}^2 - 1)) + J\tilde{x}\sqrt{4S - (\tilde{x}^2 + \tilde{p}^2 - 1)} \tag{967}$$

In the limit of $S >> 1$, we can approximate the raising and lowering operators, as follows: $S^\dagger = \hbar\hat{a}\sqrt{2S}$ and $S^- = \hbar\hat{a}^\dagger\sqrt{2S}$, since the square roots can be expanded in Taylor series, as follows:

$$\sqrt{S - \frac{\hat{N}}{2}} = \sqrt{S} + \frac{1}{2}(2S)^{-1/2}(-\frac{\hat{N}}{2}) + \cdots$$
$$\approx \sqrt{S} \tag{968}$$

Therefore, truncating the expansion introduced by Eq. (968) after the first term and substituting into Eq. (966), we obtain the 'simple spin-wave (SW) theory' expression:

$$\hat{H} = H_b(\mathbf{y}, \mathbf{p}_y) + (\epsilon + f(\mathbf{y}))(2S - (\tilde{x}^2 + \tilde{p}^2 - 1)) + 2J\sqrt{S}\tilde{x} \tag{969}$$

50   Schwinger Mapping

Spin operators can also be mapped according to the Schwinger-boson representation, in terms of pairs of (constrained) bosons, as follows:

$$\begin{aligned}
\hat{S}^+ &= \hbar\hat{b}^\dagger\hat{a}, \\
\hat{S}^- &= \hbar\hat{a}^\dagger\hat{b}, \\
\hat{S}_z &= \frac{1}{2}\left[\hat{S}^+, \hat{S}^-\right] = \frac{\hbar}{2}\left(\hat{b}^\dagger\hat{b} - \hat{a}^\dagger\hat{a}\right), \\
\hat{S}_x &= \frac{1}{2}\left(\hat{S}^+ + \hat{S}^-\right) = \frac{\hbar}{2}\left(\hat{b}^\dagger\hat{a} + \hat{a}^\dagger\hat{b}\right), \\
\hat{S}_y &= \frac{1}{2i}\left(\hat{S}^+ - \hat{S}^-\right) = \frac{\hbar}{2i}\left(\hat{b}^\dagger\hat{a} - \hat{a}^\dagger\hat{b}\right),
\end{aligned} \tag{970}$$

satisfying the usual commutation relations, $[\hat{S}_x, \hat{S}_y] = i\hbar\hat{S}_z$. In addition $\hat{S}^2|\chi_+\rangle = \hbar^2 S(S+1)|\chi_+\rangle$, which establishes the constraint

$$S = \frac{1}{2}(\hat{b}^\dagger\hat{b} + \hat{a}^\dagger\hat{a}). \tag{971}$$

Substituting the spin operators in Eq. (950), according to the Schwinger mapping, introduced by Eqs. (970), we obtain:

$$\hat{H} = (\epsilon + f(\mathbf{y}))\left(\hat{b}^\dagger\hat{b} - \hat{a}^\dagger\hat{a}\right) + J\left(\hat{b}^\dagger\hat{a} + \hat{a}^\dagger\hat{b}\right) + H_b(\mathbf{y}, \mathbf{p}_y), \tag{972}$$

Changing variables, $\hat{b} = \hat{b}_{\nu_1}$ and $\hat{a} = \hat{b}_{\nu_2}$, defined in terms of the Cartesian coordinates, as follows:

$$\hat{b}^\dagger_{\nu_j} = \frac{1}{\sqrt{2}}\left[\tilde{x}_{\nu_j} - i\tilde{p}_{\nu_j}\right], \tag{973}$$

and

$$\hat{b}_{\nu_j} = \frac{1}{\sqrt{2}}\left[\tilde{x}_{\nu_j} + i\tilde{p}_{\nu_j}\right], \tag{974}$$

with

$$\hat{n}_{\nu_j} = \frac{1}{2}(\tilde{x}^2_{\nu_j} + \tilde{p}^2_{\nu_j} - 1) \tag{975}$$

we obtain

$$\hat{H} = H_b(\mathbf{y}, \mathbf{p}_y) + (\epsilon + f(\mathbf{y}))\left[(\tilde{x}^2_{\nu_1} + \tilde{p}^2_{\nu_1} - 1) - (\tilde{x}^2_{\nu_2} + \tilde{p}^2_{\nu_2} - 1)\right] + J\left(\tilde{x}_{\nu_1}\tilde{x}_{\nu_2} + \tilde{p}_{\nu_1}\tilde{p}_{\nu_2}\right), \tag{976}$$

which is the same Hamiltonian given by Eq. (957), according to the second quantization mapping, although with the contraint

$$S = \frac{1}{4}\left[(\tilde{x}^2_{\nu_1} + \tilde{p}^2_{\nu_1} - 1) + (\tilde{x}^2_{\nu_2} + \tilde{p}^2_{\nu_2} - 1)\right]. \tag{977}$$

51   Single Boson Hamiltonian

We consider writing an arbitrary Hamiltonian with matrix elements H_{jk}, where $j, k = 1$–n, in the basis set of a single Harmonic oscillator. As an example, we consider the case of a 6×6 Hamiltonian (*i.e.*, $n = 6$).

First, we introduce the matrices for the creation and anihilation operators $a^\dagger$ and a, representing the first 10 states ($\nu = 0$–9), with

$$|0\rangle = \left(\frac{m\omega}{\pi\hbar}\right)^{1/4}\exp(-\frac{m\omega}{2\hbar}x^2), \text{ and}$$
$$|\nu + 1\rangle = \frac{1}{\sqrt{\nu + 1}}\hat{a}^\dagger|\nu\rangle, \tag{978}$$

with matrix elements $\langle\mu|a^\dagger_j|\nu\rangle = \sqrt{\nu + 1}\delta_{\mu,\nu+1}$ and $\langle\nu|a|\mu\rangle = \langle\mu|a^\dagger|\nu\rangle$.

$$a^\dagger = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sqrt{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sqrt{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{5} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sqrt{6} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sqrt{7} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2\sqrt{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \end{pmatrix}$$

$$a = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sqrt{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sqrt{3} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sqrt{5} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sqrt{6} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sqrt{7} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2\sqrt{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

we obtain the number operator $\hat{N} = \hat{a}^\dagger \hat{a}$ matrix elements, $N_{jk} = \delta_{j,k}(j-1)$:

$$
N = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 7 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9
\end{pmatrix}
$$

Introducing the operator $\hat{\Gamma} = \left(5 - \hat{N}\right)\hat{a} = 5\hat{a} - \hat{a}^\dagger \hat{a}^2$, with matrix:

$$
\Gamma = \begin{pmatrix}
0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 4\sqrt{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 3\sqrt{3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \sqrt{5} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -\sqrt{7} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4\sqrt{2} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -9 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
$$

we find that $\hat{P}_{1,6} = |1\rangle\langle 6| = \frac{1}{5!\sqrt{5!}}\hat{\Gamma}^5 = \frac{1}{5!\sqrt{5!}}(5\hat{a} - \hat{a}^\dagger \hat{a}^2)^5$, with matrix

$$P_{1,6} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Operators $\hat{P}_{n,m}$, with $n = 2$–6, are obtained by the recursive relation: $\hat{P}_{n,m} = \frac{1}{\sqrt{n-1}}\hat{a}^\dagger \hat{P}_{n-1,m}$. The other matrix elements of the first row are obtained from the operators

$$\hat{P}_{1,5} = \frac{1}{5!\sqrt{4!}}\hat{\Gamma}^4 - \frac{1}{\sqrt{5}}\hat{P}_{2,6},$$

$$= \frac{1}{5!\sqrt{4!}}(5\hat{a} - \hat{a}^\dagger\hat{a}^2)^4 - \frac{1}{\sqrt{5}}\hat{a}^\dagger\frac{1}{5!\sqrt{5!}}(5\hat{a} - \hat{a}^\dagger\hat{a}^2)^5$$

$$\hat{P}_{1,4} = \Gamma^3\frac{2}{5!\sqrt{3!}} - \hat{P}_{3,6}\frac{1}{\sqrt{10}} - \hat{P}_{2,5}\frac{4}{5},$$

$$= (5a - a^\dagger a^2)^3\frac{2}{5!\sqrt{3!}} - \frac{1}{\sqrt{2}}(a^\dagger)^2\frac{1}{5!\sqrt{5!}}(5a - a^\dagger a^2)^5\frac{1}{\sqrt{10}} - \hat{a}^\dagger\hat{P}_{1,5}\frac{4}{5},$$

$$\hat{P}_{1,3} = \Gamma^2/(20\sqrt{2}) - \hat{P}_{4,6}/\sqrt{10} - \hat{P}_{3,5}3\sqrt{3}/(5\sqrt{2}) - \hat{P}_{2,4}3\sqrt{3}/5,$$

$$\hat{P}_{1,2} = \Gamma/5 - \hat{P}_{5,6}/\sqrt{5} - 4\hat{P}_{4,5}/5 - 3\sqrt{3}\hat{P}_{3,4}/5 - 4\sqrt{2}\hat{P}_{2,3}/5,$$

$$\hat{P}_{1,1} = \hat{I} - \hat{N}/5 - 4\hat{P}_{2,2}/5 - 3\hat{P}_{3,3}/5 - 2\hat{P}_{4,4}/5 - \hat{P}_{5,5}/5.$$

(979)

The expressions introduced by Eq. (979) can be obtained by inspection of the matrix representation of the corresponding operators. For example, the expression of $\hat{P}_{1,5}$ can be obtained by noticing that

$$\frac{1}{5!\sqrt{4!}}\Gamma^4 = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{5}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Analogously, one can obtain the expressions $\hat{P}_{1,4}$, $\hat{P}_{1,3}$, $\hat{P}_{1,2}$ and $\hat{P}_{1,1}$ and with those the other matrix elements.

Therefore, the example above illustrates how to write any 6×6 Hamiltonian, as an operator that can be evaluated in the basis of a single Harmonic oscillator, according to the following expression:

$$\hat{H} = \sum_{j,k=1}^{N} H_{jk} P_{j,k}(\hat{a}, \hat{a}^\dagger), \tag{980}$$

with $N = 6$, where $\hat{P}_{j,k}$ are functions of powers of $\hat{a}$ and $\hat{a}^\dagger$, as defined above. For comparison, note that such a Hamiltonian can also be written according to the second quantization representation, as follows:

$$\hat{H} = \sum_{j,k=1}^{N} H_{jk} \hat{a}_j^\dagger \hat{a}_k, \tag{981}$$

51.1 Two-level Systems

Analogously, introducing the operator $\hat{\Gamma} = (1 - \hat{N})\,\hat{a} = \hat{a} - \hat{a}^\dagger \hat{a}^2$, we find that any 2×2 Hamiltonian can be written, as follows:

$$\hat{H} = \sum_{j,k=1}^{2} H_{jk} \hat{P}_{j,k}, \tag{982}$$

212

where

$$\begin{aligned}
\hat{P}_{1,2} &= a - a^\dagger a^2, \\
\hat{P}_{2,2} &= a^\dagger a - (a^\dagger)^2 a^2, \\
\hat{P}_{1,1} &= 1 - a^\dagger a, \\
\hat{P}_{2,1} &= a^\dagger - (a^\dagger)^2 a.
\end{aligned} \tag{983}$$

Therefore,

$$\begin{aligned}
\hat{H} &= H_{11}(1 - a^\dagger a) + H_{12}(a - a^\dagger a^2) + H_{21}(a^\dagger - (a^\dagger)^2 a) + H_{22}(a^\dagger a - (a^\dagger)^2 a^2), \\
&= H_{11}(1 - \hat{N}) + H_{12}(\hat{a} - \hat{N}\hat{a}) + H_{21}(\hat{a}^\dagger - a^\dagger \hat{N}) + H_{22}(\hat{N} - (\hat{a}^\dagger \hat{N}\hat{a})), \\
&= H_{11}(1 - \hat{N}) + H_{12}(1 - \hat{N})\hat{a} + H_{21}\hat{a}^\dagger(1 - \hat{N}) + H_{22}\hat{N}(2 - \hat{N}).
\end{aligned} \tag{984}$$

When $H_{12} = H_{21}$,

$$\hat{H} = H_{11}(1 - \hat{N}) + H_{12}((1 - \hat{N})\hat{a} + \hat{a}^\dagger(1 - \hat{N})) + H_{22}\hat{N}(2 - \hat{N}). \tag{985}$$

Introducing the Cartesian operators,

$$\begin{aligned}
\hat{a}^\dagger &= \frac{1}{\sqrt{2}}(\tilde{x} - i\tilde{p}), \\
\hat{a} &= \frac{1}{\sqrt{2}}(\tilde{x} + i\tilde{p}), \\
\hat{N} &= \frac{1}{2}(\tilde{x}^2 + \tilde{p}^2 - 1),
\end{aligned} \tag{986}$$

we obtain

$$\begin{aligned}
(1 - \hat{N}) &= 1 - \frac{1}{2}(\tilde{x}^2 + \tilde{p}^2 - 1), \\
&= \frac{3}{2} - \frac{1}{2}(\tilde{x}^2 + \tilde{p}^2), \\
(1 - \hat{N})\hat{a} &= \left(\frac{3}{2} - \frac{1}{2}(\tilde{x}^2 + \tilde{p}^2)\right)\frac{1}{\sqrt{2}}(\tilde{x} + i\tilde{p}), \\
&= \frac{3}{2\sqrt{2}}(\tilde{x} + i\tilde{p}) - \frac{1}{2\sqrt{2}}(\tilde{x}^2 + \tilde{p}^2)(\tilde{x} + i\tilde{p}), \\
&= \frac{3}{2\sqrt{2}}(\tilde{x} + i\tilde{p}) - \frac{1}{2\sqrt{2}}(\tilde{x}^3 + \tilde{p}^2\tilde{x} + i\tilde{x}^2\tilde{p} + i\tilde{p}^3), \\
\hat{a}^\dagger(1 - \hat{N}) &= \frac{1}{\sqrt{2}}(\tilde{x} - i\tilde{p})\left(\frac{3}{2} - \frac{1}{2}(\tilde{x}^2 + \tilde{p}^2)\right), \\
&= \frac{3}{2\sqrt{2}}(\tilde{x} - i\tilde{p}) - \frac{1}{2\sqrt{2}}(\tilde{x} - i\tilde{p})(\tilde{x}^2 + \tilde{p}^2), \\
&= \frac{3}{2\sqrt{2}}(\tilde{x} - i\tilde{p}) - \frac{1}{2\sqrt{2}}(\tilde{x}^3 + \tilde{x}\tilde{p}^2 - i\tilde{p}\tilde{x}^2 - i\tilde{p}^3).
\end{aligned} \tag{987}$$

213

Therefore,

$$
\begin{aligned}
(1 - \hat{N})\hat{a} + \hat{a}^{\dagger}(1 - \hat{N}) &= \frac{3}{\sqrt{2}}\tilde{x} - \frac{1}{\sqrt{2}}(\tilde{x}^3 + \frac{1}{2}(\tilde{p}^2\tilde{x} + \tilde{x}\tilde{p}^2) + \frac{i}{2}(\tilde{x}^2\tilde{p} - \tilde{p}\tilde{x}^2)), \\
&= \frac{3}{\sqrt{2}}\tilde{x} - \frac{1}{\sqrt{2}}(\tilde{x}^3 + \frac{1}{2}(\tilde{p}^2\tilde{x} + \tilde{x}\tilde{p}^2) - \tilde{x}), \\
&= \frac{1}{\sqrt{2}}(4\tilde{x} - \tilde{x}^3) - \frac{1}{2\sqrt{2}}(\tilde{p}^2\tilde{x} + \tilde{x}\tilde{p}^2), \\
&= \frac{1}{\sqrt{2}}(4\tilde{x} - \tilde{x}^3) + \frac{1}{\sqrt{2}}(i\tilde{p} - \tilde{x}\tilde{p}^2), \\
(2 - \hat{N}) &= 2 - \frac{1}{2}(\tilde{x}^2 + \tilde{p}^2 - 1), \\
&= \frac{5}{2} - \frac{1}{2}(\tilde{x}^2 + \tilde{p}^2), \\
\hat{N}(2 - \hat{N}) &= \left(\frac{1}{2}(\tilde{x}^2 + \tilde{p}^2 - 1)\right)\left(2 - \frac{1}{2}(\tilde{x}^2 + \tilde{p}^2 - 1)\right), \\
&= \left(\frac{1}{2}(\tilde{x}^2 + \tilde{p}^2) - \frac{1}{2}\right)\left(\frac{5}{2} - \frac{1}{2}(\tilde{x}^2 + \tilde{p}^2)\right), \\
&= \frac{5}{4}(\tilde{x}^2 + \tilde{p}^2) - \frac{1}{4}(\tilde{x}^2 + \tilde{p}^2)^2 - \frac{5}{4} + \frac{1}{4}(\tilde{x}^2 + \tilde{p}^2).
\end{aligned}
\tag{988}
$$

According to these expressions, the 2×2 Hamiltonian with $H_{12} = H_{21}$ can be written in Cartesian coordinates, as follows:

$$
\begin{aligned}
\hat{H} = &\frac{H_{11}}{2}\left(3 - \tilde{x}^2 - \tilde{p}^2\right) \\
&+ \frac{H_{12}}{\sqrt{2}}\left(4\tilde{x} - \tilde{x}^3 - \frac{1}{2}\tilde{p}^2\tilde{x} - \frac{1}{2}\tilde{x}\tilde{p}^2\right) \\
&+ \frac{H_{22}}{2}\left(3(\tilde{x}^2 + \tilde{p}^2) - \frac{1}{2}(\tilde{x}^2 + \tilde{p}^2)^2 - \frac{5}{2}\right).
\end{aligned}
\tag{989}
$$

When $H_{11} = -H_{22}$, as in the spin-Boson Hamiltonian, we obtain:

$$
\begin{aligned}
\hat{H} = &\frac{H_{11}}{4}\left(11 - 8(\tilde{x}^2 + \tilde{p}^2) + (\tilde{x}^2 + \tilde{p}^2)^2\right) \\
&+ \frac{H_{12}}{\sqrt{8}}\left(8\tilde{x} - 2\tilde{x}^3 - \tilde{p}^2\tilde{x} - \tilde{x}\tilde{p}^2\right)
\end{aligned}
\tag{990}
$$

The initial state $\psi(x; 0) = c_0(0)\langle x|0\rangle + c_1(0)\langle x|1\rangle$, with $|0\rangle$ and $|1\rangle$ defined according to Eq. (978), evolves in the 2-level system described by the Hamiltonian introduced by Eq. (990), as follows:

$$
\psi(x; t) = e^{-\frac{i}{\hbar}\hat{H}t}\psi(x; 0).
\tag{991}
$$

To apply the propagation scheme, introduced by Eq. (991), we first write the Hamiltonian in the Fourier grid form introduced in Sec. (20), as follows:

$$
\begin{aligned}
H_{j,j'} = {} & \left(\frac{H_{11}}{2} \left(3 - x_j^2 \right) + \frac{H_{12}}{\sqrt{2}} \left(4x_j - x_j^3 \right) + \frac{H_{22}}{2} \left(3x_j^2 - \frac{5}{2} - \frac{1}{2} x_j^4 \right) \right) \delta_{jj'} \\
& + \left(\frac{3}{2} H_{22} - \frac{1}{2} H_{11} - \frac{H_{12}}{\sqrt{8}} \left(x_{j'} + x_j \right) - \frac{H_{22}}{4} (x_j^2 + x_{j'}^2) \right) \frac{\Delta x \Delta p}{2\pi\hbar} \sum_{k=1}^{n_p} e^{\frac{i}{\hbar}(x_{j'}-x_j)p_k} p_k^2 \quad (992) \\
& - \frac{H_{22}}{4} \frac{\Delta x \Delta p}{2\pi\hbar} \sum_{k=1}^{n_p} e^{\frac{i}{\hbar}(x_{j'}-x_j)p_k} p_k^4,
\end{aligned}
$$

which is a special case of the DVR in the representation of equally spaced delta functions $\delta(x - x_j)$, at $x_j = (j - n_x/2)\Delta_x$, with $\Delta_x = (x_{max} - x_{min})/n_x$ and $j = 1\text{–}n_x$, where $p_k = \Delta p(k - n_p/2)$ with $\Delta p = 2\pi/(x_{max} - x_{min})$.

Having defined the Hamiltonian, according to Eq. (992), the propagation can then be carried out efficiently by using the *Expokit* software package that implements the Lanczos method based on the truncated Krylov basis set $u_j = H^j \psi(x,0)$ (usually with j=6-10). After orthogonalization of the basis with the *Arnoldi procedure* , the Hamiltonian is written in that small Lanczos basis as a tridiagonal matrix, H_N. It is subsequently diagonalized, $D_N = Z^\dagger H_N Z$, and used to propagate the initial state, as follows:

$$
\psi(x;t) = Z e^{-\frac{i}{\hbar} D_N t} Z^\dagger \psi(x;0), \quad (993)
$$

using a *Pade approximation* to the exponential function.

51.2 Computational Problem: SBH

1. Write a program to solve the time independent Schrödinger equation by using the Single Boson Hamiltonian (SBH) method and apply it to find the first 4 eigenvalues and eigenfunctions of the DVR Hamiltonian, introduced by Eq. (992), with $H_{11} = 1$, $H_{22} = -1$ and $H_{12} = 0.1$.
2. Propagate a state initialized as $|0\rangle$, according to Eq. (978), using the DVR Hamiltonian introduced by Eq. (992) and the Lanczos propagation scheme in the Krylov basis as implemented in *Expokit*. Compute the time-dependent probability of finding the system in the initial state $P_0(t) = \langle \psi(t)|0\rangle\langle 0|\psi(t)\rangle$ and compare your results to benchmark calculations based on the standard SOFT approach described in Sec. 7.
Solution: Sec. 61.22.

51.3 Wigner Transform Propagation Based on the SBH

We note that the Hamiltonian, introduced by Eq. (985), can also be written in terms of action-angle operators $\hat{n}$ and $\hat{\phi}$, satisfying

$$
\begin{aligned}
\hat{a} &= \hat{N}^{1/2}e^{i\hat{\phi}}, \\
&= \hat{N}^{1/2}\cos\hat{\phi} + i\hat{N}^{1/2}\sin\hat{\phi}, \\
&= \frac{1}{\sqrt{2}}\left(\tilde{x} + i\tilde{p}\right),
\end{aligned}
\tag{994}
$$

as follows:

$$
\begin{aligned}
\hat{H} &= H_{11}(1 - \hat{N}) + 2H_{12}(1 - \hat{N})\hat{N}^{1/2}\cos(\hat{\phi}) + H_{22}\hat{N}(2 - \hat{N}), \\
&= H_{11}(1 - \hat{N}) + \sqrt{2}H_{12}(1 - \hat{N})\tilde{x} + H_{22}\hat{N}(2 - \hat{N}),
\end{aligned}
\tag{995}
$$

where $\hat{N} = \frac{1}{2}\left(\tilde{x}^2 + \tilde{p}^2 - 1\right)$.

The equation of motion of $\rho_t^W(p, q)$ based on the SBH can be obtained by computing the time-derivative of both sides of Eq. (590), and substituting the time-derivative of the wavefunctions by using the time-dependent Schrödinger equation

$$
\begin{aligned}
i\hbar\frac{\partial\Psi(q \pm s/2; t)}{\partial t} &= \left(H_{11}(1 - \hat{N}) + \sqrt{2}H_{12}(1 - \hat{N})\tilde{x} + H_{22}\hat{N}(2 - \hat{N})\right)\Psi(q \pm s/2; t), \\
&= \left(H_{11}(1 - \hat{N}) + \sqrt{2}H_{12}(1 - \hat{N})(q \pm s/2) + H_{22}\hat{N}(2 - \hat{N})\right)\Psi(q \pm s/2; t),
\end{aligned}
\tag{996}
$$

or

$$
\begin{aligned}
i\hbar\frac{\partial\Psi(q \pm s/2; t)}{\partial t} &= \frac{H_{11}}{2}\left(3 - \tilde{x}^2 - \tilde{p}^2\right)\Psi(q \pm s/2; t) \\
&\quad + \frac{H_{12}}{\sqrt{2}}\left(4\tilde{x} - \tilde{x}^3 + i\tilde{p} - \tilde{x}\tilde{p}^2\right)\Psi(q \pm s/2; t) \\
&\quad + \frac{H_{22}}{2}\left(3(\tilde{x}^2 + \tilde{p}^2) - \frac{1}{2}(\tilde{x}^2 + \tilde{p}^2)^2 - \frac{5}{2}\right)\Psi(q \pm s/2; t),
\end{aligned}
\tag{997}
$$

giving

$$
\begin{aligned}
i\hbar\frac{\partial\Psi(q \pm s/2; t)}{\partial t} &= \left(\frac{H_{11}}{2}\left(3 - (q \pm s/2)^2\right) + \frac{H_{12}}{\sqrt{2}}\left(4(q \pm s/2) - (q \pm s/2)^3\right)\right. \\
&\quad \left. -\frac{H_{22}}{4}\left((q \pm s/2)^4 - 6(q \pm s/2)^2 + 3\right)\right)\Psi(q \pm s/2; t) \\
&\quad \pm (H_{12} + 2H_{22})\sqrt{2}\frac{\partial}{\partial s}\Psi(q \pm s/2; t) \\
&\quad + \left(H_{22}(2(q \pm s/2)^2 - 6) + H_{12}\sqrt{2}(q \pm s/2) + 2H_{11}\right)\frac{\partial^2}{\partial s^2}\Psi(q \pm s/2; t) \\
&\quad - H_{22}4\frac{\partial^4}{\partial s^4}\Psi(q \pm s/2; t).
\end{aligned}
\tag{998}
$$

where we have introduced the substitution $\frac{\partial \Psi(q \pm s/2; t)}{\partial q} = \pm 2 \frac{\partial \Psi(q+s/2;t)}{\partial s}$.

Therefore,

$$i\hbar \frac{\partial \Psi^*(q + s/2; t)}{\partial t} \Psi(q - s/2; t) = -f_1(q + s/2)\Psi^*\Psi - (H_{12} + 2H_{22})\sqrt{2}\frac{\partial \Psi^*}{\partial s}\Psi$$
$$- f_2(q + s/2)\frac{\partial^2 \Psi^*}{\partial s^2}\Psi + H_{22}4\frac{\partial^4 \Psi^*}{\partial s^4}\Psi. \tag{999}$$

and

$$i\hbar\Psi^* \frac{\partial \Psi(q - s/2; t)}{\partial t} = f_1(q - s/2)\Psi^*\Psi - (H_{12} + 2H_{22})\sqrt{2}\Psi^*\frac{\partial \Psi}{\partial s}$$
$$+ f_2(q - s/2)\Psi^*\frac{\partial^2 \Psi}{\partial s^2} - H_{22}4\Psi^*\frac{\partial^4 \Psi}{\partial s^4}. \tag{1000}$$

where

$$f_1(x) = \frac{H_{11}}{2}\left(3 - x^2\right) + \frac{H_{12}}{\sqrt{2}}\left(4x - x^3\right) - \frac{H_{22}}{4}\left(x^4 - 6x^2 + 3\right),$$
$$= \frac{3H_{11}}{2} - \frac{3H_{22}}{4} + 4\frac{H_{12}}{\sqrt{2}}x + \frac{1}{2}\left(3H_{22} - H_{11}\right)x^2 - \frac{H_{12}}{\sqrt{2}}x^3 - \frac{H_{22}}{4}x^4,$$
$$f_1'(x) = 4\frac{H_{12}}{\sqrt{2}} + \left(3H_{22} - H_{11}\right)x - \frac{3H_{12}}{\sqrt{2}}x^2 - H_{22}x^3,$$
$$f_1'''(x) = -\frac{6H_{12}}{\sqrt{2}} - 6H_{22}x, \tag{1001}$$

and

$$f_2(x) = H_{22}(2x^2 - 6) + H_{12}\sqrt{2}x + 2H_{11},$$
$$f_2'(x) = H_{22}4x + H_{12}\sqrt{2}, \tag{1002}$$
$$f_2''(x) = H_{22}4,$$

Thus, the time-derivative of the Wigner transform is

$$i\hbar\frac{\partial \rho^W(p, q; t)}{\partial t} = \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\left[\frac{\partial \Psi^*(q + s/2; t)}{\partial t}\Psi(q - s/2; t) + \Psi^*(q + s/2; t)\frac{\partial \Psi(q - s/2; t)}{\partial t}\right],$$
$$= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\left[f_1(q - s/2) - f_1(q + s/2)\right]\Psi^*\Psi$$
$$- (H_{12} + 2H_{22})\sqrt{2}\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\frac{\partial}{\partial s}\Psi^*\Psi$$
$$+ \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\left[f_2(q - s/2)\Psi^*\frac{\partial^2 \Psi}{\partial s^2} - f_2(q + s/2)\frac{\partial^2 \Psi^*}{\partial s^2}\Psi\right]$$
$$+ H_{22}4\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\left[\frac{\partial^4 \Psi^*}{\partial s^4}\Psi - \Psi^*\frac{\partial^4 \Psi}{\partial s^4}\right] \tag{1003}$$

or

$$i\hbar\frac{\partial \rho^W(p,q;t)}{\partial t} = \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\left[\frac{\partial \Psi^*(q+s/2;t)}{\partial t}\Psi(q-s/2;t) + \Psi^*(q+s/2;t)\frac{\partial \Psi(q-s/2;t)}{\partial t}\right],$$

$$= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\left[f_1(q-s/2) - f_1(q+s/2)\right]\Psi^*\Psi$$

$$- (H_{12} + 2H_{22})\sqrt{2}\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\frac{\partial}{\partial s}\Psi^*\Psi$$

$$+ \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\left[f_2(q-s/2)\Psi^*\frac{\partial^2 \Psi}{\partial s^2} - f_2(q+s/2)\frac{\partial^2 \Psi^*}{\partial s^2}\Psi\right]$$

$$+ H_{22}4\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\left[\frac{\partial^4 \Psi^*}{\partial s^4}\Psi - \Psi^*\frac{\partial^4 \Psi}{\partial s^4}\right]$$

$$\tag{1004}$$

Therefore

$$i\hbar\frac{\partial \rho^W(p,q;t)}{\partial t} = i\hbar f_1'(q)\frac{\partial \rho^W}{\partial p} + \frac{i\hbar}{3!}\left(\frac{\hbar}{2i}\right)^2 f_1'''(q)\frac{\partial^3 \rho^W}{\partial p^3},$$

$$+ \frac{i}{\hbar}p(H_{12} + 2H_{22})\sqrt{2}\rho^W$$

$$+ f_2(q)\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\left[\Psi^*\frac{\partial^2 \Psi}{\partial s^2} - \frac{\partial^2 \Psi^*}{\partial s^2}\Psi\right]$$

$$- \frac{f_2'(q)}{2}\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}s\left[\Psi^*\frac{\partial^2 \Psi}{\partial s^2} + \frac{\partial^2 \Psi^*}{\partial s^2}\Psi\right]$$

$$+ \frac{f_2''(q)}{4}\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}s^2\left[\Psi^*\frac{\partial^2 \Psi}{\partial s^2} - \frac{\partial^2 \Psi^*}{\partial s^2}\Psi\right]$$

$$+ \frac{H_{22}}{4}\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\left[\frac{\partial^4 \Psi^*}{\partial q^4}\Psi - \Psi^*\frac{\partial^4 \Psi}{\partial q^4}\right]$$

$$\tag{1005}$$

218

which gives

$$
\begin{aligned}
i\hbar\frac{\partial\rho^W(p,q;t)}{\partial t} = {} & i\hbar f_1'(q)\frac{\partial\rho^W}{\partial p} + \frac{i\hbar}{3!}\left(\frac{\hbar}{2i}\right)^2 f_1'''(q)\frac{\partial^3\rho^W}{\partial p^3}, \\
& + \frac{i}{\hbar}p(H_{12}+2H_{22})\sqrt{2}\rho^W \\
& + \frac{i}{\hbar}p\frac{f_2(q)}{2}\frac{\partial\rho^W}{\partial q} \\
& - \frac{f_2'(q)}{2}\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}s\frac{\partial}{\partial q}\left[-\Psi^*\frac{\partial\Psi}{\partial s}+\frac{\partial\Psi^*}{\partial s}\Psi\right] \\
& + \frac{f_2'(q)}{2}\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}s\left[-\frac{\partial\Psi^*}{\partial q}\frac{\partial\Psi}{\partial s}+\frac{\partial\Psi^*}{\partial s}\frac{\partial\Psi}{\partial q}\right] \\
& - \frac{f_2''(q)}{8}\frac{\partial}{\partial q}\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}s^2\frac{\partial}{\partial s}\Psi^*\Psi \\
& + \frac{H_{22}}{4}\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\left[\frac{\partial^4\Psi^*}{\partial q^4}\Psi-\Psi^*\frac{\partial^4\Psi}{\partial q^4}\right]
\end{aligned}
\tag{1006}
$$

$$
\begin{aligned}
i\hbar\frac{\partial\rho^W(p,q;t)}{\partial t} = {} & i\hbar f_1'(q)\frac{\partial\rho^W}{\partial p} + \frac{i\hbar}{3!}\left(\frac{\hbar}{2i}\right)^2 f_1'''(q)\frac{\partial^3\rho^W}{\partial p^3}, \\
& + \frac{i}{\hbar}p(H_{12}+2H_{22})\sqrt{2}\rho^W \\
& + \frac{i}{\hbar}p\frac{f_2(q)}{2}\frac{\partial\rho^W}{\partial q} \\
& - \frac{f_2'(q)}{2}\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}s\left[\Psi^*\frac{\partial^2\Psi}{\partial s^2}+\frac{\partial^2\Psi^*}{\partial s^2}\Psi\right] \\
& - i\hbar\frac{f_2''(q)}{8}\left(2\frac{\partial^2\rho^W}{\partial q\partial p}+p\frac{\partial^3\rho^W}{\partial q\partial p^2}\right) \\
& + \frac{H_{22}}{4}\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\left[\frac{\partial^4\Psi^*}{\partial q^4}\Psi-\Psi^*\frac{\partial^4\Psi}{\partial q^4}\right]
\end{aligned}
\tag{1007}
$$

219

$$i\hbar\frac{\partial\rho^W(p,q;t)}{\partial t} = i\hbar f_1'(q)\frac{\partial\rho^W}{\partial p} + \frac{i\hbar}{3!}\left(\frac{\hbar}{2i}\right)^2 f_1'''(q)\frac{\partial^3\rho^W}{\partial p^3},$$
$$+ \frac{i}{\hbar}p(H_{12}+2H_{22})\sqrt{2}\rho^W$$
$$+ \frac{i}{\hbar}p\frac{f_2(q)}{2}\frac{\partial\rho^W}{\partial q}$$
$$- \frac{f_2'(q)}{2}\frac{\hbar}{i}\frac{\partial^2}{\partial p\partial q}\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty}ds\,e^{\frac{i}{\hbar}ps}\left[-\Psi^*\frac{\partial\Psi}{\partial s}+\frac{\partial\Psi^*}{\partial s}\Psi\right] \qquad (1008)$$
$$- \frac{f_2'(q)}{2}\frac{\hbar}{i}\frac{\partial}{\partial p}\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty}ds\,e^{\frac{i}{\hbar}ps}\left[\frac{\partial\Psi^*}{\partial q}\frac{\partial\Psi}{\partial s}-\frac{\partial\Psi^*}{\partial s}\frac{\partial\Psi}{\partial q}\right]$$
$$- i\hbar\frac{f_2''(q)}{8}\left(2\frac{\partial^2\rho^W}{\partial q\partial p}+p\frac{\partial^3\rho^W}{\partial q\partial p^2}\right)$$
$$+ \frac{H_{22}}{4}\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty}ds\,e^{\frac{i}{\hbar}ps}\left[\frac{\partial^4\Psi^*}{\partial q^4}\Psi-\Psi^*\frac{\partial^4\Psi}{\partial q^4}\right]$$

$$i\hbar\frac{\partial\rho^W(p,q;t)}{\partial t} = i\hbar f_1'(q)\frac{\partial\rho^W}{\partial p} + \frac{i\hbar}{3!}\left(\frac{\hbar}{2i}\right)^2 f_1'''(q)\frac{\partial^3\rho^W}{\partial p^3},$$
$$+ \frac{i}{\hbar}p(H_{12}+2H_{22})\sqrt{2}\rho^W$$
$$+ \frac{i}{\hbar}p\frac{f_2(q)}{2}\frac{\partial\rho^W}{\partial q}$$
$$- \frac{f_2'(q)}{2}\frac{\hbar}{i}\frac{\partial^2}{\partial p\partial q}\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty}ds\,e^{\frac{i}{\hbar}ps}\frac{\partial}{\partial q}\left[\Psi^*\Psi\right] \qquad (1009)$$
$$+ f_2'(q)\frac{\hbar}{i}\frac{\partial}{\partial p}\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty}ds\,e^{\frac{i}{\hbar}ps}\frac{\partial\Psi^*}{\partial q}\frac{\partial\Psi}{\partial q}$$
$$- i\hbar\frac{f_2''(q)}{8}\left(2\frac{\partial^2\rho^W}{\partial q\partial p}+p\frac{\partial^3\rho^W}{\partial q\partial p^2}\right)$$
$$+ \frac{H_{22}}{4}\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty}ds\,e^{\frac{i}{\hbar}ps}\left[\frac{\partial^4\Psi^*}{\partial q^4}\Psi-\Psi^*\frac{\partial^4\Psi}{\partial q^4}\right]$$

$$i\hbar\frac{\partial\rho^W}{\partial t} = i\hbar f_1'(q)\frac{\partial\rho^W}{\partial p} + \frac{i\hbar}{3!}\left(\frac{\hbar}{2i}\right)^2 f_1'''(q)\frac{\partial^3\rho^W}{\partial p^3} + \frac{i}{\hbar}p(H_{12}+2H_{22})\sqrt{2}\rho^W$$
$$+ \frac{i}{\hbar}p\frac{f_2(q)}{2}\frac{\partial\rho^W}{\partial q} + i\hbar\frac{f_2'(q)}{8}\frac{\partial}{\partial p}\frac{\partial^2\rho^W}{\partial q^2} - i\hbar\frac{f_2''(q)}{8}\left(2\frac{\partial^2\rho^W}{\partial q\partial p}+p\frac{\partial^3\rho^W}{\partial q\partial p^2}\right)$$
$$- i\hbar\frac{f_2'(q)}{4}\frac{\partial}{\partial p}\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty}ds\,e^{\frac{i}{\hbar}ps}\frac{\partial\Psi^*}{\partial q}\frac{\partial\Psi}{\partial q} + \frac{H_{22}}{4}\frac{1}{2\pi\hbar}\int_{-\infty}^{\infty}ds\,e^{\frac{i}{\hbar}ps}\left[\frac{\partial^4\Psi^*}{\partial q^4}\Psi-\Psi^*\frac{\partial^4\Psi}{\partial q^4}\right]$$
$$\tag{1010}$$

Pauli Matrices: Pauli matrices in Cartesian representation are written, as follows:

$$\hat{\sigma}_z = (1 - \hat{N}) - \hat{N}(2 - \hat{N})$$

$$= \frac{11}{4} - 2(\tilde{x}^2 + \tilde{p}^2) + \frac{1}{4}(\tilde{x}^2 + \tilde{p}^2)^2$$

$$\hat{\sigma}_x = (1 - \hat{N})\hat{a} + \hat{a}^\dagger(1 - \hat{N})$$

$$= \frac{1}{\sqrt{8}}\left(8\tilde{x} - 2\tilde{x}^3 - \tilde{p}^2\tilde{x} - \tilde{x}\tilde{p}^2\right) \tag{1011}$$

$$\hat{\sigma}_y = -i(1 - \hat{N})\hat{a} + i\hat{a}^\dagger(1 - \hat{N})$$

$$= \frac{i}{2\sqrt{2}}\left(-i6\tilde{p} + \tilde{p}^2\tilde{x} - \tilde{x}\tilde{p}^2 + i(\tilde{p}\tilde{x}^2 + \tilde{x}^2\tilde{p}) + 2i\tilde{p}^3\right)$$

$$= \frac{1}{\sqrt{8}}\left(6\tilde{p} + i\tilde{p}^2\tilde{x} - i\tilde{x}\tilde{p}^2 - (\tilde{p}\tilde{x}^2 + \tilde{x}^2\tilde{p}) - 2\tilde{p}^3\right),$$
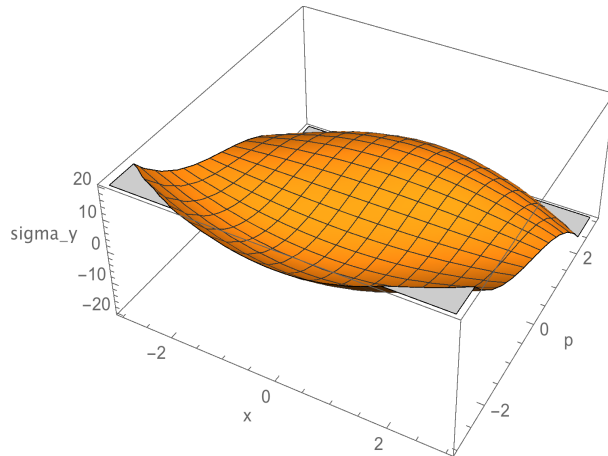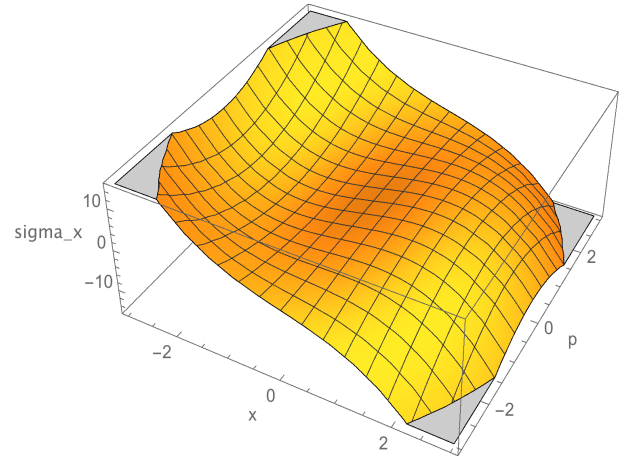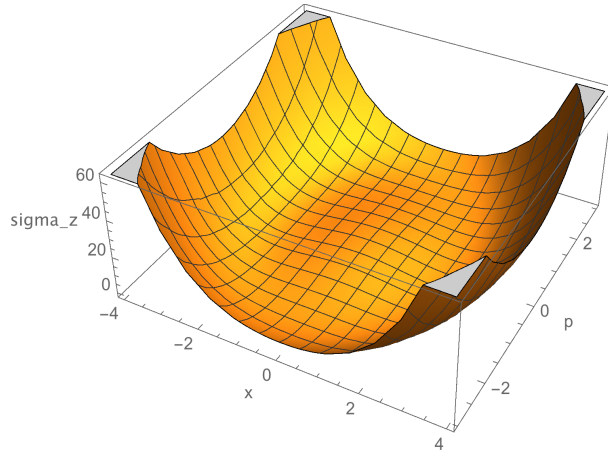
$$\hat{\sigma}^+ = \frac{1}{2}\left(\hat{\sigma}_x + i\hat{\sigma}_y\right) = (1-\hat{N})\hat{a},$$

$$= \frac{1}{2\sqrt{8}}\left(8\tilde{x} - 2\tilde{x}^3 - \tilde{p}^2\tilde{x} - \tilde{x}\tilde{p}^2\right)$$

$$+ \frac{i}{2\sqrt{8}}\left(6\tilde{p} + i\tilde{p}^2\tilde{x} - i\tilde{x}\tilde{p}^2 - (\tilde{p}\tilde{x}^2 + \tilde{x}^2\tilde{p}) - 2\tilde{p}^3\right),$$

$$= \frac{1}{2\sqrt{8}}\left(8\tilde{x} - 2\tilde{x}^3 - 2\tilde{p}^2\tilde{x}\right)$$

$$+ \frac{i}{2\sqrt{8}}\left(6\tilde{p} - (\tilde{p}\tilde{x}^2 + \tilde{x}^2\tilde{p}) - 2\tilde{p}^3\right),$$

$$= \frac{1}{\sqrt{8}}\left(5\tilde{x} - \tilde{x}^3 - \tilde{p}^2\tilde{x}\right) + \frac{i}{\sqrt{8}}\left(3\tilde{p} - \tilde{p}^3 - \tilde{p}\tilde{x}^2\right),$$

$$\hat{\sigma}^- = \frac{1}{2}\left(\hat{\sigma}_x - i\hat{\sigma}_y\right) = \hat{a}^\dagger(1-\hat{N}),$$

$$= \frac{1}{2\sqrt{8}}\left(8\tilde{x} - 2\tilde{x}^3 - \tilde{p}^2\tilde{x} - \tilde{x}\tilde{p}^2\right)$$

$$- \frac{i}{2\sqrt{8}}\left(6\tilde{p} + i\tilde{p}^2\tilde{x} - i\tilde{x}\tilde{p}^2 - (\tilde{p}\tilde{x}^2 + \tilde{x}^2\tilde{p}) - 2\tilde{p}^3\right),$$

$$= \frac{1}{2\sqrt{8}}\left(8\tilde{x} - 2\tilde{x}^3 - 2\tilde{x}\tilde{p}^2\right)$$

$$- \frac{i}{2\sqrt{8}}\left(6\tilde{p} - (\tilde{p}\tilde{x}^2 + \tilde{x}^2\tilde{p}) - 2\tilde{p}^3\right),$$

$$= \frac{1}{\sqrt{8}}\left(3\tilde{x} - \tilde{x}^3 - \tilde{x}\tilde{p}^2\right) - \frac{i}{\sqrt{8}}\left(3\tilde{p} - \tilde{p}^3 - \tilde{p}\tilde{x}^2\right).$$

(1012)
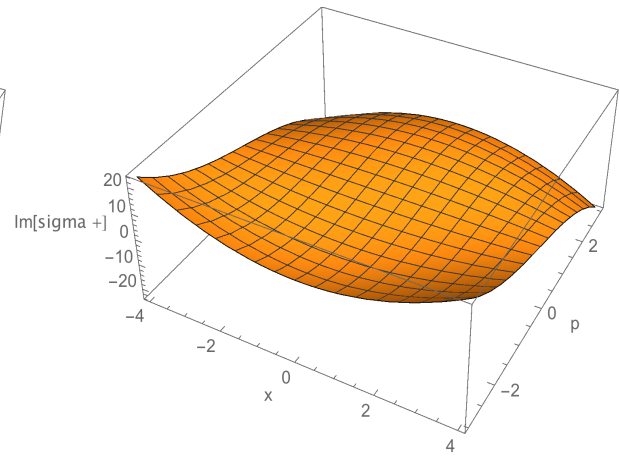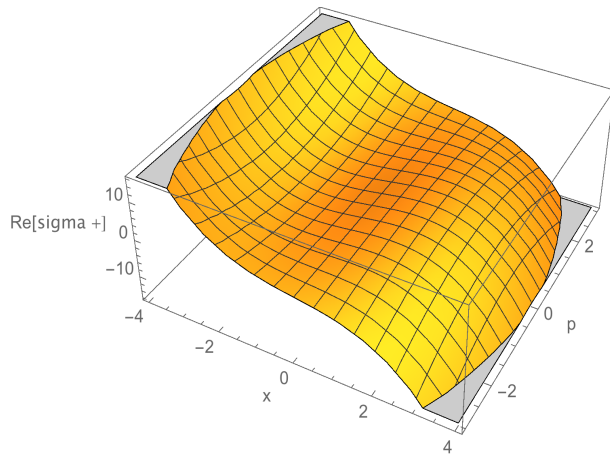
52 Tunneling Current: Formula

We consider a 1-dimensional electron tunneling

$$\hat{H}\psi = \left[-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + eV(x) \right]\psi, \tag{1013}$$

problem described by the Hamiltonian

$$\hat{H} = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + eV(x), \tag{1014}$$

where e is the charge of the electron and

$$V(x) = \begin{cases} V_l & \text{if} \quad x < 0, \\ V_b & \text{if} \quad 0 < x < a, \\ V_r & \text{if} \quad x > a, \end{cases} \tag{1015}$$

where V_b defines the tunneling barrier, and $\Delta V = (V_l - V_r)$ defines the voltage drop across the barrier. Outside the tunneling interval $x_l < x < x_r$, the solutions of the Schrödinger equation are superpositions of plane waves since the potential is constant. For energy $E > eV_l$ and $E > eV_r$, there are two independent solutions ψ_l and ψ_r for incident electrons from the left and from the right, respectively.

Considering the solution for incidence from the left, we obtain:

$$\psi_l(x) = \begin{cases} \phi_l^+ + r_l\phi_l^- & \text{if} \quad x < 0 \\ Ae^{ik_b x} + Be^{-ik_b x} & \text{if} \quad 0 < x < a \\ t_r\phi_r^+ & \text{if} \quad x > a \end{cases} \tag{1016}$$

where $\phi_j^\pm = k_j^{-1/2}e^{\pm ik_j x}$, are defined divided by the square root of k_j so they are normalized to carry the unit of current density $\hbar/m$, as shown below, and ensure that the S matrix is unitary (conservation of charge). The labels $j = l, r$ indicate the left (l) and right (r) side of the barrier, $k_l = \sqrt{2m(E - eV_l)/\hbar^2}$ and $k_r = \sqrt{2m(E - eV_r)/\hbar^2}$.

Applying the continuity conditions for ψ_l and $\partial\psi_l/\partial x$ at $x = 0$ and $x = a$, we obtain:

$$\begin{aligned} k_l^{-1/2} + k_l^{-1/2}r_l &= A + B, \\ Ae^{ik_b a} + Be^{-ik_b a} &= k_r^{-1/2}t_r e^{ik_r a}, \\ k_l^{1/2}(1 - r_l) &= k_b(A - B), \\ k_b(Ae^{ik_b a} - Be^{-ik_b a}) &= k_r^{1/2}t_r e^{ik_r a}. \end{aligned} \tag{1017}$$

The transmission amplitude t_r, reflection amplitude r_l and coefficients A and B can be obtained by solving for them from Eq. (1017).

The probability flux (or current density) of incoming electrons from the left, described by the incident wave $\psi_i(x, t) = k_l^{-1/2}e^{i(k_l x - wt)}$ with momentum k_l and energy $E(k_l) = eV_l + \hbar^2 k_l^2/(2m)$, is:

$$j_i(x, t) = \frac{\hbar}{2mi}\left(\psi_i^*(x, t)\frac{\partial\psi_i(x, t)}{\partial x} - \psi_i(x, t)\frac{\partial\psi_i^*(x, t)}{\partial x} \right), \tag{1018}$$

223

or

$$j_i(x) = \frac{1}{2}\psi_i^*(x,t)\left(-i\frac{\hbar}{m}\frac{\partial}{\partial x}\right)\psi_i(x,t) + c.c.,$$
$$= \frac{1}{2}\psi_i^*(x,t)\frac{\hat{p}}{m}\psi_i(x,t) + c.c., \tag{1019}$$
$$= \mathrm{Re}[\psi_i^*(x,t)\hat{v}\psi_i(x,t)] = \frac{\hbar}{m}.$$

The flux of transmitted electrons described by transmitted wave $\psi_t(x,t) = t_r k_r^{-1/2} e^{i(k_r x - wt)}$, with momentum k_r and energy $E(k_r) = eV_r + \hbar^2 k_r^2/(2m)$, is:

$$j_t(x) = \frac{1}{2}\psi_t^*(x,t)\left(-i\frac{\hbar}{m}\frac{\partial}{\partial x}\right)\psi_t(x,t) + c.c.,$$
$$= |t_r|^2\frac{\hbar}{m}. \tag{1020}$$

Therefore, the transmission coefficient $T_l = j_t/j_i$, defined as the transmitted flux j_t over the incident flux at energy E is: $T_l = |t_r|^2$. The reflection coefficient $R_l = 1 - T_l$ is the reflected flux over the incident flux.

Analogously, we consider incidence from the right of the tunneling barrier, as follows:

$$\psi_r(x) = \begin{cases} \phi_r^- + r_r\phi_r^+ & \text{if } x > a \\ Ae^{ik_b x} + Be^{-ik_b x} & \text{if } 0 < x < a \\ t_l\phi_l^- & \text{if } x < 0 \end{cases} \tag{1021}$$

Solving for t_l, we obtained the transmission coefficient $T_r = |t_l|^2$, due to incidence from the right.

More generally, we can consider incoming waves from both left *and* right (ϕ_l^+ and ϕ_r^-, respectively) with amplitudes $\mathbf{c}_{in} = c_{in}^{(l)}, c_{in}^{(r)}$ that generate outgoing waves to the left and right (ϕ_l^- and ϕ_r^+, respectively) with amplitudes $\mathbf{c}_{out} = c_{out}^{(l)}, c_{out}^{(r)}$. The amplitudes of outgoing and incoming waves are related by the linear transformation defined by the scattering matrix (or, 'S-matrix') $\mathbf{S}$, as follows: $\mathbf{c}_{out} = \mathbf{S}\mathbf{c}_{in}$:

$$\begin{pmatrix} c_{out}^l \\ c_{out}^r \end{pmatrix} = \begin{pmatrix} r_l & t_l \\ t_r & r_r \end{pmatrix}\begin{pmatrix} c_{in}^l \\ c_{in}^r \end{pmatrix} \tag{1022}$$

Due to the conservation of probability, the S-matrix must be unitary: $\mathbf{S}^{-1} = \mathbf{S}^\dagger$. Therefore, $\mathbf{S}\mathbf{S}^\dagger = 1$:

$$\begin{pmatrix} r_l & t_l \\ t_r & r_r \end{pmatrix}\begin{pmatrix} r_l^\dagger & t_r^\dagger \\ t_l^\dagger & r_r^\dagger \end{pmatrix} = 1 \tag{1023}$$

which gives

$$r_l r_l^\dagger + t_l t_l^\dagger = 1. \tag{1024}$$

224

In addition, $\mathbf{S}^\dagger \mathbf{S} = 1$:

$$\begin{pmatrix} r_l^\dagger & t_r^\dagger \\ t_l^\dagger & r_r^\dagger \end{pmatrix} \begin{pmatrix} r_l & t_l \\ t_r & r_r \end{pmatrix} = 1 \tag{1025}$$

which gives

$$t_l^\dagger t_l + r_r^\dagger r_r = 1 \tag{1026}$$

Therefore, according to Eqs. (1024) and (1026), we obtain: $1 - t_l t_l^\dagger = r_l r_l^\dagger = r_r^\dagger r_r$. For our 1-dimensional case, we obtain:

$$|r_l|^2 = |r_r|^2 = R. \tag{1027}$$

Under stationary state, $\partial\rho/\partial t = 0$, with $\rho = |\psi^*\psi|$. Then, according to the continuity equation $\partial\rho/\partial t = -\partial j/\partial x$, we obtain: $\partial j/\partial x = 0$. Therefore, j_l for $x < 0$ must be equal to j_l for $x > a$. Also, j_r for $x < 0$ must be equal to j_r for $x > a$:

$$(1 - |r_l|^2) = |t_r|^2, \tag{1028}$$

and

$$(1 - |r_r|^2) = |t_l|^2. \tag{1029}$$

Dividing Eq. (1028) by Eq. (1029) and using (1027), we obtain:

$$t_l = t_r. \tag{1030}$$

Therefore,

$$\begin{aligned} T_l(E) &= |t_r|^2, \\ &= |t_l|^2 = T_r(E), \end{aligned} \tag{1031}$$

so the transmission coefficient is the same for both directions of incidence and $R + T = 1$.

Considering that the number of electrons with energy E incident from the left and right of the barrier are $n_l(E)$ and $n_r(E)$, respectively, the net flux of charge from left to right is:

$$\begin{aligned} I &= 2e \int_0^\infty dk_l\, n_l(k_l)\frac{\hbar k_l}{m}T_l - 2e \int_0^\infty dk_r\, n_r(k_r)\frac{\hbar k_r}{m}T_r, \\ &= \frac{2e}{2\pi} \int_0^\infty dE\, T(E) \left(n_l(E)\frac{\hbar k_l}{m}\left|\frac{\partial k_l}{\partial E}\right| - n_r(E)\frac{\hbar k_r}{m}\left|\frac{\partial k_r}{\partial E}\right| \right), \\ &= \frac{2e}{2\pi} \int_0^\infty dE\, T(E) \left(n_l(E)\frac{\hbar k_l}{m}\left|\frac{m}{\hbar^2 k_l}\right| - n_r(E)\frac{\hbar k_r}{m}\left|\frac{m}{\hbar^2 k_r}\right| \right), \\ &= \frac{2e}{h} \int_0^\infty dE\, T(E) \left(n_l(E) - n_r(E) \right), \end{aligned} \tag{1032}$$

where factor of 2 accounts for the two possible spin states, the first term on the r.h.s. accounts for the forward flux (i.e., from left to right) and the second term accounts for the backward flux

(i.e., from right to left). Note that in the second row of Eq. (1032) we used the following equality: $\hat{1} = \int dE |E\rangle\langle E| = 2\pi \int dk |k\rangle\langle k|$.

At equilibrium, the population of energy levels is determined by the Fermi-Dirac distribution:

$$n(E) = \frac{1}{e^{\beta(E-E_F)} + 1},$$

(1033)

where E_F is the Fermi level and the factor of 2 in the numerator accounts for the 2 possible spin states. Considering the potentials for electrons at either side of the barrier, we obtain $n_l(E) = n(E - eV_l)$ and $n_r(E) = n(E - eV_r)$. Therefore, we can expand these distributions, as follows:

$$n_l(E) = n(E - E_F) + \frac{\partial n(E)}{\partial E}eV_l + \cdots,$$

$$n_r(E) = n(E - E_F) + \frac{\partial n(E)}{\partial E}eV_r + \cdots,$$

(1034)

and write the Landauer formula, giving the current in the form of the Ohm's lAF, as follows:

$$\begin{aligned} I &= \frac{2e}{h} \int_0^\infty dE\, T(E)\, (n_l(E) - n_r(E)), \\ &= \frac{2e^2}{h} \int dE\, T(E)\frac{\partial n(E)}{\partial E}\Delta V, \\ &= G(E)\Delta V, \end{aligned}$$

(1035)

where $G(E) = R^{-1} = G_0 \int dE\, T(E)\frac{\partial n(E)}{\partial E}$ is the conductance, or inverse of the resistance R, with $G_0 = \frac{2e^2}{h} = [12.906 \text{ k}\Omega]^{-1}$ the quantum unit of conductance. Note that G_0 defines the maximum conductance (minimum resistance) per conduction channel with perfect transmission, $T(E) = 1$ (i.e., if the transport through the channel is ballistic and therefore the probability for transmitting the electron that enters the channel is unity), as observed in experiments.

At low temperature (i.e., $\beta \to \infty$), the Fermi-Dirac distributions become step functions $n_l(E) = 2H(E_F - (E - eV_l))$ and $n_r(E) = 2H(E_F - (E - eV_r))$, with $H(x)$ the Heaviside function equal to 1 for $x > 0$, and 0 for $x < 0$. Therefore, $\frac{\partial n(E)}{\partial E} = \delta(E_F - E)$, and

$$\begin{aligned} I &= \frac{2e^2}{h} \int dE\, T(E)\delta(E_F - E)\Delta V, \\ &= \frac{2e^2}{h}\, T(E_F)\Delta V. \end{aligned}$$

(1036)

In this low-temperature limit, the conductance is the transmission times the quantum of conductance, $G(E) = \frac{2e^2}{h}\, T(E_F)$.

52.1 WKB Transmission

The goal of this subsection is to show that the transmission coefficient $T(E)$ can be estimated, under the WKB approximation, as follows:

$$T(E) = e^{-2\int_0^a dx\sqrt{2m|E-\xi(x)|/\hbar^2}},$$

(1037)
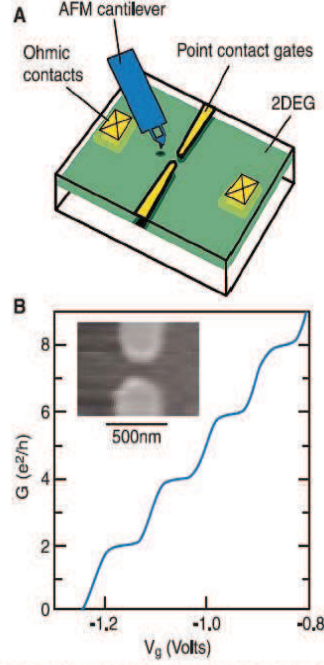
Fig. 1. (A) Schematic diagram of the experimental setup. QPC conductance is measured as a function of AFM tip position. **(B)** Point contact conductance G versus gate voltage V_g with no tip present at temperature $T = 1.7$ K. Plateaus at integer multiples of $2 e^2/h$ are clearly seen. The inset shows a topographic image of the point contact device.

where $\xi(x) = V_b$ describes the tunneling barrier according to Eq. (1015).

To derive Eq. (1037), we consider the WKB approximate solution of Eq. (1126), with the following functional form:

$$\psi(x) = \psi_0 e^{i \int_0^x k(x')dx'}, \tag{1038}$$

where $k(x) = \sqrt{2m[E - V(x)]/\hbar^2}$. Note that when $V(x)$ is constant, $\psi(x)$ corresponds to a particle moving to the right with constant momentum k. Substituting $\psi(x)$ as defined in Eq. (1038), into Eq. (1126), we obtain:

$$-\frac{\hbar^2}{2m}\frac{\partial^2 \psi(x)}{\partial x^2} + V(x)\psi(x) = E\psi(x) - \Delta, \tag{1039}$$

with $\Delta = ik'(x)\frac{\hbar^2}{2m}\psi(x)$. Therefore, the WKB solution is a good approximation when $|k'(x)| << k(x)^2$.

According to the WKB solution, the probability density $|\psi(x)|^2$ remains constant on the left of the tunneling barrier, when $E > V_l$, since $\psi(x) = \psi(-\infty)e^{-i\int_{-\infty}^x dx' k_l}$ for $x < 0$. Inside the barrier, however, the probability density decays exponentially:

$$\psi(x) = \psi(0)e^{-\int_0^x dx' \sqrt{2m|E-\xi(x')|/\hbar^2}}, \tag{1040}$$

227

since $E < \xi(x)$. In particular, at $x = a$, the probability density is

$$|\psi(x)|^2 = |\psi(0)|^2 e^{-2\int_0^a dx' |k(x')|}. \tag{1041}$$

In the region with $x > a$, the probability density remains constant again since $\psi(x) = \psi(a)e^{-i\int_a^x dx' k_r}$ and $|\psi(x)|^2 = |\psi(a)|^2$. Therefore, estimating the transmission coefficient as the ratio of the probability densities to the right and to the left of the barrier, we obtain

$$
\begin{aligned}
T(E) &= \frac{|\psi(a)|^2}{|\psi(0)|^2}, \\
&= e^{-2\int_0^a dx \sqrt{2m|E-\xi(x)|/\hbar^2}}.
\end{aligned}
\tag{1042}
$$

52.2   Green's Function Formalism

The transmission coefficient can be defined as follows:

$$T = Tr[\Gamma_s G(E)\Gamma_d G^\dagger(E)], \tag{1043}$$

where $G(E)$ is the

53 Supplement: Functions of Matrices

The goal of this supplementary section is to explain how to compute a function $f(\mathbf{A})$ of a hermitian matrix $\mathbf{A}$. Analogously to the function $f(x)$ of simple variable x, $f(\mathbf{A})$ can be expanded in powers of $\mathbf{A}$:

$$f(\mathbf{A}) = \sum_{n=0}^{\infty} \alpha_n \mathbf{A}^n. \tag{1044}$$

with $\alpha_n = \frac{1}{n!} \frac{\partial f(x)}{\partial x}\Big|_{x=0}$. Equation (1044) shows how to compute a function of a matrix as a sum of products of such matrix.

Note that when the matrix is diagonal

$$\mathbf{A} = \begin{pmatrix} a_1 & 0 & \cdots & 0 \\ 0 & a_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & a_N \end{pmatrix}, \tag{1045}$$

then

$$\mathbf{A}^n = \begin{pmatrix} a_1^n & 0 & \cdots & 0 \\ 0 & a_2^n & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & a_N^n \end{pmatrix}. \tag{1046}$$

When $\mathbf{A}$ is not diagonal, it can be diagonalized by the similarity transformation $\mathbf{a} = \mathbf{c}^\dagger \mathbf{A} \mathbf{c}$, where $\mathbf{a}$ is the diagonal matrix of eigenvalues $a_1, a_2, \cdots, a_N$ and $\mathbf{c}$ is the matrix of column eigenvectors of $\mathbf{A}$. Therefore,

$$\begin{aligned} \mathbf{A}^n &= [\mathbf{c}\mathbf{a}\mathbf{c}^\dagger]^n, \\ &= \mathbf{c}\mathbf{a}^n\mathbf{c}^\dagger \end{aligned} \tag{1047}$$

since $\mathbf{c}^\dagger \mathbf{c} = 1$. Substituting Eq. (1047) into Eq. (1044), we obtain:

$$f(\mathbf{A}) = \mathbf{c} f(\mathbf{a}) \mathbf{c}^\dagger. \tag{1048}$$

53.1 Optional Exercise: Green's Function

(A) Show that the matrix elements of the Green's function $\mathbf{G_0} = (E\mathbf{1} - \mathbf{H_0})^{-1}$ can be computed, according to Eq. (1048), as follows:

$$[\mathbf{G_0}]_{ik} = \sum_{j=1}^{N} \frac{c_j^{(i)} c_j^{(k)*}}{E - E_j^{(0)}}, \tag{1049}$$

where $E_j^{(0)}$ is the j-th eigenvalue of $\mathbf{H_0}$ associated with the eigenvector $\mathbf{c}_j = (c_j^{(1)}, c_j^{(2)}, \cdots, c_j^{(N)})$.

(B) Show that the Green's function

$$\mathbf{G} = (E\mathbf{1} - (\mathbf{H_0} + \mathbf{V}))^{-1}, \tag{1050}$$

obeys the *Dyson equation*:

$$\mathbf{G} = \mathbf{G_0} + \mathbf{G_0}\mathbf{V}\mathbf{G}. \tag{1051}$$

Hint: Note that by multiplying both sides of Eq. (1050) by $\mathbf{G_0}(E\mathbf{1} - (\mathbf{H_0} + \mathbf{V}))$, we obtain:

$$\mathbf{G_0}(E\mathbf{1} - (\mathbf{H_0} + \mathbf{V}))\mathbf{G} = \mathbf{G_0}. \tag{1052}$$

As an application of the Green's function, we consider the eigenvalue problem:

$$\left[\hat{H}_0 + \hat{V}\right]\psi_E(x) = E\psi_E(x), \tag{1053}$$

which can be re-written, as follows:

$$(E - \hat{H}_0)\psi_E(x) = \phi(x), \tag{1054}$$

with $\phi(x) = \hat{V}\psi_E(x)$. Expanding $\psi_E(x)$ and $\phi(x)$ in the basis of eigenfunctions of $\hat{H}_0$:

$$\hat{H}_0\psi_\alpha(x) = E_\alpha^{(0)}\psi_\alpha(x), \tag{1055}$$

we obtain:

$$\psi_E(x) = \sum_\alpha a_{\alpha,E}\psi_\alpha(x), \tag{1056}$$

and

$$\phi(x) = \sum_\alpha b_\alpha\psi_\alpha(x), \tag{1057}$$

where

$$b_\alpha = \int dx\,\psi_\alpha^*(x)\phi(x). \tag{1058}$$

In addition, substituting Eqs. (1056) and (1057) into Eq. (1054), we obtain:

$$\sum_\beta a_{\beta,E}(E - E_\beta^{(0)})\psi_\beta(x) = \sum_\beta b_\beta\psi_\beta(x), \tag{1059}$$

and multiplying both sides of Eq. (1059) by $\psi_\alpha^*(x)$ and integrating over x, we obtain:

$$\begin{aligned}
a_{\alpha,E} &= \frac{b_\alpha}{E - E_\alpha^{(0)}}, \\
&= \frac{1}{E - E_\alpha^{(0)}}\int dx'\,\psi_\alpha^*(x')\phi(x').
\end{aligned} \tag{1060}$$

Substituting Eq. (1060) into Eq. (1056), we obtain:

$$\psi_E(x) = \int dx' \sum_\alpha \frac{\psi_\alpha^*(x')\psi_\alpha(x)}{E - E_\alpha^{(0)}}\phi(x'),$$

$$= \int dx' G_0(x, x'; E)V(x')\psi_E(x'),$$

(1061)

where $G_0(x, x'; E) = \sum_\alpha \frac{\psi_\alpha^*(x')\psi_\alpha(x)}{E - E_\alpha^{(0)}}$ has poles at values of E equal to the eigenvalues of $\hat{H}_0$.

As a by-product of Eq. (1061), we note that in the particular case of $\phi(x') = \delta(x' - x'')$, we obtain:

$$\psi_E(x) = \sum_\alpha \frac{\psi_\alpha^*(x'')\psi_\alpha(x)}{E - E_\alpha^{(0)}},$$

$$= G_0(x, x''; E).$$

(1062)

Substituting Eq. (1062) into Eq. (1054), we obtain:

$$(E \overset{\hat{}}{-} H_0)G_0(x, x''; E) = \delta(x - x''),$$

(1063)

or in matrix form,

$$(E\mathbf{1} - \mathbf{H_0})\mathbf{G_0}(E) = \mathbf{1}.$$

(1064)

Equation (1062) shows that the Green's function can be computed in terms of the eigenfunctions $\psi_\alpha(x)$ and eigenvalues E_α of $\hat{H}_0$.

Analogously, we obtain the Green's function of $\hat{H} = \hat{H}_0 + \hat{V}$:

$$(E - \hat{H}_0 - V)G(x, x''; E) = \delta(x - x''),$$

(1065)

or in matrix form,

$$(E\mathbf{1} - \mathbf{H}_0 - V)\mathbf{G}(E) = \mathbf{1},$$

(1066)

which, according to Eq. (1051), obeys the Dyson integral equation:

$$G(x, x''; E) = G_0(x, x''; E) + \int dx' G_0(x, x'; E)V(x')G(x', x''; E),$$

(1067)

and has poles at values of E equal to the eigenvalues of $\hat{H}$.

It is important to note that the Green's function can be used to compute several functions, including the density matrix $\rho(x, x'') = \sum_\alpha n_\alpha \psi_\alpha^*(x'')\psi_\alpha(x)$, as follows:

$$\rho(x, x'') = \frac{i}{\pi} \int_{-\infty}^{\infty} dE \, n(E)G(x, x''; E).$$

(1068)

To prove Eq. (1068), we compute the integral

$$I(x, x'') = \int_{-\infty}^{\infty} dE \, n(E)G(x, x''; E).$$

(1069)

Substituting Eq. (1062) into Eq. (1069), we obtain:

$$I(x, x'') = \sum_\alpha \psi_\alpha^*(x'')\psi_\alpha(x) \int_{-\infty}^{\infty} dE\, \frac{n(E)}{E - E_\alpha^{(0)}}$$
$$= \int_{-\infty}^{\infty} dE\, f(E) \tag{1070}$$

with $f(E) = \sum_\alpha \psi_\alpha^*(x'')\psi_\alpha(x)n(E)/(E - E_\alpha^{(0)})$. Note that the argument $f(E)$ of this integral, introduced by Eq. (1070), has singularities (poles) at the energy values $E = E_\alpha^{(0)}$. To evaluate that integral we introduce the complex 'energy' $Z = Z_r + iZ_i$, with $Z_r = E$, and the Cauchy principal value $P[f(Z)]$ (*i.e.*, integral over the real axis excluding each singularity with an exclusion radius ϵ), which converges to the desired integral in the limit with $\epsilon \to 0$ (see Fig. (1)). We note that
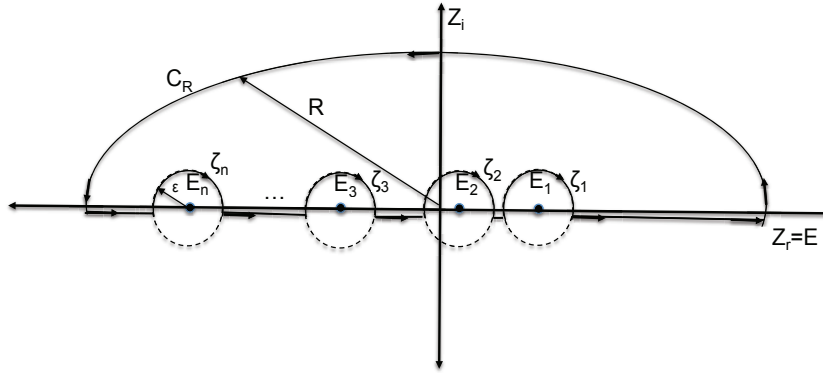
Figure 1: Poles of the integrand of Eq. (1070) at $E = E_\alpha^{(0)}$ and integration contour.

according to the residue theorem, $\oint_C dZ f(Z) = 0$, since the overall integration contour C (*i.e.*, including the real axis excluding the singularities, the half-circles around the singularities and the big half-circle C_R) encloses an area without singularities. In addition,

$$\oint_C dZ f(Z) = P[f(Z)] + \frac{1}{2}\sum_\alpha \oint_{\zeta_\alpha} dZ f(Z) + \oint_{C_R} dZ f(Z). \tag{1071}$$

Since the full-circle contour integrals around the poles are:

$$\oint_{\zeta_\alpha} f(Z)dZ = 2\pi i \, \text{Res}[f, E_\alpha], \tag{1072}$$

with $\text{Res}[f, E_\alpha] = n(E_\alpha)\psi_\alpha^*(x'')\psi_\alpha(x)$ and the integral over C_R vanishes for $R \to \infty$, we obtain:

$$
\begin{aligned}
\int_{-\infty}^{\infty} dE \, n(E)G(x, x''; E) &= P[f(Z)] \\
&= -i\pi \sum_\alpha \text{Res}[f, E_\alpha] \\
&= -i\pi \sum_\alpha n(E_\alpha)\psi_\alpha^*(x'')\psi_\alpha(x) \\
&= -i\pi \rho(x, x'')
\end{aligned} \tag{1073}
$$

which is equivalent to Eq. (1068).

53.2 Factorization

Vector-vector multiplication: To illustrate the benefit of factorization, we first consider the problem of computing $\mathbf{a} \cdot \mathbf{b} + \mathbf{a} \cdot \mathbf{c}$, where $\mathbf{a} = (a_1, \cdots, a_n)$, $\mathbf{b} = (b_1, \cdots, b_n)$ and $\mathbf{c} = (c_1, \cdots, c_n)$ are n-dimensional vectors. Such a computational task requires 2 internal products, each of them with n products and $(n-1)$ sums, or $n + (n-1) = (2n-1)$ operations, plus the sum of the resulting internal products, giving a total of $2(2n-1) + 1 = (4n-1)$ operations.

Factorization, as follows: $\mathbf{a} \cdot (\mathbf{b} + \mathbf{c})$, reduces the computational cost since it allows one to perform the same calculation by first computing the sum $(\mathbf{b}+\mathbf{c})$ in n operations and then computing the internal product of vector $\mathbf{a}$ and the resulting sum $(\mathbf{b} + \mathbf{c})$ in $(2n-1)$ operations, requiring a total of $(3n-1)$ operations.

Matrix-vector multiplication: Analogously, we analyze the number of operations necessary to compute the product of an $n \times n$ matrix $\mathbf{M}$ and a vector $\mathbf{w} = (w_1, \cdots, w_n)$, as follows: $\mathbf{x} = \mathbf{M}\mathbf{w}$,

$$x_j = \sum_{k=1}^{n} M_{j,k} w_k. \tag{1074}$$

To obtain each one of the n elements of $\mathbf{x}$, we need n multiplications plus $(n-1)$ sums. Computing all of the elements of $\mathbf{x}$ thus requires $\mathcal{O}(n^2)$ operations –*i.e.*, a total of $n(2n-1) = 2n^2 - n$ operations.

Factorization of $\mathbf{M}$ as the tensor product $\mathbf{M} = \mathbf{v} \otimes \mathbf{u}$, where $\mathbf{v} = (v_1, \cdots, v_n)$ and $\mathbf{u} = (u_1, \cdots, u_n)$,

$$\mathbf{M} = \begin{pmatrix} v_1 u_1 & v_2 u_1 & \cdots & v_n u_1 \\ v_1 u_2 & v_2 u_2 & \cdots & v_n u_2 \\ \cdots & \cdots & \cdots & \cdots \\ v_1 u_n & v_2 u_n & \cdots & v_n u_n \end{pmatrix}, \tag{1075}$$

defines the matrix elements, as follows $M_{jk} = v_j u_k$, so

$$x_j = v_j \sum_{k=1}^{n} u_k w_k. \tag{1076}$$

To compute all of the elements of $\mathbf{x}$, we need to compute the sum $\sum_{k=1}^{n} u_k w_k$ only once, and then mutiply that sum by each of the corresponding elements of $\mathbf{v}$. Since computing the sum requires $(2n-1)$ operations, the total number of operations to compute $\mathbf{x}$ is $\mathcal{O}(n)$ –*i.e.*, a total of $(3n-1)$ operations.

We now consider the particular case when $\mathbf{v}$ and $\mathbf{u}$ are factorizable as tensor products of m-dimensional vectors, with $n = m^2$. Considering $\mathbf{v} = \tilde{\mathbf{v}}_1 \otimes \tilde{\mathbf{v}}_2$ and $\mathbf{u} = \tilde{\mathbf{u}}_1 \otimes \tilde{\mathbf{u}}_2$, with $\tilde{\mathbf{u}}_j = (\tilde{u}_{1j}, \cdots, \tilde{u}_{mj})$ and $\tilde{\mathbf{v}}_j = (\tilde{v}_{1j}, \cdots, \tilde{v}_{mj})$, we obtain:

$$x_j = \sum_{k=1}^{n} v_j u_k w_k,$$

$$x_{(j_1-1)m+j_2} = \sum_{k_1=1}^{m} \sum_{k_2=1}^{m} \tilde{v}_{j_1 1} \tilde{v}_{j_2 2} \tilde{u}_{k_1} \tilde{u}_{k_2} w_{(k_1-1)m+k_2}, \tag{1077}$$

where $j = (j_1 - 1)m + j_2$ and $k = (k_1 - 1)m + k_2$.

Computing each x_j without factorizing any term of the sum introduced by Eq. (1077), thus requires $4m^2$ products plus $m^2 - 1$ sums. Therefore, the total number of operations necessary to compute the m^2 elements of $\mathbf{x}$ is $\mathcal{O}(m^4)$ –i.e., a total of $m^2(4m^2 + m^2 - 1) = (5m^4 - m^2)$ operations.

Factorizing common terms in the sum, we obtain:

$$x_{(j_1-1)m+j_2} = \tilde{v}_{j_1 1} \tilde{v}_{j_2 2} \sum_{k_1=1}^{m} \tilde{u}_{k_1} \sum_{k_2=1}^{m} \tilde{u}_{k_2} w_{(k_1-1)m+k_2} \tag{1078}$$

To compute all of the elements of $\mathbf{x}$, according to Eq. (1078), we need to compute the product of sums $\sum_{k_1=1}^{m} \tilde{u}_{k_1} \sum_{k_2=1}^{m} \tilde{u}_{k_2} w_{(k_1-1)m+k_2}$ only once, and then mutiply that result by each value of $\tilde{v}_{j_2 2}$ to obtain a weighted vector $\tilde{v}_{j_2 2} = \tilde{v}_{j_2 2} \sum_{k_1=1}^{m} \tilde{u}_{k_1} \sum_{k_2=1}^{m} \tilde{u}_{k_2} w_{(k_1-1)m+k_2}$, requiring m products. Finally, each element of $\mathbf{x}$ is obtained by multiplying $\tilde{v}_{j_1 1}$ and $\tilde{v}_{j_2 2}$, requiring m^2 products. Computing the product of sums requires m products plus $(m-1)$ sums plus one product, for each k_1, times m values of k_1 plus $(m-1)$ sums. The product of sums thus requires a total of $m - 1 + m(m + (m-1) + 1) = (m - 1 + 2m^2)$ operations. Therefore, the total number of operations necessary to compute the m^2 elements of $\mathbf{x}$ is $\mathcal{O}(m^2)$ –i.e., a total of $m - 1 + 2m^2 + m + m^2 = (3m^2 + 2m - 1)$ operations.

We note that factorization thus introduces substantial savings in the number of operations necessary for multiplications of matrices and vectors. In addition, we note that while the matrix is $m^2 \times m^2$, the elements of the matrix are never computed and we only need to allocate memory for arrays of length m^2.

53.3 Schmidt Decomposition

In the most general form, any $m \times n$ matrix $\mathbf{M}$ can be written as a sum of tensor products:

$$M = \sum_{l=1}^{m} \lambda_l \mathbf{u}_l \otimes \mathbf{v}_l, \tag{1079}$$

with elements $M_{jk} = \sum_{l=1}^{m} \lambda_l u_{jl} v_{kl}^T$. Here, u_{jl} are the elements of the $m \times m$ matrix U composed of the column eigenvectors of MM^T, v_{jl} are the elements of the $n \times n$ matrix V composed of the column eigenvectors of $M^T M$, and λ_l are the square roots of eigenvalues from either U or V in descending order –i.e., the non-zero eigenvalues of U and V are always the same.

Equation (1079) is known as the *Schmidt decomposition*, which is another way of presenting the *singular value decomposition* (SVD) of $\mathbf{M} = \mathbf{USV}^T$, defined by the sum of products of elements of the matrices $\mathbf{U}$, $\mathbf{S}$ and $\mathbf{V}$, as follows:

$$M_{jk} = \sum_{l=1}^{m} u_{jl} \sum_{m=1}^{n} s_{lm} v_{mk}^T, \tag{1080}$$

where $s_{lm} = \lambda_l \delta_{lm}$ define the non-zero elements of the $m \times n$ matrix S giving

$$
\begin{aligned}
M_{jk} &= \sum_{l=1}^{m} u_{jl} \lambda_l v_{lk}^T, \\
&= \sum_{l=1}^{m} u_{jl} \lambda_l v_{kl}.
\end{aligned}
\tag{1081}
$$

53.4 Higher Order Singular Value Decomposition

Factorization is elegantly exploited by the so-called the higher-order singular value decomposition (HOSVD) of the d-mode tensor, defined by the following Tucker product:

$$
A(x_1, x_2, \cdots, x_n) = \sum_{\alpha_1, \alpha_2, \cdots, \alpha_d} U(x_1, \alpha_1) U(x_2, \alpha_2) \cdots U(x_n, \alpha_n) S(\alpha_1, \alpha_2, \cdots, \alpha_d)
\tag{1082}
$$

which can be written in a more compact form, as follows:

$$
A = S \times_1 U_1 \times_2 U_2 \cdots \times_d U_d.
\tag{1083}
$$

The single mode matrices U_k are obtained from the SVD's of the k-mode unfoldings of A, as follows:

$$
A_{(k)} = U_k \sigma V_k^T,
\tag{1084}
$$

which can be computed with Matlab, as follows:

$$
[Uk, \sigma, Vk] = \text{svd}(\text{matricize}(A, k)),
\tag{1085}
$$

where $Uk = U_k(x_k, \alpha_k)$ is an $n_k \times r_k$ matrix, with n_k the number of possible values of x_k and r_k the number of diagonal values of σ larger than a desired precision ϵ. Note that Matlab returns $Vk = V_k$, not V_k^T. The d-mode core tensor S is obtained, according to Eq. (1083), as follows:

$$
S = A \times_1 U_1^{-1} \times_2 U_2^{-1} \cdots \times_d U_d^{-1},
\tag{1086}
$$

where $U_j^{-1} = U_j^T$.

A Matlab function for computation of the HOSVD can be defined, as follows:

```
function [S,U] = HOSVD(A)
% A is an n(1) -by -... -by -n(d) tensor .
% U is a length -d cell array with the
% property that U{k} is the left singular
% vector matrix of A's mode -k unfolding .
% S is an n(1) -by -... -by -n(d) tensor given by
% A x1 U{1} x2 U{2} ... xd U{d}

S = A;
```

```
l=length(size(A));
 for k=l:-1:1
     C = tenmat(A,k);
     [U{k},Sigma,V] = svd(C.data);
     S = ttt(conj(U{k}'),S,2,l);
 end
```

and the recovery of the tensor from its HOSVD can be performed, as follows:

```
function [A] = rHOSVD(S,U)
% recovery of A: an n(1) -by -... -by -n(d) tensor .
% U is a length -d cell array with the
% property that U{k} is the left singular
% vector matrix of A's mode -k unfolding .
% S is an n(1) -by -... -by -n(d) tensor given by
% A x1 U{1} x2 U{2} ... xd U{d}

A=S;
l=length(size(A));
 for k=l:-1:1
     A = ttt(conj(U{k}),A,2,l);
 end
```

which are tested by the following code, as follows:

```
clear;
rng(1000);

epsilon=0.0;
n=[5,3,2,4];

% real part of tensor to be decomposed
Ar = randi([0,10],n(1),n(2),n(3),n(4));
% imaginary part tensor to be decomposed
Ai = randi([0,10],n(1),n(2),n(3),n(4));
A=Ar+i*Ai;
%
% Testing HOSVD for complex tensors
 [Sho,Uho] = HOSVD(A);
 [AAA_rec] = rHOSVD(Sho,Uho);
```

53.5   Hierarchical Tucker Format

Analogously to the HOSVD, the h-tucker decomposition represents A according to a product of the single mode matrices, as follows: $A = S \times_1 U_1 \times_2 U_2 \cdots \times_d U_d$, where the matrices U_j are obtained as the left matrices generated by svd after unfolding A with respect to each mode, just

as in the HOSVD. The core tensor S is decomposed as a contraction of 2- and 3-mode transfer tensors which are generated as in the HOSVD after hierarchical matricization. of A For example, the core S for a 4-mode tensor $A(x_1, x_2, x_3, x_4)$ (i.e., $d = 4$), is defined as the contraction $S_{ijkl} = \sum_{\alpha,\beta} B(2)_{ij\alpha} B(1)_{\alpha\beta} B(3)_{kl\beta}$. The 2-mode tensor $B(1)$ is generated by matricization of A with respect to modes 1 and 2 and computing the svd to obtain the left frame matrix $U12$ and then doing the same thing for modes 3 and 4 we obtain $U34$, that can be used to compute the root transfer matrix $B(1)$, as follows:

$$B(1) = A_{(12)} \times_1 U12^T \times_2 U34^T, \tag{1087}$$

The transfer tensors $B(2)$ and $B(3)$ are computed analgously, as follows:

$$B(2) = U12 \times_1 U_1^T \times_2 U_2^T,$$
$$B(3) = U34 \times_1 U_3^T \times_2 U_4^T. \tag{1088}$$

A Matlab implementation of the h-tucker decomposition of a 4-mode tensor, and comparison to the corresponding h-tucker decomposition generated by available subroutines, is given in the following lines:

```
clear;
rng(1000);

epsilon=0.0;
n=[5,3,2,4];

% real part of tensor to be decomposed
Ar = randi([0,10],n(1),n(2),n(3),n(4));
% imaginary part tensor to be decomposed
Ai = randi([0,10],n(1),n(2),n(3),n(4));
A=Ar+i*Ai;
%
% my version of h-tucker decomposition
%
% first level of frames
A12= matricize(A,[1 2]);
nr=rank(A12);
[B1,UV12] = HOSVD(A12);
% check for recovery
%[A12_recv] = rHOSVD(B1,UV12);

% second level of matricization reaches the leaves UU
[UU] = LL(A);
% left frame
U12m=reshape(UV12{1},[n(1),n(2),nr]);
% left transfer tensor
B2 = ttt(conj(UU{1}'),ttt(conj(UU{2}'),U12m,2,2),2,2);
```

238

```
% check for recovery of U12m
U12m_rec = ttt(conj(UU{1}),ttt(conj(UU{2}),B2,2,2),2,2);

% right frame
U34m=reshape(UV12{2},[n(3),n(4),nr]);
% right transfer tensor
B3 = ttt(conj(UU{3}'),ttt(conj(UU{4}'),U34m,2,2),2,2);
% check for recovery of U34m
U34m_rec = ttt(conj(UU{3}),ttt(conj(UU{4}),B3,2,2),2,2);
% check for recovery of A
A_recc = ttt(conj(U12m_rec),ttt(conj(U34m_rec),B1,3,2),3,3);


%
% compare to h-tucker output
%
 opts.max_rank=1000;
 opts.rel_eps=1.0e-12;
 h_A_t = htensor.truncate_rtl(A,opts);

 % check for recovery of U12m
 U12m_rect = ttt(conj(h_A_t.U{4}),ttt(conj(h_A_t.U{5}),h_A_t.B{2},2,2),2,2);
 % check for recovery of U34m
 U34m_rect = ttt(conj(h_A_t.U{6}),ttt(conj(h_A_t.U{7}),h_A_t.B{3},2,2),2,2);
 % check for recovery of A
 A_recct = ttt(conj(U12m_rect),ttt(conj(U34m_rect),h_A_t.B{1},3,2),3,3);
```

where the function [U] = LL(A) is defined analogously to the HOSVD but without computation of the core, as follows:

```
function [U] = LL(A)
% A is an n(1) -by -... -by -n(d) tensor .
% U is a length -d cell array with the
% property that U{k} is the left singular
% vector matrix of A's mode -k unfolding .

l=length(size(A));
 for k=l:-1:1
     C = tenmat(A,k);
     [U{k},Sigma,V] = svd(C.data);
 end
```

An example of the h-tucker representation and reconstruction of a Gaussian, initially repre-sented on a grid, and a Pade time-evolution operator for a 1-dimensional Harmonic oscillator, described by a Fourier grid Hamiltonian, is provided by the notebook

```
htucker_HO.m
```
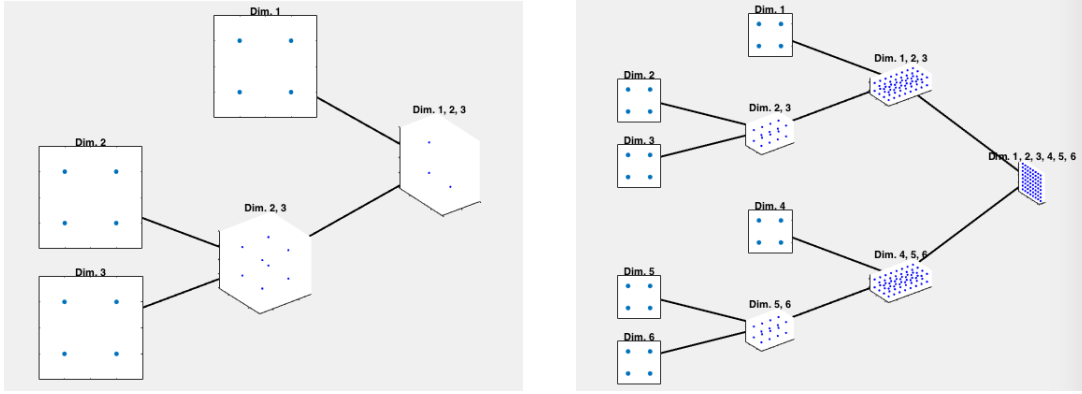
Figure 2: Left: htucker representation of a Gaussian, initially represented on a grid, as a sum of 8 delta functions. The array of the 8 expansion coefficients was first shaped as a $2 \times 2 \times 2$ tensor and then represented in the htucker form by 2 successive SVD's. Right: htucker representaiton of a Pade propagator for a harmonic oscillator, initially given in as an 8×8 matrix. The 64 entries of the matrix were first shaped as a $2 \times 2 \times 2 \times 2 \times 2 \times 2$ tensor and then decomposed by 3 successive SVD's.

included in the tarball file that can be downloaded from here.

The propagation of a 1-d harmonic oscillator, implementing the hierarchical Tucker representation in conjunction with the Pade propagator is provided by the notebook

hos_tucker_c.m

in the same tarball file.

53.6   Tensor Train Decomposition

A simple approach for decomposing a d-mode tensor as a contraction of one-mode tensors is given by the so-called tensor-train decomposition. The tensor is first unfolded with respect to mode 1, and the svd is computed to obtain the first left leaf U_1, as follows:

$$[U_1, S_1, V_1] = svd(matricize(A, 1)). \tag{1089}$$

Then, the matrix SV_1^T is matricized with respect to mode 2 and the rank(S_1), and the second left leaf U_2 is obtained by svd. The process is repeated $d-1$ times to obtain $U_1, \cdots U_{d-1}$. The last leaf is defined from the previous svd, as follows: $U_d = S_{d-1}V_{d-1}^T$.

A Matlab implementation of the tensor train decomposition is given in the following lines:

```
function [ G ] = my_ttrain( A,n,epsilon)
% My version of tensor train decomposition of A with tolerance epsilon
```

```
d=ndims(A);      % # dimensions of A
r=ones(d,1);     % ranks of each dimension
NN=numel(A);     % total number of elements of A
% svd truncation constant
delta = epsilon*norm(reshape(A,[1,numel(A)]))/sqrt(d-1);

for k=1:(d-1)
    if(k == 1)
        VW = reshape(A,[n(k),NN/n(k)]);
    else
        VW=reshape(SVT,[r(k-1)*n(k),NN/(r(k-1)*n(k))]);
    end
    [U,S,V]=svd(VW);
    nsv=0;
    for i=1:rank(S)
        if(S(i,i) > delta)
            nsv=nsv+1;            % # of singular values > epsilon
        end
    end
    for i=1:nsv
        U1{k}(:,i)=U(:,i);
        V1{k}(:,i)=V(:,i);
        for j=1:nsv
            S1{k}(j,i)=S(j,i);
        end
    end
    r(k)=nsv;
    if(k == 1)
        G{k} = reshape(U1{k},[n(k),r(k)]);
    else
        G{k} = reshape(U1{k},[r(k-1),n(k),r(k)]);
        NN = NN*r(k)/(n(k)*r(k-1));
    end
    SVT=ttt(S1{k},V1{k}',2,1);
end
G{d}=SVT;
end
```

which can be tested by the following example:

```
clear;
rng(1000);

epsilon=0.0;
n=[5,3,2,4];
```

```
% real part of tensor to be decomposed
Ar = randi([0,10],n(1),n(2),n(3),n(4));
% imaginary part tensor to be decomposed
Ai = randi([0,10],n(1),n(2),n(3),n(4));
A=Ar+i*Ai;
%
% my version of train tensor decomposition
%
[G] = my_ttrain(A,n,epsilon);
% reconstruct A from my tt decomposision
A_rec_tt=G{1};
d=length(G);
for k=1:(d-1)
    A_r1 = ttt(conj(A_rec_tt),G{k+1},k+1,1);
    A_rec_tt=A_r1;
end
```

53.7 Time-Evolving Block Decimation

Another example of factorization, as applied to quantum dynamics simulations, is the so-called *time-evolving block decimation method* that could be applied to the simulation of dynamics of a $2d$-dimensional system described by the Henon-Heiles Hamiltonian:

$$
\begin{aligned}
\hat{H} &= -\frac{\hbar^2}{2m} \sum_{j=1}^{2d} \frac{\partial^2}{\partial x_j^2} + \frac{m\omega^2}{2} \sum_{j=1}^{2d} x_j^2 + \lambda \sum_{j=1}^{2d-1} \left(x_j^2 x_{j+1} - \frac{1}{3} x_{j+1}^3 \right), \\
&= \sum_{j=1}^{2d} \hat{h}_j + \sum_{j=1}^{d} \hat{V}_{2j-1,2j} + \sum_{j=1}^{d-1} \hat{V}_{2j,2j+1},
\end{aligned}
\tag{1090}
$$

where

$$
\hat{h}_j = -\frac{\hbar^2}{2m} \frac{\partial^2}{\partial x_j^2} + \frac{m\omega^2}{2} x_j^2,
\tag{1091}
$$

and

$$
\hat{V}_{j,j+1} = \lambda \left(x_j^2 x_{j+1} - \frac{1}{3} x_{j+1}^3 \right).
\tag{1092}
$$

The initial state is assumed to be defined, as follows:

$$
\psi(\mathbf{x}; 0) = \sum_k \lambda_k \prod_{j=1}^{2d} \phi_j^k(x_j),
\tag{1093}
$$

where $\phi_j^k(x_j)$ are linear combinations of basis of functions such as delta functions on a grid, or Gaussians.

242

The time-evolved state $\psi(\mathbf{x}; \tau)$ is obtained by using the splitting approximation of the time-evolution operator, according to the three sums introduced by Eq. (1090), as follows:

$$
\begin{aligned}
\psi(\mathbf{x}; \tau) &= \sum_k \lambda_k \prod_{l=1}^{d-1} e^{-\frac{i}{\hbar}\hat{V}_{2l,2l+1}\tau} \prod_{j=1}^{d} e^{-\frac{i}{\hbar}\hat{V}_{2j-1,2j}\tau} \prod_{j'=1}^{d} \langle x_{2j'-1}|e^{-\frac{i}{\hbar}\hat{h}_{2j'-1}\tau}|\phi_{2j'-1}^k\rangle \langle x_{2j'}|e^{-\frac{i}{\hbar}\hat{h}_{2j'}\tau}|\phi_{2j'}^k\rangle, \\
&= \sum_k \lambda_k \prod_{l=1}^{d-1} e^{-\frac{i}{\hbar}\hat{V}_{2l,2l+1}\tau} \prod_{j=1}^{d} \langle x_{2j-1}x_{2j}|e^{-\frac{i}{\hbar}\hat{V}_{2j-1,2j}\tau}|u_{2j-1}^k v_{2j}^k\rangle,
\end{aligned}
\tag{1094}
$$

where

$$
\begin{aligned}
|u_{2j-1}^k\rangle &= e^{-\frac{i}{\hbar}\hat{h}_{2j-1}\tau}|\phi_{2j-1}^k\rangle, \\
|v_{2j}^k\rangle &= e^{-\frac{i}{\hbar}\hat{h}_{2j}\tau}|\phi_{2j}^k\rangle.
\end{aligned}
\tag{1095}
$$

Implementing a SVD, according to Eq. (1081), we obtain:

$$
e^{-\frac{i}{\hbar}V_{2j-1,2j}(x_{2j-1},x_{2j})\tau} \circ \langle x_{2j-1}|u_{2j-1}^k\rangle \otimes \langle x_{2j}|v_{2j}^k\rangle = \sum_{\alpha} \lambda_\alpha \tilde{u}_{2j-1}^k(x_{2j-1}, \alpha)\tilde{v}_{2j}^k(x_{2j}, \alpha), \tag{1096}
$$

where $\circ$ denotes the Hadamard product of matrices $(\mathbf{A} \circ \mathbf{B})_{ij} = A_{ij}B_{ij}$, giving:

$$
\begin{aligned}
\psi(\mathbf{x}; \tau) &= \sum_k \lambda_k \sum_{\alpha_1 \cdots \alpha_d} \lambda_{\alpha_1}, \cdots \lambda_{\alpha_d} \prod_{l=1}^{d-1} e^{-\frac{i}{\hbar}\hat{V}_{2l,2l+1}\tau} \prod_{j=1}^{d} \tilde{u}_{2j-1}^k(x_{2j-1}, \alpha_j)\tilde{v}_{2j}^k(x_{2j}, \alpha_j), \\
&= \sum_k \lambda_k \sum_{\alpha_1 \cdots \alpha_d} \lambda_{\alpha_1} \cdots \lambda_{\alpha_d} \tilde{u}_1^k(x_1, \alpha_1)\tilde{v}_{2d}^k(x_{2d}, \alpha_d) \prod_{l=1}^{d-1} e^{-\frac{i}{\hbar}\hat{V}_{2l,2l+1}\tau} \prod_{j=1}^{d-1} \tilde{v}_{2j}^k(x_{2j}, \alpha_j)\tilde{u}_{2j+1}^k(x_{2j+1}, \alpha_{j+1}), \\
&= \sum_k \lambda_k \sum_{\alpha_1 \cdots \alpha_d} \lambda_{\alpha_1} \cdots \lambda_{\alpha_d} \tilde{u}_1^k(x_1, \alpha_1)\tilde{v}_{2d}^k(x_{2d}, \alpha_d) \prod_{j=1}^{d-1} e^{-\frac{i}{\hbar}\hat{V}_{2j,2j+1}\tau} \tilde{v}_{2j}^k(x_{2j}, \alpha_j)\tilde{u}_{2j+1}^k(x_{2j+1}, \alpha_{j+1}).
\end{aligned}
\tag{1097}
$$

Implementing another SVD, according to Eq. (1081), we obtain:

$$
e^{-\frac{i}{\hbar}V_{2j,2j+1}(x_{2j+1},x_{2j})\tau} \circ \langle x_{2j}|\tilde{v}_{2j}^{k,\alpha_j}\rangle \otimes \langle x_{2j+1}|u_{2j+1}^{k,\alpha_{j+1}}\rangle = \sum_{\alpha_j'} \lambda_{\alpha_j'} \tilde{v}_{2j}^{k,\alpha_j}(x_{2j}, \alpha_j')\tilde{u}_{2j+1}^{k,\alpha_{j+1}}(x_{2j+1}, \alpha_j'), \tag{1098}
$$

giving the matrix-product state

$$
\begin{aligned}
\psi(\mathbf{x}; \tau) &= \sum_k \lambda_k \sum_{\alpha_1 \cdots \alpha_d} \sum_{\alpha_1' \cdots \alpha_{d-1}'} \tilde{u}_1^k(x_1, \alpha_1)\lambda_{\alpha_1}\tilde{v}_2^{k,\alpha_1}(x_2, \alpha_1')\lambda_{\alpha_1'}\tilde{u}_3^{k,\alpha_2}(x_3, \alpha_1') \cdots \\
&\quad \times \tilde{v}_{2d-2}^{k,\alpha_{d-1}}(x_{2d-2}, \alpha_{d-1}')\lambda_{\alpha_{d-1}'}\tilde{u}_{2d-1}^{k,\alpha_d}(x_{2d-1}, \alpha_{d-1}')\lambda_{\alpha_d}\tilde{v}_{2d}^k(x_{2d}, \alpha_d),
\end{aligned}
\tag{1099}
$$
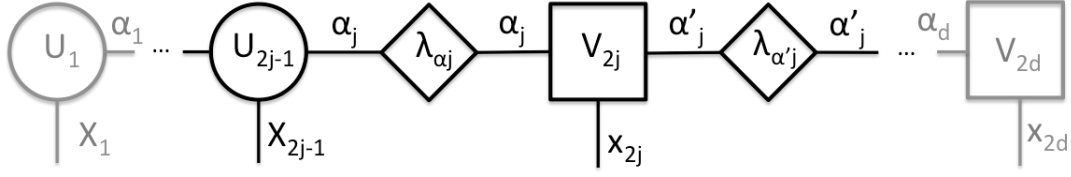
represented by the Penrose-like diagram of Fig. 53.7, where λ are vectors while $\tilde{u}^k$ and $\tilde{v}^k$ are tensors.

Subsequent propagation steps are implemented analogously, although including a sum in the l.h.s. of the SVD's, over index α or α' common to the states $\tilde{u}$ and $\tilde{v}$, as follows:

$$
e^{-\frac{i}{\hbar}V_{2j,2j+1}(x_{2j+1},x_{2j})\tau} \circ \sum_{\alpha'} \lambda_{\alpha'} \langle x_{2j}|\tilde{v}_{2j}^{k,\alpha_j}(\alpha')\rangle \otimes \langle x_{2j+1}|u_{2j+1}^{k,\alpha_{j+1}}(\alpha')\rangle
$$

$$
= \sum_{\alpha'_j} \tilde{\lambda}_{\alpha'_j} \tilde{v}_{2j}^{k,\alpha_j}(x_{2j},\alpha'_j)\tilde{u}_{2j+1}^{k,\alpha_{j+1}}(x_{2j+1},\alpha'_j). \tag{1100}
$$

54   Sokhotsky Formula

The goal of this section is to prove the Sokhotsky formula, introduced as follows:

$$\lim_{\epsilon \to 0^+} \frac{1}{x \pm i\epsilon} = \text{p.v.} \frac{1}{x} \mp i\pi\delta(x), \tag{1101}$$

where the Cauchy principal value (p.v.) transforms smooth and rapidly decaying functions $f(x)$ (*i.e.*, Schwartz functions), as follows:

$$\int dx \ \text{p.v.} \frac{1}{x} f(x) = \lim_{\epsilon \to 0^+} \left[\int_{-\infty}^{-\epsilon} dx \frac{f(x)}{x} + \int_{-\epsilon}^{\infty} dx \frac{f(x)}{x} \right]. \tag{1102}$$

Analogously, the delta function transforms $f(x)$, as follows:

$$\int dx \ \delta(x) f(x) = f(0). \tag{1103}$$

Therefore, multiplying both sides of Eq. (1101) by an analytic function $f(x)$ and integrating over x, we obtain:

$$\lim_{\epsilon \to 0^+} \int dx \frac{f(x)}{x \pm i\epsilon} = \int dx \, \text{p.v.} \frac{1}{x} f(x) \mp i\pi f(0), \tag{1104}$$

To prove Eq. (1104), we multiply and divide the integrand in the l.h.s by $x \mp i\epsilon$, as follows:

$$\lim_{\epsilon \to 0^+} \int dx \frac{f(x)}{x \pm i\epsilon} = \int dx \frac{f(x)}{x} \lim_{\epsilon \to 0^+} \frac{x^2}{x^2 + \epsilon^2} \mp i\pi \int dx f(x) \lim_{\epsilon \to 0^+} \frac{\epsilon}{\pi(x^2 + \epsilon^2)}$$
$$= \int dx \frac{f(x)}{x} \lim_{\epsilon \to 0^+} \frac{x^2}{x^2 + \epsilon^2} \mp i\pi \int dx f(x) \delta(x) \tag{1105}$$

where $\frac{x^2}{x^2 + \epsilon^2} = 1$ when $x \gg \epsilon$ and $\frac{x^2}{x^2 + \epsilon^2} = 0$ when $x \to 0$ and $x \ll \epsilon$. Therefore, the integral becomes the Cauchy principal value.

55   Inverse Design by Constrained Optimization

This section applies the constrained optimization methodology to the problem of inverse design of molecular structures. We consider a molecular system described by the linear-combination-of-atomic-potential (LCAP) Hamiltonian,

$$H_{ij}(t) = \sum_{\alpha=1}^{N_{type}^j} \sum_{\alpha'=1}^{N_{type}^i} p_i^\alpha(t) p_j^{\alpha'}(t) h_{i,j}^{\alpha,\alpha'}, \tag{1106}$$

with participation coefficients p_i^α giving the probability weight for various possible atom types α in each atomic site i of the molecule. Real molecules have participation coefficients $p_i^\alpha = 1$, or 0,

defining pure atomic types. However, they can change in time *in-silico* and assume intermediate values in the 0–1 range while the molecule undergoes transformations in the alchemist space of superposition states. The main goal of the inverse design methodology is to evolve the participation coefficients to optimize the expectation value of a property of interest. Upon convergence, the resulting optimized state typically has fractional participation coefficients that are rounded off to obtain the closest real molecule with optimized properties.

The main advantage of this approach when compared to random search procedures is determined by the relatively modest scalability of the optimization of participation coefficients. Although computationally demanding, these calculations rely upon calculations of time-dependent expectation values at a final propagation time t_f,

$$\bar{O}(t_f) = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{\alpha=1}^{N_{type}^i} \sum_{\alpha'=1}^{N_{type}^j} S_{\alpha,\beta}^{i,j} B_{i,\alpha}^*(t_f) B_{j,\alpha'}(t_f) O_{i,j}^{\alpha,\alpha'}, \tag{1107}$$

after the system has evolved according to the time-dependent wave function $\psi(t)$, typically expanded as a linear combination of atomic orbitals $|j,\alpha\rangle$,

$$|\psi(t)\rangle = \sum_{j,\alpha} B_{j,\alpha}(t)|j,\alpha\rangle, \tag{1108}$$

where the time-dependent expansion coefficients are computed as follows:

$$B_{j,\alpha}(t) = \sum_{k} Q_{j,\alpha}^k C_k e^{-(i/\hbar)E_k t}, \tag{1109}$$

with C_k defining the initial state, while E_k and $Q_{j,\alpha}^k$ are found 'on-the-fly' by solving the generalized eigenvalue problem,

$$HQ^k = E_k SQ^k. \tag{1110}$$

Here, H is the LCAP Hamiltonian matrix, introduced in Eq. (1106), and S is the overlap matrix in the atomic orbital basis. Therefore, calculations of $\bar{O}(t)$ scale as $(N \times N_{type})^2$, with N and N_{type} the number of atoms and atom types, respectively. In contrast, assessing all possible structures would scale exponential, as $N^{N_{type}}$.

The participation coefficients are optimized by using constrained optimization. To illustrate the method, we consider the problem of propagating a state $\psi(t)$ from t_0 to t_f, and we use constrained optimization to maximize $O(t_f) = \langle\psi(t_f)|\hat{O}|\psi(t_f)\rangle$, with $\hat{O} = |\phi_1\rangle\langle\phi_1|$. To obtain the optimal control equations, we expand the LCAP Hamiltonian, as follows:

$$H(t) = H(t_0) - \mu(t) \cdot \varepsilon(t), \tag{1111}$$

where

$$\mu(t) \cdot \varepsilon(t) = \sum_{j=1}^{N} \sum_{\alpha=1}^{N_{type}^j} \mu_j^\alpha(t) \varepsilon_j^\alpha(t), \tag{1112}$$

246

with

$$\varepsilon_j^\alpha (t) = p_j^\alpha (t_0) - p_j^\alpha (t), \tag{1113}$$

and

$$\mu_j^\alpha (t) = \frac{\partial H}{\partial p_j^\alpha}. \tag{1114}$$

We evolve the system according to the Schrödinger equation,

$$i\hbar \frac{\partial}{\partial t} \psi (t) = [H(t_0) - \mu (t) \cdot \varepsilon (t)] \psi (t). \tag{1115}$$

and we optimize $p_j^\alpha(t)$ as externally driven coefficients, subject to minimum changes relative to their initial values $p_j^\alpha(t_0)$. According to the standard methodology of constrained optimization, we maximize the functional

$$\begin{aligned}
K(\psi, p, \gamma, \lambda) = O(t_f) &- \int_{t_i}^{t_f} dt \gamma(t) |\varepsilon(t)|^2 \\
&- 2\mathrm{Im} \int_{t_i}^{t_f} dt \int_{-\infty}^{\infty} dx \lambda(x, t) \left[H(t_0) - \mu(t) \cdot \varepsilon(t) - i\hbar \frac{\partial}{\partial t} \right] \psi(t),
\end{aligned} \tag{1116}$$

where $|\varepsilon(t)|^2 = \sum_{j=1}^{N} \sum_{\alpha=1}^{N_{type}^j} |\varepsilon_j^\alpha(t)|^2$, $\gamma(t)$ are the Lagrange multipliers ensuring minimum changes of the participation coefficients at all times t, and $\lambda(x, t)$ are the Lagrange multipliers ensuring that $\psi(t)$ and $\psi^*(t)$ satisfy Eq. (1115) for all values of x and t. To find the conditions under which the Lagrange multipliers enforce these conditions, we find the extreme of K by making the derivatives of K with respect to ε, ψ and λ equal to zero.[4] For simplicity we make $\gamma(t) = \gamma/s(t)$, where γ is a real constant and $s(t)$ is a given function.

Considering that $2\mathrm{Im}[z] = i(z^* - z)$, the derivative of K with respect to λ gives:

$$\frac{\partial K}{\partial \lambda(t)} = i \int_{t_i}^{t_f} dt \left[H(t_0) - \mu(t) \cdot \varepsilon(t) - i\hbar \frac{\partial}{\partial t} \right] \psi(t) = 0. \tag{1117}$$

Therefore, $\psi(t)$ must satisfy the Schrödinger equation:

$$\boxed{i\hbar \frac{\partial}{\partial t} \psi(t) = [H(t_0) - \mu(t) \cdot \varepsilon(t)] \psi(t)} \tag{1118}$$

with the boundary condition $\psi(t_i) = \phi_0$. Furthermore, the derivative of K with respect to ψ gives:

$$\begin{aligned}
\frac{\partial K}{\partial \psi(t)} &= O\psi^*(t_f)\delta(t - t_f) - [\lambda^*(t)\delta(t - t_f)]|_{t_i}^{t_f} \\
&- i \left(i\hbar \frac{\partial}{\partial t} + [H(t_0) - \mu(t) \cdot \varepsilon(t)] \right) \lambda^*(t) = 0.
\end{aligned} \tag{1119}$$

---

[4]Note that the imaginary part of the third term in Eq. (374) ensures that K remains *real* even when ψ and λ are *complex*, thus, the derivatives with respect to ψ^* and λ^* are simply the complex conjugate of equations for ψ and λ.

since by partial integration we obtain

$$\int\limits_{t_i}^{t_f} \left\langle \lambda(t) \left| i\hbar\frac{\partial}{\partial t} - [H(t_0) - \mu(t) \cdot \varepsilon(t)] \right| \psi(t) \right\rangle dt = i \left\langle \lambda(t) | \psi(t) \right\rangle|_{t_i}^{t_f}$$

$$-i\int\limits_{t_i}^{t_f} \left\langle \hbar\frac{\partial}{\partial t}\lambda(t) | \psi(t) \right\rangle dt - \int\limits_{t_i}^{t_f} \left\langle [H(t_0) - \mu(t) \cdot \varepsilon(t)] \lambda(t) | \psi(t) \right\rangle dt = i \left\langle \lambda(t) | \psi(t) \right\rangle|_{t_i}^{t_f}$$

$$+\int\limits_{t_i}^{t_f} \left\langle \left(i\hbar\frac{\partial}{\partial t} - [H(t_0) - \mu(t) \cdot \varepsilon(t)] \right) \lambda(t) | \psi(t) \right\rangle dt$$

Equation (1119) is satisfied when

$$\boxed{\lambda(t_f) = O\psi(t_f)} \tag{1120}$$

and $\lambda(t)$ satisfies the Schrödinger equation,

$$\boxed{i\hbar\frac{\partial}{\partial t}\lambda(t) = [H(t_0) - \mu(t) \cdot \varepsilon(t)] \lambda(t)} \tag{1121}$$

Finally, the variation of the external field that affects the participation coefficients gives,

$$\frac{\delta K}{\delta \varepsilon} = -2\gamma\frac{|\varepsilon(t)|}{s(t)}$$
$$-2\text{Im}\left\langle \lambda(t) |\mu(t)| \psi(t) \right\rangle = 0, \tag{1122}$$

hence

$$\varepsilon(t) = -\frac{s(t)}{\hbar\gamma} \cdot \text{Im}\left\langle \lambda(t) |\mu(t)| \psi(t) \right\rangle \tag{1123}$$

and thus

$$\boxed{p_j^\alpha(t) = p_j^\alpha(t_0) + \frac{s(t)}{\hbar\gamma} \cdot \text{Im}\left\langle \lambda(t) \left|\mu_j^\alpha(t)\right| \psi(t) \right\rangle} \tag{1124}$$

The boxed equations are the so-called 'control-field equations' since they can be used to obtain the *optimum control field* according to the following iterative procedure. Starting with an initial guess for the field ε, propagate ψ from t_i to t_f, according to Eq. (1118). Having $\psi(t_f)$, we compute $\lambda(t_f)$ according to Eq. (1120) and we propagate it backwards to time t_i according to Eq. (381) to obtain $\lambda(t_i)$. Propagating $\lambda(t_i)$ and $\psi(t_i)$ from t_i to t_f allow us to update $\varepsilon(t)$ according to Eq. (383). The propagation also yields a new $\psi(t_f)$ which can be used to obtain a new $\lambda(t_f)$, and therefore a new $\lambda(t_i)$ by backward propagation. This procedure is repeated to update $\varepsilon(t)$ and $\psi(t_f)$ multiple times until convergence.

56 Solvation of a Point Dipole Morse Oscillator

The goal of this section is to derive the vibrational frequency shifts induced by a dielectric continuum medium solvating a point dipole Morse oscillator.

We start by expanding the Morse potential in terms of the vibrational coordinate Q, as follows:

$$V_M(Q) = D\left[1 - e^{-\alpha Q}\right]^2 = \frac{1}{2}V_M'' Q^2 + \frac{1}{3}V_M''' Q^3 + \cdots, \tag{1125}$$

where D is the dissociation energy, $V_M'' = 2D\alpha^2$, and $V_M''' = -6D\alpha^2$.

We consider that the Morse oscillator interacts with the surrounding solvent as a point dipole $\mu(Q)$, at the center of a spherical cavity of radius a, embedded in a dielectric continuum medium. Therefore, the solvation energy is

$$V_s(Q) = \frac{1}{2}\mu(Q)R(Q) = V_s^0 + V_s' Q + \frac{1}{2}V_s'' Q^2 + \cdots, \tag{1126}$$

where $R(Q)$ is the Onsager reaction field,

$$R(Q) = f(\varepsilon_0)\mu^0 + f(\varepsilon_\infty)\mu' Q + f(\varepsilon_\infty)\mu'' Q^2 + \cdots, \tag{1127}$$

with ε_0 and ε_∞ the static and optical dielectric constants of the surrounding medium and

$$f(\varepsilon) = \frac{1}{a^3}\left(\frac{2\varepsilon - 2}{2\varepsilon + 1}\right), \tag{1128}$$

giving $f(\varepsilon_\infty) = f(\varepsilon_0) + f(\varepsilon_\infty)'(Q - Q^*) + \cdots$, with Q^* the equilibrium position of the solvated Morse oscillator. Substituting Eq. (1127) into Eq. (1126), we obtain:

$$\begin{aligned} V_s(Q) &= \frac{1}{2}\left(\mu^0 + \mu' Q + \mu'' Q^2 + \cdots\right)\left(f(\varepsilon_0)\mu^0 + f(\varepsilon_\infty)\mu' Q + f(\varepsilon_\infty)\mu'' Q^2 + \cdots\right) \\ &= V_s^0 + V_s' Q + \frac{1}{2!}V_s'' Q^2 + \cdots, \end{aligned} \tag{1129}$$

where

$$V_s' = -\frac{1}{2}\left(f(\varepsilon_0) + f(\varepsilon_\infty)\right)\mu^0 \cdot \mu', \tag{1130}$$

and

$$V_s'' = -(f(\varepsilon_0) + f(\varepsilon_\infty))\mu^0 \cdot \mu'' - f(\varepsilon_\infty)\mu' \cdot \mu'. \tag{1131}$$

Therefore, evaluating V_s' at Q^*, we obtain:

$$V_s' = -f(\varepsilon_0)\mu^0 \cdot \mu'. \tag{1132}$$

According to Eqs. (1125) and (1129), the total energy of the solvated Morse oscillator is

$$\begin{aligned} V(Q) &= V_M(Q) + V_s(Q), \\ &= \frac{1}{2!}V_M'' Q^2 + \frac{1}{3!}V_M''' Q^3 + V_s^0 + V_s' Q + \frac{1}{2!}V_s'' Q^2 + \cdots \end{aligned} \tag{1133}$$

defining the equilibrium position Q^*, as follows:

$$V'(Q^*) = 0,$$
$$= V_M'' Q^* + \frac{1}{2!} V_M''' Q^{*2} + V_s' + V_s'' Q^* + \cdots, \tag{1134}$$

giving

$$Q^* \approx -\frac{V_s'}{V_M'' + V_s''}. \tag{1135}$$

The oscillator frequency

$$\omega = \sqrt{\frac{V''(Q^*)}{m}}, \tag{1136}$$

is obtained by computing $V''(Q^*)$ from Eqs. (1133) and (1135), as follows:

$$V''(Q^*) = V_M'' + V_M''' Q^* + V_s'' + \cdots,$$
$$= V_M'' - V_M''' \frac{V_s'}{V_M'' + V_s''} + V_s'' + \cdots \tag{1137}$$

Assuming $V_M'' \gg V_s''$,

$$V''(Q^*) \approx V_M'' \left(1 - \frac{1}{V_M''} \left[\frac{V_M'''}{V_M''} V_s' - V_s'' \right] \right) \tag{1138}$$

Substituting V_s' and V_s'', according to Eqs. (1130) and (1131) respectively, into Eq. (1138), we obtain:

$$V''(Q^*) \approx V_M'' \left(1 + \frac{1}{V_M''} \left[\frac{V_M'''}{V_M''} \frac{1}{2} \left(f(\varepsilon_0) + f(\varepsilon_\infty) \right) \mu^0 \cdot \mu' - (f(\varepsilon_0) + f(\varepsilon_\infty)) \mu^0 \cdot \mu'' - f(\varepsilon_\infty) \mu' \cdot \mu' \right] \right)$$
$$\approx V_M'' \left(1 - \frac{1}{V_M''} \left[f(\varepsilon_0) \left(\mu^0 \cdot \mu'' - \frac{V_M'''}{V_M''} \mu^0 \cdot \mu' \right) + f(\varepsilon_\infty) \left(\mu^0 \cdot \mu'' + \mu' \cdot \mu' \right) \right] \right),$$
$$\approx V_M'' (1 - \Delta),$$
$$\tag{1139}$$

with

$$\Delta = \frac{1}{V_M''} \left[f(\varepsilon_0) \left(\mu^0 \cdot \mu'' - \frac{V_M'''}{V_M''} \mu^0 \cdot \mu' \right) + f(\varepsilon_\infty) \left(\mu^0 \cdot \mu'' + \mu' \cdot \mu' \right) \right]. \tag{1140}$$

Equation (1139) allows us to compute the vibrational frequency shift due to solvation, relative to the frequency of the oscillator in vacuum, as follows:

$$\frac{\Delta \omega}{\omega_0} = \frac{\omega_0 - \omega}{\omega_0}$$
$$= 1 - \sqrt{\frac{V''(Q^*)}{V_M''}} \tag{1141}$$
$$= 1 - \sqrt{(1 - \Delta)} = 1 - \left(1 - \frac{1}{2} \Delta + \cdots \right).$$

Substituting Eq. (1140) into Eq. (1141), we obtain an approximate linear dependence of the vibrational frequency shift with $F(\varepsilon)$,

$$\Delta\omega \approx \frac{1}{2m\omega_0} \left[f(\varepsilon_0) \left(\mu^0 \cdot \mu'' - \frac{V_M'''}{V_M''} \mu^0 \cdot \mu' \right) + f(\varepsilon_\infty) \left(\mu' \cdot \mu' + \mu^0 \cdot \mu'' \right) \right], \tag{1142}$$

suggested by Kirkwood as acknowledged by W. West and R. T. Edwards *J. Chem. Phys.* **5**, 14 (1937). Therefore, we can estimate $\Delta\omega$, as follows:

$$\Delta\omega = S_0 + S_1 f(\varepsilon_0), \tag{1143}$$

where the slope S_1 is

$$S_1 = \frac{1}{2m\omega_0} \left(\mu^0 \cdot \mu'' - \frac{V_M'''}{V_M''} \mu^0 \cdot \mu' \right). \tag{1144}$$

57 EPR Signal of IET

We consider photoexcitation of an adsorbate molecule covalently bound to a semiconductor surface, a process that promotes an electron in the adsorbate from the ground to the excited state. We assume
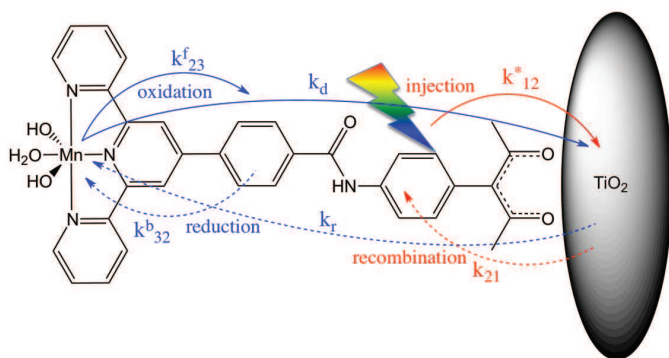
Figure 3: Schematic diagram of the mechanism of of IET probed by EPR spectroscopy, where photoelectron injection creates a hole in the ligand that induces $Mn^{2+} \rightarrow Mn^{3+}$ oxidation.

that the photoexcited state is isoenergetic with an electronic state in the semiconductor conduction band, inducing IET, as follows:

$$1 \rightleftharpoons 2, \tag{1145}$$

The effective rate constant for electron injection is $k_{12} = [p]k_{12}^*$, where $[p]$ is the effective concentration of photons as determined by the intensity of the light. The rate for recombination from **2** to **1** is k_{21}. In addition, we assume that the hole left in the adsorbate oxidizes a redox active center as follows,

$$2 \rightleftharpoons 3, \tag{1146}$$

with an oxidation rate constant k_{23} and recombination k_{32}. In addition, $\mathbf{1}$ forms $\mathbf{3}$ with rate constant k_d and $\mathbf{3}$ recombines into $\mathbf{1}$ with rate constant k_r. Considering that resulting kinetics, we obtain:

$$
\begin{aligned}
\frac{d\mathbf{1}}{dt} &= -(k_{12} + k_d)\mathbf{1} + k_{21}\mathbf{2} + k_r\mathbf{3}, \\
\frac{d\mathbf{2}}{dt} &= k_{12}\mathbf{1} - k_{21}\mathbf{2} - k_{23}\mathbf{2} + k_{32}\mathbf{3}, \\
\frac{d\mathbf{3}}{dt} &= k_{23}\mathbf{2} - (k_{32} + k_r)\mathbf{3} + k_d\mathbf{1},
\end{aligned}
\tag{1147}
$$

When $\mathbf{p}(0) = \mathbf{1}(0) + \mathbf{2}(0) + \mathbf{3}(0)$, we obtain:

$$
\begin{aligned}
\frac{d\mathbf{1}}{dt} &= -(k_{12} + k_d)\mathbf{1} + k_{21}(\mathbf{p}(0) - \mathbf{1} - \mathbf{3}) + k_r\mathbf{3}, \\
\frac{d\mathbf{3}}{dt} &= k_{23}(\mathbf{p}(0) - \mathbf{1} - \mathbf{3}) - (k_{32} + k_r)\mathbf{3} + k_d\mathbf{1},
\end{aligned}
\tag{1148}
$$

with

$$
\begin{aligned}
\frac{d\mathbf{1}}{dt} &= -k_{11}\mathbf{1} + k_{13}\mathbf{3} + k_{21}\mathbf{p}(0), \\
\frac{d\mathbf{3}}{dt} &= -k_{33}\mathbf{3} + k_{31}\mathbf{1} + k_{23}\mathbf{p}(0),
\end{aligned}
\tag{1149}
$$

with $k_{11} = k_d + k_{12} + k_{21}$, $k_{13} = k_r - k_{21}$, $k_{33} = k_r + k_{32} + k_{23}$ and $k_{31} = k_d - k_{23}$.

Solving for $\mathbf{3}$, we obtain:

$$
\begin{aligned}
\mathbf{3} &= k_{13}^{-1}\frac{d\mathbf{1}}{dt} + k_{13}^{-1}k_{11}\mathbf{1} - k_{13}^{-1}k_{21}\mathbf{p}(0), \\
k_{13}^{-1}\frac{d^2\mathbf{1}}{dt^2} + k_{13}^{-1}k_{11}\frac{d\mathbf{1}}{dt} &= k_{31}\mathbf{1} - k_{33}k_{13}^{-1}\frac{d\mathbf{1}}{dt} - k_{33}k_{13}^{-1}k_{11}\mathbf{1} + k_{33}k_{13}^{-1}k_{21}\mathbf{p}(0).
\end{aligned}
\tag{1150}
$$

Associating terms,

$$
\frac{d^2\mathbf{1}}{dt^2} + (k_{11} + k_{33})\frac{d\mathbf{1}}{dt} + (k_{33}k_{11} - k_{13}k_{31})\mathbf{1} - k_{33}k_{21}\mathbf{p}(0) = 0.
\tag{1151}
$$

or

$$
\frac{d^2\mathbf{1}}{dt^2} + A\frac{d\mathbf{1}}{dt} + B\mathbf{1} + C = 0.
\tag{1152}
$$

with $A = k_{11} + k_{33}$, $B = k_{33}k_{11} - k_{13}k_{31}$ and $C = k_{33}k_{21}\mathbf{p}(0)$, with solution

$$
\begin{aligned}
\mathbf{1} &= C_2 e^{-\frac{A - \sqrt{A^2 - 4B}}{2}t} + C_1 e^{-\frac{A + \sqrt{A^2 - 4B}}{2}t} + \frac{C}{B} \\
\frac{d\mathbf{1}}{dt} &= \tilde{C}_2 e^{-\frac{A - \sqrt{A^2 - 4B}}{2}t} + \tilde{C}_1 e^{-\frac{A + \sqrt{A^2 - 4B}}{2}t}.
\end{aligned}
\tag{1153}
$$

where $\tilde{C}_2 = -C_2 \frac{A-\sqrt{A^2-4B}}{2}$ and $\tilde{C}_1 = -C_1 \frac{A+\sqrt{A^2-4B}}{2}$. Therefore,

$$
\begin{aligned}
\mathbf{3} = {} & k_{13}^{-1}(\tilde{C}_2 e^{-\frac{A-\sqrt{A^2-4B}}{2}t} + \tilde{C}_1 e^{-\frac{A+\sqrt{A^2-4B}}{2}t}) + \\
& + k_{13}^{-1} k_{11}(C_2 e^{-\frac{A-\sqrt{A^2-4B}}{2}t} + C_1 e^{-\frac{A+\sqrt{A^2-4B}}{2}t} + \frac{C}{B}) - \\
& - k_{13}^{-1} k_{21}\mathbf{p}(0),
\end{aligned}
\tag{1154}
$$

and gathering terms, we obtain:

$$
\begin{aligned}
\mathbf{3} = {} & (k_{13}^{-1}\tilde{C}_2 + k_{13}^{-1}k_{11}C_2)e^{-\frac{A-\sqrt{A^2-4B}}{2}t} + \\
& + e^{-\frac{A+\sqrt{A^2-4B}}{2}t}(\tilde{C}_1 k_{13}^{-1} + C_1 k_{13}^{-1}k_{11}) + \mathbf{p}(0)\left(\frac{k_{21}k_{31}}{k_{33}k_{11} - k_{13}k_{31}}\right).
\end{aligned}
\tag{1155}
$$

58   Terahertz Signal of IET

We consider photoexcitation of an adsorbate molecule covalently bound to a semiconductor surface,

$$G \rightarrow E, \tag{1156}$$

a process that promotes the an electron in the adsorbate from the ground G to the excited E state. We assume that E is isoenergetic with an electronic state B in the semiconductor conduction band. The rate constant for electron injection:

$$E \rightarrow B \tag{1157}$$

is k_1. In addition, we assume that the injected carriers equilibrate with a trap state C,

$$B \rightleftharpoons C, \tag{1158}$$

with a rate constant of trapping k_2 and detrapping k_3. Considering that resulting kinetics, we obtain:

$$\begin{aligned}
\frac{dE}{dt} &= -k_1 E, \\
\frac{dB}{dt} &= +k_1 E - k_2 B + k_3 C, \\
\frac{dC}{dt} &= +k_2 B - k_3 C.
\end{aligned} \tag{1159}$$

Solving for E from the first equation in Eq. (1159), $E(t) = E(0)e^{-k_1 t}$, we obtain:

$$\begin{aligned}
\frac{dB}{dt} &= k_1 E(0)e^{-k_1 t} - k_2 B + k_3 C, \\
\frac{dC}{dt} &= k_2 B - k_3 C.
\end{aligned} \tag{1160}$$

Rewriting Eq. (1160) in matrix notation, we obtain:

$$\frac{d}{dt}\mathbf{v} = \mathbf{M} \times \mathbf{v} + \mathbf{f}(t), \tag{1161}$$

where

$$\mathbf{v} = \begin{pmatrix} B(t) - B(0) \\ C(t) - C(0) \end{pmatrix}, \quad \mathbf{M} = \begin{pmatrix} -k_2 & k_3 \\ k_2 & -k_3 \end{pmatrix}, \quad \mathbf{f}(t) = \begin{pmatrix} k_1 E(0)e^{-k_1 t} \\ 0 \end{pmatrix}. \tag{1162}$$

Multiplying Eq. (1161) by $e^{-\mathbf{M}t}$, we obtain:

$$\frac{d}{dt}e^{-Mt}\mathbf{v} = e^{-\mathbf{M}t}\mathbf{f}(t). \tag{1163}$$

Integrating Eq. (1163), we obtain:

$$\mathbf{v}(t) = e^{\mathbf{M}t} \int_0^t dt' e^{-\mathbf{M}t'} \mathbf{f}(t').$$

(1164)

In order to operate with the exponential of $\mathbf{M} = \mathbf{M}_d + \mathbf{M}_o$, we rewrite the matrix as the sum of its diagonal $\mathbf{M}_d$ and off-diagonal $\mathbf{M}_o$ components:

$$
\begin{aligned}
e^{-\mathbf{M}t} &= e^{-\mathbf{M}_d t} e^{-\mathbf{M}_o t}, \\
&= \begin{pmatrix} e^{k_2 t} & 0 \\ 0 & e^{k_3 t} \end{pmatrix} e^{-\begin{pmatrix} 0 & k_3 \\ k_2 & 0 \end{pmatrix} t}, \\
&= \begin{pmatrix} e^{k_2 t} & 0 \\ 0 & e^{k_3 t} \end{pmatrix} e^{-\mathbf{D} \begin{pmatrix} -\sqrt{k_2 k_3} & 0 \\ 0 & \sqrt{k_2 k_3} \end{pmatrix} \mathbf{D}^{-1} t}, \\
&= \begin{pmatrix} e^{k_2 t} & 0 \\ 0 & e^{k_3 t} \end{pmatrix} \mathbf{D} \begin{pmatrix} e^{\sqrt{k_2 k_3} t} & 0 \\ 0 & e^{-\sqrt{k_2 k_3} t} \end{pmatrix} \mathbf{D}^{-1}, \\
&= \begin{pmatrix} e^{k_2 t} & 0 \\ 0 & e^{k_3 t} \end{pmatrix} \begin{pmatrix} -\sqrt{\frac{k_3}{k_2+k_3}} & \sqrt{\frac{k_3}{k_2+k_3}} \\ \sqrt{\frac{k_2}{k_2+k_3}} & \sqrt{\frac{k_2}{k_2+k_3}} \end{pmatrix} \begin{pmatrix} e^{\sqrt{k_2 k_3} t} & 0 \\ 0 & e^{-\sqrt{k_2 k_3} t} \end{pmatrix} \begin{pmatrix} -\sqrt{\frac{k_2+k_3}{4k_3}} & \sqrt{\frac{k_2+k_3}{4k_2}} \\ \sqrt{\frac{k_2+k_3}{4k_3}} & \sqrt{\frac{k_2+k_3}{4k_2}} \end{pmatrix},
\end{aligned}
$$

(1165)

since

$$\mathbf{D} = \begin{pmatrix} -\sqrt{\frac{k_3}{k_2+k_3}} & \sqrt{\frac{k_3}{k_2+k_3}} \\ \sqrt{\frac{k_2}{k_2+k_3}} & \sqrt{\frac{k_2}{k_2+k_3}} \end{pmatrix}, \quad \mathbf{D}^{-1} = \begin{pmatrix} -\sqrt{\frac{k_2+k_3}{4k_3}} & \sqrt{\frac{k_2+k_3}{4k_2}} \\ \sqrt{\frac{k_2+k_3}{4k_3}} & \sqrt{\frac{k_2+k_3}{4k_2}} \end{pmatrix}$$

(1166)

and $\mathbf{D}^{-1}\mathbf{D} = 1$.[5]

---

[5] We diagonalize M, as follows:

$$\begin{vmatrix} -(k_2 + \lambda) & k_3 \\ k_2 & -(k_3 + \lambda) \end{vmatrix} = 0,$$

(1167)

giving $(k_2 + \lambda)(k_3 + \lambda) - k_2 k_3 = 0$, or $\lambda(k_2 + k_3) + \lambda^2 = 0$. Therefore $\lambda = 0$ and $\lambda = -(k_2 + k_3)$ are solutions.

$$\begin{pmatrix} -k_2 & k_3 \\ k_2 & -k_3 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} -(k_2 + k_3) & 0 \\ 0 & -(k_2 + k_3) \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix},$$

(1168)

giving $-k_2 v_1 + k_3 v_2 = -(k_2 + k_3)v_1$ and $v_2 = \pm\sqrt{1 - v_1^2}$.

Therefore, $k_3\sqrt{1 - v_1^{\pm 2}} = \sqrt{k_2 k_3} v_1^{\pm}$, or $k_3^2(1 - v_1^{\pm 2}) = k_2 k_3 v_1^{\pm 2}$, giving $v_1^{\pm} = \pm\sqrt{\frac{k_3}{k_2+k_3}}$ and $v_2^{\pm} = \sqrt{\frac{k_2}{k_2+k_3}}$.
We compute the inverse, as follows:

$$\begin{pmatrix} -a & a \\ b & b \end{pmatrix} \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

(1169)

Therefore,

$$
\begin{aligned}
e^{-\mathbf{M}t} &= \begin{pmatrix} e^{k_2 t} & 0 \\ 0 & e^{k_3 t} \end{pmatrix} \begin{pmatrix} -\sqrt{\frac{k_3}{k_2+k_3}} & \sqrt{\frac{k_3}{k_2+k_3}} \\ \sqrt{\frac{k_2}{k_2+k_3}} & \sqrt{\frac{k_2}{k_2+k_3}} \end{pmatrix} \begin{pmatrix} e^{\sqrt{k_2 k_3}t} & 0 \\ 0 & e^{-\sqrt{k_2 k_3}t} \end{pmatrix} \begin{pmatrix} -\sqrt{\frac{k_2+k_3}{4k_3}} & \sqrt{\frac{k_2+k_3}{4k_2}} \\ \sqrt{\frac{k_2+k_3}{4k_3}} & \sqrt{\frac{k_2+k_3}{4k_2}} \end{pmatrix}, \\[2mm]
&= \begin{pmatrix} e^{k_2 t} & 0 \\ 0 & e^{k_3 t} \end{pmatrix} \begin{pmatrix} -\sqrt{\frac{k_3}{k_2+k_3}} & \sqrt{\frac{k_3}{k_2+k_3}} \\ \sqrt{\frac{k_2}{k_2+k_3}} & \sqrt{\frac{k_2}{k_2+k_3}} \end{pmatrix} \begin{pmatrix} -e^{\sqrt{k_2 k_3}t}\sqrt{\frac{k_2+k_3}{4k_3}} & e^{\sqrt{k_2 k_3}t}\sqrt{\frac{k_2+k_3}{4k_2}} \\ e^{-\sqrt{k_2 k_3}t}\sqrt{\frac{k_2+k_3}{4k_3}} & e^{-\sqrt{k_2 k_3}t}\sqrt{\frac{k_2+k_3}{4k_2}} \end{pmatrix}, \\[2mm]
&= \begin{pmatrix} e^{k_2 t} & 0 \\ 0 & e^{k_3 t} \end{pmatrix} \begin{pmatrix} \left(e^{\sqrt{k_2 k_3}t} + e^{-\sqrt{k_2 k_3}t}\right)\frac{1}{2} & \left(-e^{\sqrt{k_2 k_3}t} + e^{-\sqrt{k_2 k_3}t}\right)\frac{1}{2}\sqrt{\frac{k_3}{k_2}} \\ \left(-e^{\sqrt{k_2 k_3}t} + e^{-\sqrt{k_2 k_3}t}\right)\frac{1}{2}\sqrt{\frac{k_2}{k_3}} & \left(e^{-\sqrt{k_2 k_3}t} + e^{\sqrt{k_2 k_3}t}\right)\frac{1}{2} \end{pmatrix} \\[2mm]
&= \begin{pmatrix} \left(e^{\sqrt{k_2 k_3}t} + e^{-\sqrt{k_2 k_3}t}\right)\frac{e^{k_2 t}}{2} & \left(-e^{\sqrt{k_2 k_3}t} + e^{-\sqrt{k_2 k_3}t}\right)\frac{e^{k_2 t}}{2}\sqrt{\frac{k_3}{k_2}} \\ \left(-e^{\sqrt{k_2 k_3}t} + e^{-\sqrt{k_2 k_3}t}\right)\frac{e^{k_3 t}}{2}\sqrt{\frac{k_2}{k_3}} & \left(e^{-\sqrt{k_2 k_3}t} + e^{\sqrt{k_2 k_3}t}\right)\frac{e^{k_3 t}}{2} \end{pmatrix} \\[2mm]
&= \begin{pmatrix} \cosh(\sqrt{k_2 k_3}t)e^{k_2 t} & -\sinh(\sqrt{k_2 k_3}t)e^{k_2 t}\sqrt{\frac{k_3}{k_2}} \\ -\sinh(\sqrt{k_2 k_3}t)e^{k_3 t}\sqrt{\frac{k_2}{k_3}} & \cosh(\sqrt{k_2 k_3}t)e^{k_3 t} \end{pmatrix}
\end{aligned}
\tag{1172}
$$

giving

$$
\begin{aligned}
\mathbf{v}(t) = & \begin{pmatrix} \cosh(\sqrt{k_2 k_3}t)e^{-k_2 t} & \sinh(\sqrt{k_2 k_3}t)e^{-k_2 t}\sqrt{\frac{k_3}{k_2}} \\ \sinh(\sqrt{k_2 k_3}t)e^{-k_3 t}\sqrt{\frac{k_2}{k_3}} & \cosh(\sqrt{k_2 k_3}t)e^{-k_3 t} \end{pmatrix} \\[2mm]
& \times \begin{pmatrix} k_1 E(0)\int_0^t dt'\, \cosh(\sqrt{k_2 k_3}t')e^{-(k_1-k_2)t'} \\ -\sqrt{\frac{k_2}{k_3}}k_1 E(0)\int_0^t dt'\, \sinh(\sqrt{k_2 k_3}t')e^{-(k_1-k_3)t'} \end{pmatrix}
\end{aligned}
\tag{1173}
$$

giving $-m_{11}+m_{21} = a^{-1}$, $-m_{12}+m_{22} = 0$, $m_{11}+m_{21} = 0$, $m_{12}+m_{22} = b^{-1}$. Therefore, $-m_{11} = m_{21} = a^{-1}/2$ and $m_{12} = m_{22} = b^{-1}/2$, and the inverse is

$$
\begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix} = \begin{pmatrix} -a^{-1}/2 & b^{-1}/2 \\ a^{-1}/2 & b^{-1}/2 \end{pmatrix}.
\tag{1170}
$$

Finally, we note that

$$
\begin{aligned}
& \begin{pmatrix} -\sqrt{\frac{k_3}{k_2+k_3}} & \sqrt{\frac{k_3}{k_2+k_3}} \\ \sqrt{\frac{k_2}{k_2+k_3}} & \sqrt{\frac{k_2}{k_2+k_3}} \end{pmatrix} \begin{pmatrix} -\sqrt{k_2 k_3} & 0 \\ 0 & \sqrt{k_2 k_3} \end{pmatrix} \begin{pmatrix} -\sqrt{\frac{k_2+k_3}{4k_3}} & \sqrt{\frac{k_2+k_3}{4k_2}} \\ \sqrt{\frac{k_2+k_3}{4k_3}} & \sqrt{\frac{k_2+k_3}{4k_2}} \end{pmatrix} = \\[2mm]
& \begin{pmatrix} -\sqrt{\frac{k_3}{k_2+k_3}} & \sqrt{\frac{k_3}{k_2+k_3}} \\ \sqrt{\frac{k_2}{k_2+k_3}} & \sqrt{\frac{k_2}{k_2+k_3}} \end{pmatrix} \begin{pmatrix} \sqrt{k_2 k_3}\sqrt{\frac{k_2+k_3}{4k_3}} & -\sqrt{k_2 k_3}\sqrt{\frac{k_2+k_3}{4k_2}} \\ \sqrt{k_2 k_3}\sqrt{\frac{k_2+k_3}{4k_3}} & \sqrt{k_2 k_3}\sqrt{\frac{k_2+k_3}{4k_2}} \end{pmatrix} = \begin{pmatrix} 0 & k_3 \\ k_2 & 0 \end{pmatrix}.
\end{aligned}
\tag{1171}
$$

Integrating Eq. (1173), we obtain:

$$
\mathbf{v}(t) = \begin{pmatrix} \cosh(\sqrt{k_2 k_3}\,t)e^{-k_2 t} & \sinh(\sqrt{k_2 k_3}\,t)e^{-k_2 t}\sqrt{\frac{k_3}{k_2}} \\ \sinh(\sqrt{k_2 k_3}\,t)e^{-k_3 t}\sqrt{\frac{k_2}{k_3}} & \cosh(\sqrt{k_2 k_3}\,t)e^{-k_3 t} \end{pmatrix}
$$
$$
\times \begin{pmatrix} k_1 E(0)\left(\frac{e^{t[\sqrt{k_2 k_3}-(k_1-k_2)]}-1}{[\sqrt{k_2 k_3}-(k_1-k_2)]} + \frac{e^{t[-\sqrt{k_2 k_3}-(k_1-k_2)]}-1}{[-\sqrt{k_2 k_3}-(k_1-k_2)]} \right) \\ -\sqrt{\frac{k_2}{k_3}}k_1 E(0)\left(\frac{e^{t[\sqrt{k_2 k_3}-(k_1-k_3)]}-1}{[\sqrt{k_2 k_3}-(k_1-k_3)]} - \frac{e^{t[-\sqrt{k_2 k_3}-(k_1-k_3)]}-1}{[-\sqrt{k_2 k_3}-(k_1-k_3)]} \right) \end{pmatrix}. \tag{1174}
$$

Therefore,

$$
B(t) = B(0) + k_1 E(0)\cosh(\sqrt{k_2 k_3}\,t)e^{-k_2 t}\left(\frac{e^{t[\sqrt{k_2 k_3}-(k_1-k_2)]}-1}{[\sqrt{k_2 k_3}-(k_1-k_2)]} + \frac{e^{t[-\sqrt{k_2 k_3}-(k_1-k_2)]}-1}{[-\sqrt{k_2 k_3}-(k_1-k_2)]} \right)
$$
$$
- \sinh(\sqrt{k_2 k_3}\,t)e^{-k_2 t}k_1 E(0)\left(\frac{e^{t[\sqrt{k_2 k_3}-(k_1-k_3)]}-1}{[\sqrt{k_2 k_3}-(k_1-k_3)]} - \frac{e^{t[-\sqrt{k_2 k_3}-(k_1-k_3)]}-1}{[-\sqrt{k_2 k_3}-(k_1-k_3)]} \right)
$$
$$
\tag{1175}
$$

59   Kinetics of Isomerization: Temperature Effect

We consider the rate of 11-cis/all-trans isomerization of retinal:

$$\kappa(\beta) = Ae^{-\beta\Delta G}, \tag{1176}$$

giving

$$ln[\kappa(\beta)] = ln[A] - \beta\Delta G, \tag{1177}$$

Expanding $f(\beta) = \beta\Delta(\beta)$ around β_0, with $\Delta G_0 = \Delta G(\beta_0)$, we obtain:

$$\beta\Delta G = \beta_0\Delta G_0 + (\beta - \beta_0)[\Delta G_0 + \beta_0\Delta G_0'] + \frac{1}{2}(\beta - \beta_0)^2[2\Delta G_0' + \beta_0\Delta G_0''] + ..., \tag{1178}$$

Therefore,

$$ln[\kappa(\beta)] = ln[A] - \beta_0\Delta G_0 - (\beta - \beta_0)[\Delta G_0 + \beta_0\Delta G_0'] - \frac{1}{2}(\beta - \beta_0)^2[2\Delta G_0' + \beta_0\Delta G_0''] + ... \tag{1179}$$

In the particular case when $ln[\kappa(\beta)]$ is linear, as a function of β, we can neglect the second order term and obtain,

$$ln[\kappa(\beta)] = \alpha - (\beta - \beta_0)\gamma, \tag{1180}$$

where the slope γ is defined, as follows:

$$\begin{aligned}
\gamma &= [\Delta G_0 + \beta_0\Delta G_0'], \\
&= \Delta H_0 - T_0\Delta S_0 + \beta_0[\Delta H_0' - T_0\Delta S_0' + k_B^{-1}\beta_0^{-2}\Delta S_0], \\
&= \Delta H_0 + \beta_0\Delta H_0' - k_B^{-1}\Delta S_0', \\
&= \Delta H_0 + k_B^{-1}[Cp\beta_0^{-1} - \Delta S_0'],
\end{aligned} \tag{1181}$$

and intercept

$$\begin{aligned}
\alpha &= ln[A] - \beta_0\Delta G_0, \\
&= ln[A] - \beta_0\Delta H_0 + k_B^{-1}\Delta S_0.
\end{aligned} \tag{1182}$$

Note that $\gamma > \Delta H_0$ since $\Delta S_0' < 0$ and $Cp > 0$.

60 Quantum Heat Engine

The goal of this section is to discuss how to simulate the dynamics of population transfer in a 3-level where levels 1 and 3 are coupled by a hot bath (e.g., a bath of solar photons at 6000 K), while levels 1 and 2 are coupled by a cold bath (i.e., a phonon bath at room-temperature).

We model the system in interaction with both baths, according to the following Hamiltonian: $H = H_s + H_h + H_c + V_{sh} + V_{sc}$, where

$$H_s = \sum_{j=1}^{3} E_j |j\rangle\langle j|$$

is the Hamiltonian of the system, and

$$H_h = \left[\sum_{k=1}^{N} \frac{P_k^2}{2M_k} + \frac{1}{2} M_k \omega_k^2 X_k^2 \right] \sum_{j=1}^{3} |j\rangle\langle j|,$$

and

$$H_c = \left[\sum_{k=1}^{N} \frac{p_k^2}{2m_k} + \frac{1}{2} m_k \omega_k^2 x_k^2 \right] \sum_{j=1}^{3} |j\rangle\langle j|,$$

are the Hamiltonians of the hot and cold baths, respectively. In addition, V_{sh} and V_{sc} introduce the electronic couplings, as follows:

$$V_{sh} = \hbar \sum_{k=1}^{N} g_k^{13} X_k |1\rangle\langle 3| + c.c.$$

$$V_{sc} = \hbar \sum_{k=1}^{N} g_k^{12} x_k |1\rangle\langle 2| + c.c.$$

$$(1183)$$

Given an initial state of the quantum system, defined as follows: $\psi(0) = c_1(0)|1\rangle + c_2(0)|2\rangle + c_3(0)|3\rangle$, we explore its time-evolution as it approaches the steady state of constant populations. We propagate the quantum system by using the split-operator Fourier transform (SOFT) method and the coordinates and momenta of the baths by using the Velocity-Verlet algorithm according to the time-dependent self-consistent field (TD-SCF) method. Each propagation time step τ is computed, as follows:

$$\psi(t+\tau) = e^{-\frac{i}{\hbar}[H_s + V_{sh} + V_{sc}]\tau} \psi(t),$$

$$x_k(t+\tau) = x_k(t) + \frac{p_k}{m_k}\tau + \frac{1}{2}\frac{f_k(t)}{m_k}\tau^2,$$

$$X_k(t+\tau) = X_k(t) + \frac{P_k}{M_k}\tau + \frac{1}{2}\frac{F_k(t)}{M_k}\tau^2,$$

$$p_k(t+\tau) = p_k(t) + \frac{1}{2m_k}\left[f_k(t) + f_k(t+\tau)\right]\tau,$$

$$P_k(t+\tau) = P_k(t) + \frac{1}{2M_k}\left[F_k(t) + F_k(t+\tau)\right]\tau,$$

$$(1184)$$

with

$$f_k(t) = -\frac{\partial}{\partial x_k}\langle\psi(t)|H|\psi(t)\rangle = -m_k\omega_k^2 x_k(t) - 2\hbar Re[c_1(t)^* g_k^{12} c_2(t)],$$

$$F_k(t) = -\frac{\partial}{\partial X_k}\langle\psi(t)|H|\psi(t)\rangle = -m_k\omega_k^2 X_k(t) - 2\hbar Re[c_1(t)^* g_k^{13} c_3(t)].$$

(1185)

The initial conditions for the coordinates and momenta of both baths are sampled according to the Boltzmann distributions,

$$g(p, x) = z^{-1} e^{-\beta_c \sum_{k=1}^{N}\left[\frac{p_k^2}{2m_k} + \frac{1}{2}m_k\omega_k^2 x_k^2\right]},$$

$$G(P, X) = Z^{-1} e^{-\beta_h \sum_{k=1}^{N}\left[\frac{P_k^2}{2M_k} + \frac{1}{2}M_k\omega_k^2 X_k^2\right]}.$$

(1186)

Having propagated the system for a sufficiently long time t_f, we analyze the time-dependent populations $\zeta_j(t) = |c_j(t)|^2$. We observe that $\zeta_j(t_f)$ approaches the Boltzmann distribution $\zeta_j(\beta_h) = \frac{e^{-\beta_h E_j}}{\sum_{j=1}^{3} e^{-\beta_h E_j}}$, when $\beta_c \to \beta_h$. However, when $\beta_c > \beta_h$, then $\zeta_2(t_f) < \zeta_2(\beta_h)$ while $\zeta_1(t_f) > \zeta_1(\beta_h)$ and $\zeta_3(t_f) > \zeta_3(\beta_h)$.

61   Solutions of Computational Assignments

61.1   Problem 1

Computational Problem 1: Write a computer program to represent the wave-packet, introduced by Eq. (11) on a grid of equally spaced coordinates $x_j = x_{min} + (j - 1)\Delta$ with finite resolution $\Delta = (x_{max} - x_{min})/(n - 1)$ and visualize the output. Choose $x_0 = 0$ and $p_0 = 0$, in the range x=(-20,20), with $\alpha = \omega m$, where $m = 1$ and $\omega = 1$.

   To visualize the output of this program, cut the source code attached below save it in a file named Problem1.f, compile it by typing

```
 gfortran Problem1.f -o Problem1
```

run it by typing

```
./Problem1
```

Visualize the output as follows: type

```
gnuplot
```

then type

```
plot ``arch.0000''
```

That will show the representation of the Gaussian state, introduced in Eq. (6) in terms of an array of numbers associated with a grid in coordinate space. To exit, type

```
quit
```

```
      PROGRAM Problem_1
      call Initialize()
      CALL SAVEWF(0)
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Initialize()
c
c     Wave Packet Initialization: Gaussian centered at xk, with momentum pk
c
      IMPLICIT NONE
      INTEGER nptx,npts,kk
      COMPLEX chi,EYE
      REAL omega,xmin,xmax,dx,pi,mass,xk,pk,x,alpha
      PARAMETER(npts=10,nptx=2**npts)
      COMMON / wfunc/ chi(nptx)
      common /xy/  xmin,xmax
      common /packet/mass,xk,pk
      xmin=-20.
      xmax=20.
      EYE=(0.0,1.0)
      pi= acos(-1.0)
      omega=1.
      dx=(xmax-xmin)/real(nptx)
      pk=0.0
      xk=0.0
      mass=1.0
      alpha=mass*omega
      do kk=1,nptx
         x=xmin+kk*dx
         chi(kk)=((alpha/pi)**0.25)
    1        *exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
      end do
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE SAVEWF(j)
c
c     Save Wave-packet in coordinate space
c
      IMPLICIT NONE
      INTEGER nptx,npts,kk,j
      COMPLEX chi,EYE
      REAL RV,omega,xmin,xmax,dx,pi,mass,xk,pk,x,alpha,Vpot,RKE
      character*9 B
      PARAMETER(npts=10,nptx=2**npts)
      COMMON / wfunc/ chi(nptx)
      common /xy/  xmin,xmax
      common /packet/mass,xk,pk
      write(B, '(A,i4.4)') 'arch.', j
```

```fortran
      OPEN(1,FILE=B)
      dx=(xmax-xmin)/real(nptx)
      do kk=1,nptx
         x=xmin+kk*dx
         WRITE(1,22) x,chi(kk)
      end do
      CLOSE(1)
 22   FORMAT(6(e13.6,2x))
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

61.2   Problem 2

Computational Problem 2: Write a computer program to represent the initial state, introduced by Eq. (11), in the momentum space by applying the FFT algorithm to the grid-based representation generated in Problem 1 and visualize the output. Represent the wave-packet amplitudes and phases in the range p=(-4,4) and compare your output with the corresponding values obtained from the analytic Fourier transform obtained by using:

$$\int dx \, \exp(-a_2 x^2 + a_1 x + a_0) = \sqrt{\pi/a_2} \, \exp(a_0 + a_1^2/(4a_2)).$$

In order to visualize the output of this program, cut the source code attached below save it in a file named Problem2.f, compile it by typing

```
gfortran Problem2.f -o Problem2
```

run it by typing

```
./Problem2
```

Visualize the output as follows: type

```
gnuplot
```

then type

```
plot ''nume.0000''
```

That will show the representation of the amplitude of the Fourier transform of the Gaussian state, introduced in Eq. (6), in terms of an array of numbers associated with a grid in momentum space. In order to visualize the analytic results on top of the numerical values type

```
replot ''anal.0000''
```

In order to visualize the numerically computed phases as a function of p type

```
plot ''nume.0000 u 1:3''
```

and to visualize the analytic results on top of the numerical values type

```
replot ''anal.0000''
```

To exit, type

```
quit
```

```fortran
      PROGRAM Problem2
      call Initialize()
      CALL SAVEFT()
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Initialize()
c
c     Wave Packet Initialization: Gaussian centered at xk, with momentum pk
c
      IMPLICIT NONE
      INTEGER nptx,npts,kk
      COMPLEX chi,EYE
      REAL omega,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha
      PARAMETER(npts=10,nptx=2**npts)
      COMMON / wfunc/ chi(nptx)
      common /xy/  xmin,xmax
      common /packet/rmass,xk,pk
      xmin=-20.
      xmax=20.
      EYE=(0.0,1.0)
      pi= acos(-1.0)
      omega=1.
      dx=(xmax-xmin)/real(nptx)
      pk=0.0
      xk=5.0
      rmass=1.0
      alpha=rmass*omega
      do kk=1,nptx
         x=xmin+kk*dx
         chi(kk)=((alpha/pi)**0.25)
    1        *exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
      end do
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SAVEFT()
c
c     Save wave-packet in momentum space
c
      IMPLICIT NONE
      INTEGER nptx,kx,nx,npts,j
      REAL theta,wm,p,xmin,xmax,rmass,xk,pi,alenx,pk,rm,re,ri
      COMPLEX eye,chi,Psip
      character*9 B1,B2
      parameter(npts=10,nptx=2**npts)
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
      COMMON / wfunc/ chi(nptx)
      j=0
```

```fortran
      write(B1, '(A,i4.4)') 'anal.', j
      OPEN(1,FILE=B1)
      write(B2, '(A,i4.4)') 'nume.', j
      OPEN(2,FILE=B2)
      CALL fourn(chi,nptx,1,-1)
      pi = acos(-1.0)
      alenx=xmax-xmin
      do kx=1,nptx
         if(kx.le.(nptx/2+1)) then
            nx=kx-1
         else
            nx=kx-1-nptx
         end if
         p=0.
         if(nx.ne.0) p = real(nx)*2.*pi/alenx
c     Numerical Solution
         chi(kx)=chi(kx)*alenx/sqrt(2.0*pi)/nptx
         re=chi(kx)
         ri=imag(chi(kx))
         IF(re.NE.0) theta=atan(ri/re)
         rm=abs(chi(kx))
         IF(abs(p).LE.(4.)) WRITE(2,22) p,rm,theta
         IF(nx.EQ.(nptx/2)) WRITE(2,22)
c     Analytic Solution
         CALL FT_analy(Psip,p)
         re=Psip
         ri=imag(Psip)
         IF(re.NE.0) theta=atan(ri/re)
         rm=abs(Psip)
         IF(abs(p).LE.(4.)) WRITE(1,22) p,rm,theta
         IF(nx.EQ.(nptx/2)) WRITE(1,22)
      end do
      CALL fourn(chi,nptx,1,1)
 22   FORMAT(6(e13.6,2x))
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine FT_analy(Psip,p)
c
c     Analytic Fourier Transform of the initial Gaussian wave-packet
c
      IMPLICIT NONE
      REAL p,pi,alpha,rmass,xk,pk,omega
      COMPLEX Psip,c0,c1,c2,eye
      common /packet/  rmass,xk,pk
      eye=(0.0,1.0)
      omega=1.
      alpha = rmass*omega
      pi=acos(-1.0)
      c2=alpha/2.
      c1=alpha*xk+eye*(pk-p)
```

```
      c0=-alpha/2.*xk**2-eye*pk*xk
      Psip=sqrt(pi/c2)/sqrt(2.0*pi)*(alpha/pi)**0.25
     1     *exp(c1**2/(4.0*c2))*exp(c0)
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c      Subroutines from Numerical Recipes
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE FOURN(DATA,NN,NDIM,ISIGN)
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION NN(NDIM),DATA(*)
      NTOT=1
      DO 11 IDIM=1,NDIM
         NTOT=NTOT*NN(IDIM)
 11   CONTINUE
      NPREV=1
      DO 18 IDIM=1,NDIM
         N=NN(IDIM)
         NREM=NTOT/(N*NPREV)
         IP1=2*NPREV
         IP2=IP1*N
         IP3=IP2*NREM
         I2REV=1
         DO 14 I2=1,IP2,IP1
            IF(I2.LT.I2REV)THEN
               DO 13 I1=I2,I2+IP1-2,2
                  DO 12 I3=I1,IP3,IP2
                     I3REV=I2REV+I3-I2
                     TEMPR=DATA(I3)
                     TEMPI=DATA(I3+1)
                     DATA(I3)=DATA(I3REV)
                     DATA(I3+1)=DATA(I3REV+1)
                     DATA(I3REV)=TEMPR
                     DATA(I3REV+1)=TEMPI
 12               CONTINUE
 13            CONTINUE
            ENDIF
            IBIT=IP2/2
 1          IF ((IBIT.GE.IP1).AND.(I2REV.GT.IBIT)) THEN
               I2REV=I2REV-IBIT
               IBIT=IBIT/2
               GO TO 1
            ENDIF
            I2REV=I2REV+IBIT
 14      CONTINUE
         IFP1=IP1
 2       IF(IFP1.LT.IP2)THEN
            IFP2=2*IFP1
            THETA=ISIGN*6.28318530717959D0/(IFP2/IP1)
            WPR=-2.D0*DSIN(0.5D0*THETA)**2
            WPI=DSIN(THETA)
```

268

```
            WR=1.D0
            WI=0.D0
            DO 17 I3=1,IFP1,IP1
                DO 16 I1=I3,I3+IP1-2,2
                    DO 15 I2=I1,IP3,IFP2
                        K1=I2
                        K2=K1+IFP1
                        TEMPR=SNGL(WR)*DATA(K2)-SNGL(WI)*DATA(K2+1)
                        TEMPI=SNGL(WR)*DATA(K2+1)+SNGL(WI)*DATA(K2)
                        DATA(K2)=DATA(K1)-TEMPR
                        DATA(K2+1)=DATA(K1+1)-TEMPI
                        DATA(K1)=DATA(K1)+TEMPR
                        DATA(K1+1)=DATA(K1+1)+TEMPI
15                  CONTINUE
16              CONTINUE
                WTEMP=WR
                WR=WR*WPR-WI*WPI+WR
                WI=WI*WPR+WTEMP*WPI+WI
17          CONTINUE
            IFP1=IFP2
            GO TO 2
        ENDIF
        NPREV=N*NPREV
18   CONTINUE
     RETURN
     END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

61.3 Problem 3

Computational Problem 3: Write a computer program to compute the expectation values of the position $x(0) = \langle \Psi_0 | \hat{x} | \Psi_0 \rangle$ and the potential energy $V = \langle \Psi_0 | V(\hat{x}) | \Psi_0 \rangle$, where $V(x)$ is defined according to Eq. (17) for the initial wave-packet, introduced by Eq. (11), with various possible values of x_0 and p_0, with $\alpha = \omega m$, where $m = 1$ and $\omega = 1$.

In order to visualize the output of this program, cut the source code attached below save it in a file named Problem3.f, compile it by typing

```
gfortran Problem3.f -o Problem3
```

run it by typing

```
./Problem3
```

The printout on the screen includes the numerically expectation values $\langle \Psi_t | \hat{V} | \Psi_t \rangle$ and $\langle \Psi_t | \hat{x} | \Psi_t \rangle$.

```fortran
      PROGRAM Problem3
      IMPLICIT NONE
      REAL x,VENERGY
      CALL Initialize()
      CALL PE(VENERGY)
      CALL Px(x)
      PRINT *, "<Psi|V|Psi>=",VENERGY
      PRINT *, "<Psi|x|Psi>=",x
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Initialize()
c
c     Wave Packet Initialization: Gaussian centered at xk, with momentum pk
c
      IMPLICIT NONE
      INTEGER nptx,npts,kk
      COMPLEX chi,EYE
      REAL omega,xmin,xmax,dx,pi,mass,xk,pk,x,alpha
      PARAMETER(npts=10,nptx=2**npts)
      COMMON / wfunc/ chi(nptx)
      common /xy/   xmin,xmax
      common /packet/mass,xk,pk
      xmin=-20.
      xmax=20.
      EYE=(0.0,1.0)
      pi= acos(-1.0)
      omega=1.
      dx=(xmax-xmin)/real(nptx)
      pk=0.0
      xk=0.0
      mass=1.0
      alpha=mass*omega
      do kk=1,nptx
         x=xmin+kk*dx
         chi(kk)=((alpha/pi)**0.25)
     1         *exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
      end do
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PE(RV)
c
c     Expectation Value of the Potential Enegy
c
      IMPLICIT NONE
      INTEGER nptx,npts,k
      COMPLEX chi
      REAL Vpot,RV,xmin,xmax,dx,x
      PARAMETER(npts=10,nptx=2**npts)
```

271

```fortran
      COMMON / wfunc/ chi(nptx)
      common /xy/  xmin,xmax
      dx=(xmax-xmin)/real(nptx)
      RV=0.0
      do k=1,nptx
         x=xmin+k*dx
         CALL VA(Vpot,x)
         RV=RV+chi(k)*Vpot*conjg(chi(k))*dx
      end do
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Px(RV)
c
c     Expectation Value of the position
c
      IMPLICIT NONE
      INTEGER nptx,npts,k
      COMPLEX chi
      REAL RV,xmin,xmax,dx,x
      PARAMETER(npts=10,nptx=2**npts)
      COMMON / wfunc/ chi(nptx)
      common /xy/  xmin,xmax
      dx=(xmax-xmin)/real(nptx)
      RV=0.0
      do k=1,nptx
         x=xmin+k*dx
         RV=RV+chi(k)*x*conjg(chi(k))*dx
      end do
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE VA(V,x)
c
c     Potential Energy Surface: Harmonic Oscillator
c
      IMPLICIT NONE
      REAL V,x,mass,xk,pk,rk,omega
      common /packet/ mass,xk,pk
      omega=1.0
      rk=mass*omega**2
      V=0.5*rk*x*x
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

61.4 Problem 4

Computational Problem 4: Write a computer program to compute the expectation values of the initial momentum $p(0) = \langle \Psi_0 | \hat{p} | \Psi_0 \rangle$ and the kinetic energy $T = \langle \Psi_0 | \hat{p}^2/(2m) | \Psi_0 \rangle$ by using the Fourier transform procedure, where Ψ_0 is the initial wave-packet introduced by Eq. (11), with $x_0 = 0$, $p_0 = 0$, and $\alpha = \omega m$, where $m = 1$ and $\omega = 1$. Compute the expectation value of the energy $E = \langle \Psi_0 | \hat{H} | \Psi_0 \rangle$, where $\hat{H} = \hat{p}^2/(2m) + V(\hat{x})$, with $V(x)$ defined according to Eq. (17) and compare your result with the zero-point energy $E_0 = \omega/2$.

In order to visualize the output of this program, cut the source code attached below save it in a file named Problem4.f, compile it by typing

```
gfortran Problem4.f -o Problem4
```

run it by typing

```
./Problem4
```

The printout on the screen includes the numerically expectation values $\langle \Psi_t | \hat{p} | \Psi_t \rangle$, $\langle \Psi_t | \hat{T} | \Psi_t \rangle$ and $\langle \Psi_t | \hat{H} | \Psi_t \rangle$. Note that the analytic value of $\langle \Psi_t | \hat{T} | \Psi_t \rangle$ is $\hbar\omega/2 = 0.5$ in agreement with the numerical solution.

```fortran
      PROGRAM Problem4
      CALL Initialize()
      CALL Pp(p)
      PRINT *, "<Psi|p|Psi>=",p
      CALL KE(RKE)
      PRINT *, "<Psi|T|Psi>=",RKE
      CALL PE(RV)
      PRINT *, "<Psi|H|Psi>=",RKE+RV
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Initialize()
c
c     Wave Packet Initialization: Gaussian centered at xk, with momentum pk
c
      IMPLICIT NONE
      INTEGER nptx,npts,kk
      COMPLEX chi,EYE
      REAL omega,xmin,xmax,dx,pi,mass,xk,pk,x,alpha
      PARAMETER(npts=10,nptx=2**npts)
      COMMON / wfunc/ chi(nptx)
      common /xy/  xmin,xmax
      common /packet/mass,xk,pk
      xmin=-20.
      xmax=20.
      EYE=(0.0,1.0)
      pi= acos(-1.0)
      omega=1.
      dx=(xmax-xmin)/real(nptx)
      pk=0.0
      xk=0.0
      mass=1.0
      alpha=mass*omega
      do kk=1,nptx
         x=xmin+kk*dx
         chi(kk)=((alpha/pi)**0.25)
     1        *exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
      end do
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PE(RV)
c
c     Expectation Value of the Potential Enegy
c
      IMPLICIT NONE
      INTEGER nptx,npts,k
      COMPLEX chi
      REAL Vpot,RV,xmin,xmax,dx,x
      PARAMETER(npts=10,nptx=2**npts)
```

274

```fortran
      COMMON / wfunc/ chi(nptx)
      common /xy/   xmin,xmax
      dx=(xmax-xmin)/real(nptx)
      RV=0.0
      do k=1,nptx
         x=xmin+k*dx
         CALL VA(Vpot,x)
         RV=RV+chi(k)*Vpot*conjg(chi(k))*dx
      end do
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE KE(RKE)
c
c     Expectation value of the kinetic energy
c
      IMPLICIT NONE
      INTEGER kk,nptx,kx,nx,npts
      REAL dp,RKE,p,xmin,xmax,pi,alenx,dx,mass,xk,pk
      COMPLEX eye,chi,Psip,chic
      parameter(npts=10,nptx=2**npts)
      DIMENSION chic(nptx)
      common /xy/   xmin,xmax
      common /packet/mass,xk,pk
      COMMON / wfunc/ chi(nptx)
      RKE=0.0
      pi = acos(-1.0)
      dx=(xmax-xmin)/nptx
      dp=2.*pi/(xmax-xmin)
      do kk=1,nptx
         chic(kk)=chi(kk)
      end do
      CALL fourn(chic,nptx,1,1)
      do kx=1,nptx
         if(kx.le.(nptx/2+1)) then
            nx=kx-1
         else
            nx=kx-1-nptx
         end if
         p=0.
         if(nx.ne.0) p = real(nx)*dp
         chic(kx)=p**2/(2.0*mass)*chic(kx)/nptx
      end do
      CALL fourn(chic,nptx,1,-1)
      do kk=1,nptx
         RKE=RKE+conjg(chi(kk))*chic(kk)*dx
      end do
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Pp(pe)
```

```fortran
c
c      Expectation value of the momentum
c
       IMPLICIT NONE
       INTEGER kk,nptx,kx,nx,npts
       REAL dp,pe,p,xmin,xmax,pi,alenx,dx,mass,xk,pk
       COMPLEX eye,chi,Psip,chic
       parameter(npts=10,nptx=2**npts)
       DIMENSION chic(nptx)
       common /xy/  xmin,xmax
       common /packet/mass,xk,pk
       COMMON / wfunc/ chi(nptx)
       pe=0.0
       pi = acos(-1.0)
       dx=(xmax-xmin)/nptx
       dp=2.*pi/(xmax-xmin)
       do kk=1,nptx
          chic(kk)=chi(kk)
       end do
       CALL fourn(chic,nptx,1,1)
       do kx=1,nptx
          if(kx.le.(nptx/2+1)) then
             nx=kx-1
          else
             nx=kx-1-nptx
          end if
          p=0.
          if(nx.ne.0) p = real(nx)*dp
          chic(kx)=p*chic(kx)/nptx
       end do
       CALL fourn(chic,nptx,1,-1)
       do kk=1,nptx
          pe=pe+conjg(chi(kk))*chic(kk)*dx
       end do
       return
       end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       SUBROUTINE VA(V,x)
c
c      Potential Energy Surface: Harmonic Oscillator
c
       implicit none
       REAL V,x,mass,xk,pk,rk,omega
       common /packet/ mass,xk,pk
       omega=1.0
       rk=mass*omega**2
       V=0.5*rk*x*x
       RETURN
       END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c      Subroutines from Numerical Recipes
```

```fortran
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE FOURN(DATA,NN,NDIM,ISIGN)
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION NN(NDIM),DATA(*)
      NTOT=1
      DO 11 IDIM=1,NDIM
        NTOT=NTOT*NN(IDIM)
 11   CONTINUE
      NPREV=1
      DO 18 IDIM=1,NDIM
        N=NN(IDIM)
        NREM=NTOT/(N*NPREV)
        IP1=2*NPREV
        IP2=IP1*N
        IP3=IP2*NREM
        I2REV=1
        DO 14 I2=1,IP2,IP1
          IF(I2.LT.I2REV)THEN
            DO 13 I1=I2,I2+IP1-2,2
              DO 12 I3=I1,IP3,IP2
                I3REV=I2REV+I3-I2
                TEMPR=DATA(I3)
                TEMPI=DATA(I3+1)
                DATA(I3)=DATA(I3REV)
                DATA(I3+1)=DATA(I3REV+1)
                DATA(I3REV)=TEMPR
                DATA(I3REV+1)=TEMPI
 12           CONTINUE
 13         CONTINUE
          ENDIF
          IBIT=IP2/2
 1        IF ((IBIT.GE.IP1).AND.(I2REV.GT.IBIT)) THEN
            I2REV=I2REV-IBIT
            IBIT=IBIT/2
            GO TO 1
          ENDIF
          I2REV=I2REV+IBIT
 14     CONTINUE
        IFP1=IP1
 2      IF(IFP1.LT.IP2)THEN
          IFP2=2*IFP1
          THETA=ISIGN*6.28318530717959D0/(IFP2/IP1)
          WPR=-2.D0*DSIN(0.5D0*THETA)**2
          WPI=DSIN(THETA)
          WR=1.D0
          WI=0.D0
          DO 17 I3=1,IFP1,IP1
            DO 16 I1=I3,I3+IP1-2,2
              DO 15 I2=I1,IP3,IFP2
                K1=I2
                K2=K1+IFP1
```

```fortran
                    TEMPR=SNGL(WR)*DATA(K2)-SNGL(WI)*DATA(K2+1)
                    TEMPI=SNGL(WR)*DATA(K2+1)+SNGL(WI)*DATA(K2)
                    DATA(K2)=DATA(K1)-TEMPR
                    DATA(K2+1)=DATA(K1+1)-TEMPI
                    DATA(K1)=DATA(K1)+TEMPR
                    DATA(K1+1)=DATA(K1+1)+TEMPI
15              CONTINUE
16            CONTINUE
              WTEMP=WR
              WR=WR*WPR-WI*WPI+WR
              WI=WI*WPR+WTEMP*WPI+WI
17          CONTINUE
            IFP1=IFP2
            GO TO 2
          ENDIF
          NPREV=N*NPREV
18    CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

61.5 Problem 5

Computational Problem 5: Expand the exponential operators in both sides of Eq. (26) and show that the Trotter expansion is accurate to second order in powers of τ.

Expanding the left-hand-side (l.h.s.) of Eq. (18) from the lecture notes gives:

$$e^{-i\hat{H}\tau} = 1 - i\hat{H}\tau - \frac{1}{2}\hat{H}^2\tau^2 + O(\tau^3), \tag{1187}$$

where $\hat{H} = \hat{p}^2/(2m) + \hat{V}$. Therefore,

$$e^{-i\hat{H}\tau} = 1 - i\hat{H}\tau - \frac{1}{2}\frac{\hat{p}^4}{4m^2}\tau^2 - \frac{1}{2}\hat{V}^2\tau^2 - \frac{1}{2}\frac{\hat{p}^2}{2m}\hat{V}\tau^2 - \frac{1}{2}\hat{V}\frac{\hat{p}^2}{2m}\tau^2 + O(\tau^3), \tag{1188}$$

In order to show that the Trotter expansion, introduced by Eq. (18), is accurate to second order in τ, we must expand the right-hand-side (r.h.s.) of Eq. (18) and show that suchh an expansion equals the r.h.s. of Eq. (1188).

Expanding the right-hand-side (r.h.s.) of Eq. (18) gives,

$$e^{-iV(\hat{x})\tau/2}e^{-i\hat{p}^2\tau/(2m)}e^{-iV(\hat{x})\tau/2} = \left(1 - i\hat{V}\tau/2 - \frac{1}{2}\hat{V}^2\tau^2/4 + O(\tau^3)\right)\left(1 - i\frac{\hat{p}^2}{2m}\tau - \frac{1}{2}\frac{\hat{p}^4}{4m^2}\tau^2 + O(\tau^3)\right)$$

$$\times \left(1 - i\hat{V}\tau/2 - \frac{1}{2}\hat{V}^2\tau^2/4 + O(\tau^3)\right), \tag{1189}$$

$$e^{-iV(\hat{x})\tau/2}e^{-i\hat{p}^2\tau/(2m)}e^{-iV(\hat{x})\tau/2} = \left(1 - i\hat{V}\tau/2 - \frac{1}{2}\hat{V}^2\tau^2/4 - i\frac{\hat{p}^2}{2m}\tau - \hat{V}\frac{\hat{p}^2}{2m}\tau^2/2 - \frac{1}{2}\frac{\hat{p}^4}{4m^2}\tau^2 + O(\tau^3)\right)$$

$$\times \left(1 - i\hat{V}\tau/2 - \frac{1}{2}\hat{V}^2\tau^2/4 + O(\tau^3)\right), \tag{1190}$$

$$e^{-iV(\hat{x})\tau/2}e^{-i\hat{p}^2\tau/(2m)}e^{-iV(\hat{x})\tau/2} = 1 - i\hat{V}\tau/2 - \frac{1}{2}\hat{V}^2\tau^2/4 - i\frac{\hat{p}^2}{2m}\tau - \hat{V}\frac{\hat{p}^2}{2m}\tau^2/2 - \frac{1}{2}\frac{\hat{p}^4}{4m^2}\tau^2$$

$$- i\hat{V}\tau/2 - \hat{V}^2\tau^2/4 - \frac{\hat{p}^2}{2m}\hat{V}\tau^2/2 - \frac{1}{2}\hat{V}^2\tau^2/4 + O(\tau^3), \tag{1191}$$

$$e^{-iV(\hat{x})\tau/2}e^{-i\hat{p}^2\tau/(2m)}e^{-iV(\hat{x})\tau/2} = 1 - i\hat{V}\tau - i\frac{\hat{p}^2}{2m}\tau - \frac{1}{2}\hat{V}^2\tau^2 - \hat{V}\frac{\hat{p}^2}{2m}\tau^2/2 - \frac{1}{2}\frac{\hat{p}^4}{4m^2}\tau^2$$

$$- \frac{\hat{p}^2}{2m}\hat{V}\tau^2/2 + O(\tau^3). \tag{1192}$$

Note that the r.h.s. of Eq. (1192) is identical to the r.h.s. of E. (1188), completing the proof that the Trotter expansion, introduced by Eq. (18), is accurate to second order in τ.

61.6 Problem 6

Computational Problem 6: Write a computer program that propagates the initial state $\Psi_0(x)$ for a single time increment ($\tau = 0.1$ a.u.). Use $x_0 = -2.5$, $p_0 = 0$, and $\alpha = \omega m$, where $m = 1$ and $\omega = 1$. Implement the SOFT method for the Hamiltonian $\hat{H} = \hat{p}^2/(2m) + V(\hat{x})$, where $V(x)$ is defined according to Eq. (17). Compare the resulting propagated state with the analytic solution obtained by substituting Eq. (24) into Eq. (23).

In order to visualize the output of this program, cut the source code attached below save it in a file named Problem6.f, compile it by typing

```
gfortran Problem6.f -o Problem6
```

run it by typing

```
./Problem6
```

and visualize the output as follows: type

```
gnuplot
```

then type

```
set dat sty line
```

then type

```
set yrange[0:6]
```

and the type

```
plot ''arch.0002''
```

That will show the numerical propagation after one step with $\tau = 0.1$. In order to visualize the analytic result on top of the numerical propagation, type

```
replot ''arch.0002'' u 1:3
```

To exit, type

```
quit
```

```fortran
      PROGRAM Problem6
c
c     1-D wave packet propagation
c
      IMPLICIT NONE
      INTEGER NN,npts,nptx,ndump
      INTEGER istep,nstep
      REAL dt,xc,pc
      COMPLEX vprop,tprop,x_mean,p_mean
      PARAMETER(npts=9,nptx=2**npts,NN=1)
      DIMENSION vprop(nptx,NN,NN),tprop(nptx)
      DIMENSION x_mean(NN),p_mean(NN)
      COMMON /class/ xc,pc
c
      CALL ReadParam(nstep,ndump,dt)
      call Initialize()
      CALL SetKinProp(dt,tprop)
      CALL SetPotProp(dt,vprop)
      DO istep=1,nstep+1
         IF(mod(istep-1,10).EQ.0)
     1        PRINT *, "Step=", istep-1,", Final step=", nstep
         IF(istep.GE.1) CALL PROPAGATE(vprop,tprop)
         IF(mod((istep-1),ndump).EQ.0) THEN
            CALL SAVEWF(istep,ndump,dt)
         END IF
      END DO
 22   FORMAT(6(e13.6,2x))
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine ReadParam(nstep,ndump,dt)
c
c     Parameters defining the grid (xmin, xmax), integration time step (dt),
c     mass (rmass), initial position (xk), initial momentum (pk),
c     number of propagation steps (nstep), and how often to save a pic (ndump)
c
      IMPLICIT NONE
      INTEGER ntype,nstep,nrpt,ireport,ndump,nlit
      REAL xmin,xmax,pk,rmass,xk,dt
      common /packet/  rmass,xk,pk
      common /xy/  xmin,xmax
c
      xmin=-10.0
      xmax= 10.0
      dt=0.1
      rmass=1.0
      xk=-2.5
      pk=1.0
      nstep=1
      ndump=1
```

```fortran
c
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Initialize()

      IMPLICIT NONE
      INTEGER NN,nptx,npts,kk
      COMPLEX chi0,chi,EYE,CRV
      REAL xc,pc,omega,xk2,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha,alpha2
      PARAMETER(npts=9,nptx=2**npts,NN=1)
      DIMENSION CRV(NN,NN)
      common /xy/   xmin,xmax
      common /packet/  rmass,xk,pk
      COMMON / wfunc/ chi(nptx,NN)
      COMMON / iwfunc/ chi0(nptx,NN)
      COMMON /class/ xc,pc

      EYE=(0.0,1.0)
      pi= acos(-1.0)
      omega=1.
      dx=(xmax-xmin)/real(nptx)
      xc=kk
      pc=pk
c
c     Wave Packet Initialization: Gaussian centered at xk, with momentum pk
c
      alpha=rmass*omega
      do kk=1,nptx
         x=xmin+kk*dx
         chi(kk,1)=((alpha/pi)**0.25)
     1        *exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
         chi0(kk,1)=chi(kk,1)
      end do
c
c     Hamiltonian Matrix CRV
c
      do kk=1,nptx
         x=xmin+kk*dx
         CALL HAMIL(CRV,x)
         WRITE(11,22) x,real(CRV(1,1))
      END DO
 22   FORMAT(6(e13.6,2x))
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE HAMIL(CRV,x)
c
c     Hamiltonian Matrix
c
      IMPLICIT NONE
```

282

```fortran
      INTEGER NN
      REAL x,VPOT1
      COMPLEX CRV
      PARAMETER(NN=1)
      DIMENSION CRV(NN,NN)
C
      CALL VA(VPOT1,x)
      CRV(1,1)=VPOT1
C
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE VA(V,x)
C
C     Potential Energy Surface: Harmonic Oscillator
C
      implicit none
      REAL V,x,rmass,xk,pk,rk,omega
      common /packet/ rmass,xk,pk
      omega=1.0
      rk=rmass*omega**2
      V=0.5*rk*x*x
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetKinProp(dt,tprop)
C
C     Kinetic Energy part of the Trotter Expansion: exp(-i p^2 dt/(2 m))
C
      IMPLICIT NONE
      INTEGER nptx,kx,nx,npts
      REAL xsc,xmin,xmax,propfacx,rmass,xk,pi,alenx,dt,pk
      COMPLEX tprop,eye
      parameter(npts=9,nptx=2**npts)
      DIMENSION tprop(nptx)
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
C
      eye=(0.,1.)
      pi = acos(-1.0)
      alenx=xmax-xmin
      propfacx=-dt/2./rmass*(2.*pi)**2
      do kx=1,nptx
         if(kx.le.(nptx/2+1)) then
            nx=kx-1
         else
            nx=kx-1-nptx
         end if
         xsc=0.
         if(nx.ne.0) xsc=real(nx)/alenx
         tprop(kx)=exp(eye*(propfacx*xsc**2))
```

283

```fortran
         end do
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetPotProp(dt,vprop)
c
c     Potential Energy part of the Trotter Expansion: exp(-i V dt/2)
c
      IMPLICIT NONE
      INTEGER NN,ii,nptx,npts
      REAL xmin,xmax,dx,dt,x,VPOT
      COMPLEX vprop,eye
      parameter(npts=9,nptx=2**npts,NN=1)
      DIMENSION vprop(nptx,NN,NN)
      common /xy/  xmin,xmax
      eye=(0.,1.)
      dx=(xmax-xmin)/real(nptx)
c
      do ii=1,nptx
         x=xmin+ii*dx
         CALL VA(VPOT,x)
         vprop(ii,1,1)=exp(-eye*0.5*dt*VPOT)/sqrt(nptx*1.0)
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE energies(energy)
      IMPLICIT NONE
      INTEGER j,NN
      COMPLEX energy,RV,RKE
      PARAMETER (NN=1)
      DIMENSION RV(NN),RKE(NN),energy(NN)
      CALL PE(RV)
      CALL KE(RKE)
      DO j=1,NN
         energy(j)=RV(j)+RKE(j)
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      FUNCTION Psia(x,istep,dt)
c
c     Analytic wave-packet <x|Psia(istep)> obtained by applying the
c     harmonic propagator to the initial state,
c     <x'|Psi(0)> = (alpha/pi)**.25*exp(-alpha/2*(x'-xk)**2+eye*pk*(x'-xk)),
c     where the propagator is
c     <x|exp(-beta H)|x'> = A exp(-rgamma*(x**2+x'**2)+rgammap*x*x'), with
c     A = sqrt(m*omega/(pi*(exp(beta*omega)-exp(-beta*omega)))), beta = i*t,
c     rgamma = 0.5*m*omega*cosh(beta*omega)/sinh(beta*omega) and
c     rgammap = m*omega/sinh(beta*omega).
```

```fortran
c
      IMPLICIT NONE
      INTEGER istep
      REAL pk,rmass,xk,dt,x,t,omega,pi,alpha
      COMPLEX eye,Psia,beta,A,rgamma,rgammap,c0,c1,c2
      common /packet/  rmass,xk,pk
      eye=(0.0,1.0)
      omega=1.0
      alpha = omega*rmass
      pi=acos(-1.0)
      beta = eye*dt*istep
      IF(abs(beta).EQ.0) beta = eye*1.0E-7
      A = sqrt(rmass*omega/(pi*(exp(beta*omega)-exp(-beta*omega))))
      rgamma=0.5*rmass*omega*(exp(beta*omega)+exp(-beta*omega))
     1      /(exp(beta*omega)-exp(-beta*omega))
      rgammap=2.*rmass*omega/(exp(beta*omega)-exp(-beta*omega))
      c0=-eye*pk*xk-alpha/2.*xk**2
      c1=rgammap*x+alpha*xk+eye*pk
      c2=rgamma+alpha/2.
c
      Psia = A*(alpha/pi)**.25*sqrt(pi/c2)*
     1      exp(-rgamma*x**2)*exp(c0+c1**2/(4.0*c2))
c
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE SAVEWF(je2,ndump,dt)
c
c     Dump Time Evolved Wave packet
c
      IMPLICIT NONE
      INTEGER je2,nptx,npts,kk,NN,ncount,ndump,jj
      COMPLEX chi,CRV,energy,psi,Psia
      character*9 B
      REAL V,x1,c1,c2,c1a,x,xmin,xmax,dx,EVALUES,dt
      PARAMETER(npts=9,nptx=2**npts,NN=1)
      DIMENSION CRV(NN,NN),energy(NN),EVALUES(NN)
      DIMENSION psi(NN,NN)
      common /xy/  xmin,xmax
      COMMON / wfunc/ chi(nptx,NN)
c
      CALL energies(energy)
      jj=je2/ndump
      write(B, '(A,i4.4)') 'arch.', jj
      OPEN(1,FILE=B)
      dx=(xmax-xmin)/real(nptx)
      ncount=(je2-1)/ndump
c
c     Save Wave-packet components
c
      do kk=1,nptx
```

285

```
           x=xmin+kk*dx
           c1=chi(kk,1)*conjg(chi(kk,1))
           c1a=Psia(x,je2,dt)*conjg(Psia(x,je2,dt))
           write(1,33) x,sqrt(c1)+real(energy(1))
      1         ,sqrt(c1a)+real(energy(1))
        end do
        write(1,33)
        do kk=1,nptx
           x=xmin+kk*dx
           write(1,33) x
      1         ,real(chi(kk,1))+real(energy(1))
      1         ,real(Psia(x,je2,dt))+real(energy(1))
        end do
        write(1,33)
c
c      Save Adiabatic states
c
        do kk=1,nptx
           x=xmin+kk*dx
           CALL HAMIL(CRV,x)
           write(1,33) x,CRV(1,1)
        end do
        CLOSE(1)
 33     format(6(e13.6,2x))
        RETURN
        END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        SUBROUTINE PE(RV)
c
c      Expectation Value of the Potential Enegy
c
        IMPLICIT NONE
        INTEGER nptx,npts,kk,NN,j
        COMPLEX chi,EYE,RV
        REAL Vpot,omega,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha
        PARAMETER(npts=9,nptx=2**npts,NN=1)
        DIMENSION RV(NN)
        COMMON / wfunc/ chi(nptx,NN)
        common /xy/   xmin,xmax
        common /packet/rmass,xk,pk

        dx=(xmax-xmin)/real(nptx)
        DO j=1,NN
           RV(j)=0.0
           do kk=1,nptx
              x=xmin+kk*dx
              IF(j.EQ.1) CALL VA(Vpot,x)
              RV(j)=RV(j)+chi(kk,j)*Vpot*conjg(chi(kk,j))*dx
           end do
        END DO
        RETURN
```

286

```
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine KE(RKE)
c
c     Expectation value of the kinetic energy
c
      IMPLICIT NONE
      INTEGER NN,kk,nptx,kx,nx,npts,j
      REAL dp,theta,wm,p,xmin,xmax,rmass,xk,pi,alenx,pk,rm,re,ri,dx
      COMPLEX eye,chi,Psip,chic,RKE
      parameter(npts=9,nptx=2**npts,NN=1)
      DIMENSION chic(nptx),RKE(NN)
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
      COMMON / wfunc/ chi(nptx,NN)
c
      pi = acos(-1.0)
      dx=(xmax-xmin)/nptx
      dp=2.*pi/(xmax-xmin)
c
      DO j=1,NN
         RKE(j)=0.0
         do kk=1,nptx
            chic(kk)=chi(kk,j)
         end do
         CALL fourn(chic,nptx,1,-1)
         do kx=1,nptx
            if(kx.le.(nptx/2+1)) then
               nx=kx-1
            else
               nx=kx-1-nptx
            end if
            p=0.
            if(nx.ne.0) p = real(nx)*dp
            chic(kx)=p**2/(2.0*rmass)*chic(kx)/nptx
         end do
         CALL fourn(chic,nptx,1,1)
         do kk=1,nptx
            RKE(j)=RKE(j)+conjg(chi(kk,j))*chic(kk)*dx
         end do
      END DO
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PROPAGATE(vprop,tprop)
c
c     Split Operator Fourier Transform Propagation Method
c     J. Comput. Phys. 47, 412 (1982); J. Chem. Phys. 78, 301 (1983)
c
      IMPLICIT NONE
      INTEGER i,j,NN,ii,nptx,npts
```

287

```fortran
      COMPLEX chi,vprop,chin1,chin2,tprop
      PARAMETER(npts=9,nptx=2**npts,NN=1)
      DIMENSION chin1(nptx),chin2(nptx)
      DIMENSION tprop(nptx),vprop(nptx,NN,NN)
      COMMON / wfunc/ chi(nptx,NN)
c
c     Apply potential energy part of the Trotter Expansion
c
      DO i=1,nptx
         chin1(i)=0.0
         DO j=1,NN
            chin1(i)=chin1(i)+vprop(i,1,j)*chi(i,j)
         END DO
      END DO
c
c     Fourier Transform wave-packet to the momentum representation
c
      CALL fourn(chin1,nptx,1,-1)
c
c     Apply kinetic energy part of the Trotter Expansion
c
      DO i=1,nptx
         chin1(i)=tprop(i)*chin1(i)
      END DO
c
c     Inverse Fourier Transform wave-packet to the coordinate representation
c
      CALL fourn(chin1,nptx,1,1)
c
c     Apply potential energy part of the Trotter Expansion
c
      DO i=1,nptx
         DO j=1,NN
            chi(i,j)=vprop(i,j,1)*chin1(i)
         END DO
      END DO
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutine for FFT from Numerical Recipes
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE FOURN(DATA,NN,NDIM,ISIGN)
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION NN(NDIM),DATA(*)
      NTOT=1
      DO 11 IDIM=1,NDIM
         NTOT=NTOT*NN(IDIM)
 11   CONTINUE
      NPREV=1
      DO 18 IDIM=1,NDIM
         N=NN(IDIM)
         NREM=NTOT/(N*NPREV)
```

288

```
         IP1=2*NPREV
         IP2=IP1*N
         IP3=IP2*NREM
         I2REV=1
         DO 14 I2=1,IP2,IP1
            IF(I2.LT.I2REV)THEN
               DO 13 I1=I2,I2+IP1-2,2
                  DO 12 I3=I1,IP3,IP2
                     I3REV=I2REV+I3-I2
                     TEMPR=DATA(I3)
                     TEMPI=DATA(I3+1)
                     DATA(I3)=DATA(I3REV)
                     DATA(I3+1)=DATA(I3REV+1)
                     DATA(I3REV)=TEMPR
                     DATA(I3REV+1)=TEMPI
12                CONTINUE
13             CONTINUE
            ENDIF
            IBIT=IP2/2
1           IF ((IBIT.GE.IP1).AND.(I2REV.GT.IBIT)) THEN
               I2REV=I2REV-IBIT
               IBIT=IBIT/2
               GO TO 1
            ENDIF
            I2REV=I2REV+IBIT
14       CONTINUE
         IFP1=IP1
2        IF(IFP1.LT.IP2)THEN
            IFP2=2*IFP1
            THETA=ISIGN*6.28318530717959D0/(IFP2/IP1)
            WPR=-2.D0*DSIN(0.5D0*THETA)**2
            WPI=DSIN(THETA)
            WR=1.D0
            WI=0.D0
            DO 17 I3=1,IFP1,IP1
               DO 16 I1=I3,I3+IP1-2,2
                  DO 15 I2=I1,IP3,IFP2
                     K1=I2
                     K2=K1+IFP1
                     TEMPR=SNGL(WR)*DATA(K2)-SNGL(WI)*DATA(K2+1)
                     TEMPI=SNGL(WR)*DATA(K2+1)+SNGL(WI)*DATA(K2)
                     DATA(K2)=DATA(K1)-TEMPR
                     DATA(K2+1)=DATA(K1+1)-TEMPI
                     DATA(K1)=DATA(K1)+TEMPR
                     DATA(K1+1)=DATA(K1+1)+TEMPI
15                CONTINUE
16             CONTINUE
               WTEMP=WR
               WR=WR*WPR-WI*WPI+WR
               WI=WI*WPR+WTEMP*WPI+WI
17       CONTINUE
```

```
            IFP1=IFP2
            GO TO 2
         ENDIF
         NPREV=N*NPREV
  18   CONTINUE
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

61.7 Problem 7

Computational Problem 7: Loop the computer program developed in Problem 5 with $x_0 = -2.5$ and $p_0 = 0$ for 100 steps with $\tau = 0.1$ a.u. For each step compute the expectation values of coordinates $x(t)$ and momenta $p(t)$ as done in Problems 3 and 4, respectively. Compare your calculations with the analytic solutions obtained by substituting Eq. (24) into Eq. (23). Verify that these correspond to the classical trajectories $x(t) = \bar{x} + (x_0 - \bar{x})\cos(\omega t)$ and $p(t) = p_0 - (x_0 - \bar{x})\omega m \sin(\omega t)$, which can be computed according to the Velocity-Verlet algorithm:

$$
\begin{aligned}
p_{j+1} &= p_j + (F(x_j) + F(x_{j+1}))\tau/2 \\
x_{j+1} &= x_j + p_j\tau/m + F(x_j)\tau^2/(2m).
\end{aligned}
\tag{1193}
$$

In order to visualize the output of this program, cut the source code attached below, compile it by typing

```
gfortran Problem7.f -o Problem7
```

run it by typing

```
./Problem7
```

Visualize the output of time dependent expectation values as compared to classical trajectories as follows: type

```
gnuplot
```

then type

```
set dat sty line
```

then type

```
plot ``traj.0000''
```

That will show the numerical computation of the expectation value $< \Psi_t|\hat{x}|\Psi_t >$ as a function of time. In order to visualize the classical result on top of the quantum mechanical expectation value, type

```
replot ``traj.0000'' u 1:4
```

In order to visualize the output of $< \Psi_t|\hat{p}|\Psi_t >$ as a function of time, type

```
plot ``traj.0000'' u 1:3
```

and to visualize the classical result on top of the quantum mechanical expectation value, type

```
replot ``traj.0000'' u 1:5
```

The plot of $< \Psi_t|\hat{p}|\Psi_t >$ vs. $< \Psi_t|\hat{x}|\Psi_t >$ can be obtained by typing

```
plot ``traj.0000'' u 3:2
```

, and the corresponding classical results $p(t)$ vs. $x(t)$

```
plot ``traj.0000'' u 5:4
```

To exit, type

```
quit
```

The snapshots of the time-dependent wave-packet can be visualized as a movie by typing

```
gnuplot<pp_7
```

where the file named

```
pp_7
```

has the following lines:

Download from (http://ursula.chem.yale.edu/~batista/P7/pp_7)

```
set yrange[0:6]
set xrange[-10:10]
set dat sty l
plot "arch.0001" u 1:2 lw 3
pause .1
plot "arch.0002" u 1:2 lw 3
pause .1
plot "arch.0003" u 1:2 lw 3
pause .1
plot "arch.0004" u 1:2 lw 3
pause .1
plot "arch.0005" u 1:2 lw 3
pause .1
plot "arch.0006" u 1:2 lw 3
pause .1
plot "arch.0007" u 1:2 lw 3
pause .1
plot "arch.0008" u 1:2 lw 3
pause .1
plot "arch.0009" u 1:2 lw 3
pause .1
plot "arch.0010" u 1:2 lw 3
pause .1
plot "arch.0011" u 1:2 lw 3
pause .1
plot "arch.0012" u 1:2 lw 3
pause .1
plot "arch.0013" u 1:2 lw 3
pause .1
plot "arch.0014" u 1:2 lw 3
```

```
pause .1
plot "arch.0015" u 1:2 lw 3
pause .1
plot "arch.0016" u 1:2 lw 3
pause .1
plot "arch.0017" u 1:2 lw 3
pause .1
plot "arch.0018" u 1:2 lw 3
pause .1
plot "arch.0019" u 1:2 lw 3
pause .1
plot "arch.0020" u 1:2 lw 3
pause .1
plot "arch.0021" u 1:2 lw 3
pause .1
plot "arch.0022" u 1:2 lw 3
pause .1
plot "arch.0023" u 1:2 lw 3
pause .1
plot "arch.0024" u 1:2 lw 3
pause .1
plot "arch.0025" u 1:2 lw 3
pause .1
plot "arch.0026" u 1:2 lw 3
pause .1
plot "arch.0027" u 1:2 lw 3
pause .1
plot "arch.0028" u 1:2 lw 3
pause .1
plot "arch.0029" u 1:2 lw 3
pause .1
plot "arch.0030" u 1:2 lw 3
pause .1
plot "arch.0031" u 1:2 lw 3
pause .1
plot "arch.0032" u 1:2 lw 3
pause .1
plot "arch.0033" u 1:2 lw 3
pause .1
plot "arch.0034" u 1:2 lw 3
pause .1
plot "arch.0035" u 1:2 lw 3
pause .1
plot "arch.0036" u 1:2 lw 3
pause .1
plot "arch.0037" u 1:2 lw 3
pause .1
plot "arch.0038" u 1:2 lw 3
pause .1
plot "arch.0039" u 1:2 lw 3
pause .1
```

```
plot  "arch.0040" u 1:2 lw 3
pause .1
plot "arch.0041" u 1:2 lw 3
pause .1
plot "arch.0042" u 1:2 lw 3
pause .1
plot "arch.0043" u 1:2 lw 3
pause .1
plot "arch.0044" u 1:2 lw 3
pause .1
plot "arch.0045" u 1:2 lw 3
pause .1
plot "arch.0046" u 1:2 lw 3
pause .1
plot "arch.0047" u 1:2 lw 3
pause .1
plot "arch.0048" u 1:2 lw 3
pause .1
plot "arch.0049" u 1:2 lw 3
pause .1
plot "arch.0050" u 1:2 lw 3
pause .1
plot "arch.0051" u 1:2 lw 3
pause .1
plot "arch.0052" u 1:2 lw 3
pause .1
plot "arch.0053" u 1:2 lw 3
pause .1
plot "arch.0054" u 1:2 lw 3
pause .1
plot "arch.0055" u 1:2 lw 3
pause .1
plot "arch.0056" u 1:2 lw 3
pause .1
plot "arch.0057" u 1:2 lw 3
pause .1
plot "arch.0058" u 1:2 lw 3
pause .1
plot "arch.0059" u 1:2 lw 3
pause .1
plot "arch.0060" u 1:2 lw 3
pause .1
plot "arch.0061" u 1:2 lw 3
pause .1
plot "arch.0062" u 1:2 lw 3
pause .1
plot "arch.0063" u 1:2 lw 3
pause .1
plot "arch.0064" u 1:2 lw 3
pause .1
plot "arch.0065" u 1:2 lw 3
```

```
pause .1
plot "arch.0066" u 1:2 lw 3
pause .1
plot "arch.0067" u 1:2 lw 3
pause .1
plot "arch.0068" u 1:2 lw 3
pause .1
plot "arch.0069" u 1:2 lw 3
pause .1
plot "arch.0070" u 1:2 lw 3
pause .1
plot "arch.0071" u 1:2 lw 3
pause .1
plot "arch.0072" u 1:2 lw 3
pause .1
plot "arch.0073" u 1:2 lw 3
pause .1
plot "arch.0074" u 1:2 lw 3
pause .1
plot "arch.0075" u 1:2 lw 3
pause .1
plot "arch.0076" u 1:2 lw 3
pause .1
plot "arch.0077" u 1:2 lw 3
pause .1
plot "arch.0078" u 1:2 lw 3
pause .1
plot "arch.0079" u 1:2 lw 3
pause .1
plot  "arch.0080" u 1:2 lw 3
pause .1
plot "arch.0081" u 1:2 lw 3
pause .1
plot "arch.0082" u 1:2 lw 3
pause .1
plot "arch.0083" u 1:2 lw 3
pause .1
plot "arch.0084" u 1:2 lw 3
pause .1
plot "arch.0085" u 1:2 lw 3
pause .1
plot "arch.0086" u 1:2 lw 3
pause .1
plot "arch.0087" u 1:2 lw 3
pause .1
plot "arch.0088" u 1:2 lw 3
pause .1
plot "arch.0089" u 1:2 lw 3
pause .1
plot  "arch.0090" u 1:2 lw 3
pause .1
```

```
plot "arch.0091" u 1:2 lw 3
pause .1
plot "arch.0092" u 1:2 lw 3
pause .1
plot "arch.0093" u 1:2 lw 3
pause .1
plot "arch.0094" u 1:2 lw 3
pause .1
plot "arch.0095" u 1:2 lw 3
pause .1
plot "arch.0096" u 1:2 lw 3
pause .1
plot "arch.0097" u 1:2 lw 3
pause .1
plot "arch.0098" u 1:2 lw 3
pause .1
plot "arch.0099" u 1:2 lw 3
pause .1
```

```fortran
      PROGRAM Problem7
c
c     1-D wave packet propagation and Velocity-Verlet propagation
c     on a Harmonic potential energy surface
c
      IMPLICIT NONE
      INTEGER NN,npts,nptx,ndump
      INTEGER istep,nstep,jj
      REAL dt,xc,pc
      COMPLEX vprop,tprop,x_mean,p_mean
      character*9 Bfile
      PARAMETER(npts=9,nptx=2**npts,NN=1)
      DIMENSION vprop(nptx,NN,NN),tprop(nptx)
      DIMENSION x_mean(NN),p_mean(NN)
      COMMON /class/ xc,pc
c
      jj=0
      write(Bfile, '(A,i4.4)') 'traj.', jj
      OPEN(10,FILE=Bfile)
      CALL ReadParam(nstep,ndump,dt)
      call Initialize()
      CALL SetKinProp(dt,tprop)
      CALL SetPotProp(dt,vprop)
      DO istep=1,nstep+1
         IF(mod(istep-1,10).EQ.0)
     1        PRINT *, "Step=", istep-1,", Final step=", nstep
         IF(istep.GE.1) CALL PROPAGATE(vprop,tprop)
         IF(mod((istep-1),ndump).EQ.0) THEN
            CALL SAVEWF(istep,ndump,dt)
            CALL XM(x_mean)
            CALL PM(p_mean)
            CALL VV(dt)
            WRITE(10,22) (istep-1.)*dt
     1           ,real(x_mean(1)),real(p_mean(1)),xc,pc
         END IF
      END DO
      CLOSE(10)
 22   FORMAT(6(e13.6,2x))
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine ReadParam(nstep,ndump,dt)
c
c     Parameters defining the grid (xmin, xmax), integration time step (dt),
c     rmass (rmass), initial position (xk), initial momentum (pk),
c     number of propagation steps (nstep), and how often to save a pic (ndump)
c
      IMPLICIT NONE
      INTEGER ntype,nstep,nrpt,ireport,ndump,nlit
      REAL xmin,xmax,pk,rmass,xk,dt
```

297

```fortran
      common /packet/  rmass,xk,pk
      common /xy/  xmin,xmax
c
      xmin=-10.0
      xmax= 10.0
      dt=0.1
      rmass=1.0
      xk=-2.5
      pk=0.0
      nstep=100
      ndump=1
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE VV(dt)
c
c     Velocity Verlet Algorithm J. Chem. Phys. 76, 637 (1982)
c
      IMPLICIT NONE
      REAL v,dx,dt,xc,pc,rmass,xk,pk,acc,xt,VPOT1,VPOT2,F
      COMMON /class/ xc,pc
      common /packet/  rmass,xk,pk
c
c     Compute Force
c
      dx=0.01
      xt=xc+dx
      CALL VA(VPOT1,xt)
      xt=xc-dx
      CALL VA(VPOT2,xt)
      F=-(VPOT1-VPOT2)/(2.0*dx)
      v=pc/rmass
c
c     Advance momenta half a step
c
      pc=pc+0.5*F*dt
c
c     Advance coordinates a step
c
      xc=xc+v*dt+0.5*dt**2*F/rmass
c
c     Compute Force
c
      dx=0.01
      xt=xc+dx
      CALL VA(VPOT1,xt)
      xt=xc-dx
      CALL VA(VPOT2,xt)
      F=-(VPOT1-VPOT2)/(2.0*dx)
c
```

```
c      Advance momenta half a step
c
       pc=pc+0.5*F*dt
c
       return
       end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       SUBROUTINE Initialize()

       IMPLICIT NONE
       INTEGER NN,nptx,npts,kk
       COMPLEX chi0,chi,EYE,CRV
       REAL xc,pc,omega,xk2,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha,alpha2
       PARAMETER(npts=9,nptx=2**npts,NN=1)
       DIMENSION CRV(NN,NN)
       common /xy/   xmin,xmax
       common /packet/  rmass,xk,pk
       COMMON / wfunc/ chi(nptx,NN)
       COMMON / iwfunc/ chi0(nptx,NN)
       COMMON /class/ xc,pc

       EYE=(0.0,1.0)
       pi= acos(-1.0)
       omega=1.
       dx=(xmax-xmin)/real(nptx)
       xc=xk
       pc=pk
c
c      Wave Packet Initialization: Gaussian centered at xk, with momentum pk
c
       alpha=rmass*omega
       do kk=1,nptx
          x=xmin+kk*dx
          chi(kk,1)=((alpha/pi)**0.25)
     1         *exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
          chi0(kk,1)=chi(kk,1)
       end do
       RETURN
       END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       SUBROUTINE HAMIL(CRV,x)
c
c      Hamiltonian Matrix
c
       IMPLICIT NONE
       INTEGER NN
       REAL x,VPOT1
       COMPLEX CRV
       PARAMETER(NN=1)
       DIMENSION CRV(NN,NN)
c
```

299

```fortran
      CALL VA(VPOT1,x)
      CRV(1,1)=VPOT1
c
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE VA(V,x)
c
c     Potential Energy Surface: Harmonic Oscillator
c
      implicit none
      REAL V,x,rmass,xk,pk,rk,omega
      common /packet/ rmass,xk,pk
      omega=1.0
      rk=rmass*omega**2
      V=0.5*rk*x*x
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetKinProp(dt,tprop)
c
c     Kinetic Energy part of the Trotter Expansion: exp(-i p^2 dt/(2 m))
c
      IMPLICIT NONE
      INTEGER nptx,kx,nx,npts
      REAL xsc,xmin,xmax,propfacx,rmass,xk,pi,alenx,dt,pk
      COMPLEX tprop,eye
      parameter(npts=9,nptx=2**npts)
      DIMENSION tprop(nptx)
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
c
      eye=(0.,1.)
      pi = acos(-1.0)
      alenx=xmax-xmin
      propfacx=-dt/2./rmass*(2.*pi)**2
      do kx=1,nptx
         if(kx.le.(nptx/2+1)) then
            nx=kx-1
         else
            nx=kx-1-nptx
         end if
         xsc=0.
         if(nx.ne.0) xsc=real(nx)/alenx
         tprop(kx)=exp(eye*(propfacx*xsc**2))
       end do
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetPotProp(dt,vprop)
```

```fortran
c
c      Potential Energy part of the Trotter Expansion: exp(-i V dt/2)
c
       IMPLICIT NONE
       INTEGER NN,ii,nptx,npts
       REAL xmin,xmax,dx,dt,x,VPOT
       COMPLEX vprop,eye
       parameter(npts=9,nptx=2**npts,NN=1)
       DIMENSION vprop(nptx,NN,NN)
       common /xy/  xmin,xmax
       eye=(0.,1.)
       dx=(xmax-xmin)/real(nptx)
c
       do ii=1,nptx
          x=xmin+ii*dx
          CALL VA(VPOT,x)
          vprop(ii,1,1)=exp(-eye*0.5*dt*VPOT)/sqrt(nptx*1.0)
       END DO
       RETURN
       END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       SUBROUTINE energies(energy)
       IMPLICIT NONE
       INTEGER j,NN
       COMPLEX energy,RV,RKE
       PARAMETER (NN=1)
       DIMENSION RV(NN),RKE(NN),energy(NN)
       CALL PE(RV)
       CALL KE(RKE)
       DO j=1,NN
          energy(j)=RV(j)+RKE(j)
       END DO
       RETURN
       END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       FUNCTION Psia(x,istep,dt)
c
c      Analytic wave-packet <x|Psia(istep)> obtained by applying the
c      harmonic propagator to the initial state,
c      <x'|Psi(0)> = (alpha/pi)**.25*exp(-alpha/2*(x'-xk)**2+eye*pk*(x'-xk)),
c      where the propagator is
c      <x|exp(-beta H)|x'> = A exp(-rgamma*(x**2+x'**2)+rgammap*x*x'), with
c      A = sqrt(m*omega/(pi*(exp(beta*omega)-exp(-beta*omega)))), beta = i*t,
c      rgamma = 0.5*m*omega*cosh(beta*omega)/sinh(beta*omega) and
c      rgammap = m*omega/sinh(beta*omega).
c
       IMPLICIT NONE
       INTEGER istep
       REAL pk,rmass,xk,dt,x,t,omega,pi,alpha
       COMPLEX eye,Psia,beta,A,rgamma,rgammap,c0,c1,c2
       common /packet/  rmass,xk,pk
```

```
        eye=(0.0,1.0)
        omega=1.0
        alpha = omega*rmass
        pi=acos(-1.0)
        beta = eye*dt*istep
        IF(abs(beta).EQ.0) beta = eye*1.0E-7
        A = sqrt(rmass*omega/(pi*(exp(beta*omega)-exp(-beta*omega))))
        rgamma=0.5*rmass*omega*(exp(beta*omega)+exp(-beta*omega))
     1     /(exp(beta*omega)-exp(-beta*omega))
        rgammap=2.*rmass*omega/(exp(beta*omega)-exp(-beta*omega))
        c0=-eye*pk*xk-alpha/2.*xk**2
        c1=rgammap*x+alpha*xk+eye*pk
        c2=rgamma+alpha/2.
c
        Psia = A*(alpha/pi)**.25*sqrt(pi/c2)*
     1     exp(-rgamma*x**2)*exp(c0+c1**2/(4.0*c2))
c
        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        SUBROUTINE SAVEWF(je2,ndump,dt)
c
c     Dump Time Evolved Wave packet
c
        IMPLICIT NONE
        INTEGER je2,nptx,npts,kk,NN,ncount,ndump,jj
        COMPLEX chi,CRV,energy,psi,Psia
        character*9 B
        REAL V,x1,c1,c2,c1a,x,xmin,xmax,dx,EVALUES,dt
        PARAMETER(npts=9,nptx=2**npts,NN=1)
        DIMENSION CRV(NN,NN),energy(NN),EVALUES(NN)
        DIMENSION psi(NN,NN)
        common /xy/  xmin,xmax
        COMMON / wfunc/ chi(nptx,NN)
c
        CALL energies(energy)
        jj=je2/ndump
        write(B, '(A,i4.4)') 'arch.', jj
        OPEN(1,FILE=B)
        dx=(xmax-xmin)/real(nptx)
        ncount=(je2-1)/ndump
c
c     Save Wave-packet components
c
        do kk=1,nptx
           x=xmin+kk*dx
           c1=chi(kk,1)*conjg(chi(kk,1))
           c1a=Psia(x,je2,dt)*conjg(Psia(x,je2,dt))
           write(1,33) x,sqrt(c1)+real(energy(1))
     1          ,sqrt(c1a)+real(energy(1))
        end do
```

```
      write(1,33)
       do kk=1,nptx
          x=xmin+kk*dx
          write(1,33) x
     1          ,real(chi(kk,1))+real(energy(1))
     1          ,real(Psia(x,je2,dt))+real(energy(1))
       end do
       write(1,33)
c
c      Save Adiabatic states
c
       do kk=1,nptx
          x=xmin+kk*dx
          CALL HAMIL(CRV,x)
          write(1,33) x,CRV(1,1)
       end do
       CLOSE(1)
 33    format(6(e13.6,2x))
       RETURN
       END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       SUBROUTINE XM(RV)
c
c      Expectation Value of the Position
c
       IMPLICIT NONE
       INTEGER nptx,npts,kk,NN,j
       COMPLEX chi,EYE,RV
       REAL Vpot,omega,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha
       PARAMETER(npts=9,nptx=2**npts,NN=1)
       DIMENSION RV(NN)
       COMMON / wfunc/ chi(nptx,NN)
       common /xy/   xmin,xmax
       common /packet/rmass,xk,pk

       dx=(xmax-xmin)/real(nptx)
       DO j=1,NN
          RV(j)=0.0
          do kk=1,nptx
             x=xmin+kk*dx
             IF(j.EQ.1) CALL VA(Vpot,x)
             RV(j)=RV(j)+chi(kk,j)*x*conjg(chi(kk,j))*dx
          end do
       END DO
       RETURN
       END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       SUBROUTINE PE(RV)
c
c      Expectation Value of the Potential Enegy
c
```

303

```fortran
      IMPLICIT NONE
      INTEGER nptx,npts,kk,NN,j
      COMPLEX chi,EYE,RV
      REAL Vpot,omega,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha
      PARAMETER(npts=9,nptx=2**npts,NN=1)
      DIMENSION RV(NN)
      COMMON / wfunc/ chi(nptx,NN)
      common /xy/  xmin,xmax
      common /packet/rmass,xk,pk

      dx=(xmax-xmin)/real(nptx)
      DO j=1,NN
         RV(j)=0.0
         do kk=1,nptx
            x=xmin+kk*dx
            IF(j.EQ.1) CALL VA(Vpot,x)
            RV(j)=RV(j)+chi(kk,j)*Vpot*conjg(chi(kk,j))*dx
         end do
      END DO
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine KE(RKE)
c
c     Expectation value of the kinetic energy
c
      IMPLICIT NONE
      INTEGER NN,kk,nptx,kx,nx,npts,j
      REAL dp,theta,wm,p,xmin,xmax,rmass,xk,pi,alenx,pk,rm,re,ri,dx
      COMPLEX eye,chi,Psip,chic,RKE
      parameter(npts=9,nptx=2**npts,NN=1)
      DIMENSION chic(nptx),RKE(NN)
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
      COMMON / wfunc/ chi(nptx,NN)
c
      pi = acos(-1.0)
      dx=(xmax-xmin)/nptx
      dp=2.*pi/(xmax-xmin)
c
      DO j=1,NN
         RKE(j)=0.0
         do kk=1,nptx
            chic(kk)=chi(kk,j)
         end do
         CALL fourn(chic,nptx,1,-1)
         do kx=1,nptx
            if(kx.le.(nptx/2+1)) then
               nx=kx-1
            else
               nx=kx-1-nptx
```

304

```fortran
            end if
            p=0.
            if(nx.ne.0) p = real(nx)*dp
            chic(kx)=p**2/(2.0*rmass)*chic(kx)/nptx
         end do
         CALL fourn(chic,nptx,1,1)
         do kk=1,nptx
            RKE(j)=RKE(j)+conjg(chi(kk,j))*chic(kk)*dx
         end do
      END DO
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine PM(RKE)
c
c     Expectation value of the kinetic energy
c
      IMPLICIT NONE
      INTEGER NN,kk,nptx,kx,nx,npts,j
      REAL dp,theta,wm,p,xmin,xmax,rmass,xk,pi,alenx,pk,rm,re,ri,dx
      COMPLEX eye,chi,Psip,chic,RKE
      parameter(npts=9,nptx=2**npts,NN=1)
      DIMENSION chic(nptx),RKE(NN)
      common /xy/   xmin,xmax
      common /packet/   rmass,xk,pk
      COMMON / wfunc/ chi(nptx,NN)
c
      pi = acos(-1.0)
      dx=(xmax-xmin)/nptx
      dp=2.*pi/(xmax-xmin)
c
      DO j=1,NN
         RKE(j)=0.0
         do kk=1,nptx
            chic(kk)=chi(kk,j)
         end do
         CALL fourn(chic,nptx,1,-1)
         do kx=1,nptx
            if(kx.le.(nptx/2+1)) then
               nx=kx-1
            else
               nx=kx-1-nptx
            end if
            p=0.
            if(nx.ne.0) p = real(nx)*dp
            chic(kx)=p*chic(kx)/nptx
         end do
         CALL fourn(chic,nptx,1,1)
         do kk=1,nptx
            RKE(j)=RKE(j)+conjg(chi(kk,j))*chic(kk)*dx
         end do
```

```fortran
      END DO
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PROPAGATE(vprop,tprop)
c
c     Split Operator Fourier Transform Propagation Method
c     J. Comput. Phys. 47, 412 (1982); J. Chem. Phys. 78, 301 (1983)
c
      IMPLICIT NONE
      INTEGER i,j,NN,ii,nptx,npts
      COMPLEX chi,vprop,chin1,chin2,tprop
      PARAMETER(npts=9,nptx=2**npts,NN=1)
      DIMENSION chin1(nptx),chin2(nptx)
      DIMENSION tprop(nptx),vprop(nptx,NN,NN)
      COMMON / wfunc/ chi(nptx,NN)
c
c     Apply potential energy part of the Trotter Expansion
c
      DO i=1,nptx
         chin1(i)=0.0
         DO j=1,NN
            chin1(i)=chin1(i)+vprop(i,1,j)*chi(i,j)
         END DO
      END DO
c
c     Fourier Transform wave-packet to the momentum representation
c
      CALL fourn(chin1,nptx,1,-1)
c
c     Apply kinetic energy part of the Trotter Expansion
c
      DO i=1,nptx
         chin1(i)=tprop(i)*chin1(i)
      END DO
c
c     Inverse Fourier Transform wave-packet to the coordinate representation
c
      CALL fourn(chin1,nptx,1,1)
c
c     Apply potential energy part of the Trotter Expansion
c
      DO i=1,nptx
         DO j=1,NN
            chi(i,j)=vprop(i,j,1)*chin1(i)
         END DO
      END DO
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutine for FFT from Numerical Recipes
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
      SUBROUTINE FOURN(DATA,NN,NDIM,ISIGN)
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION NN(NDIM),DATA(*)
      NTOT=1
      DO 11 IDIM=1,NDIM
         NTOT=NTOT*NN(IDIM)
11    CONTINUE
      NPREV=1
      DO 18 IDIM=1,NDIM
         N=NN(IDIM)
         NREM=NTOT/(N*NPREV)
         IP1=2*NPREV
         IP2=IP1*N
         IP3=IP2*NREM
         I2REV=1
         DO 14 I2=1,IP2,IP1
            IF(I2.LT.I2REV)THEN
               DO 13 I1=I2,I2+IP1-2,2
                  DO 12 I3=I1,IP3,IP2
                     I3REV=I2REV+I3-I2
                     TEMPR=DATA(I3)
                     TEMPI=DATA(I3+1)
                     DATA(I3)=DATA(I3REV)
                     DATA(I3+1)=DATA(I3REV+1)
                     DATA(I3REV)=TEMPR
                     DATA(I3REV+1)=TEMPI
12                CONTINUE
13             CONTINUE
            ENDIF
            IBIT=IP2/2
1           IF ((IBIT.GE.IP1).AND.(I2REV.GT.IBIT)) THEN
               I2REV=I2REV-IBIT
               IBIT=IBIT/2
               GO TO 1
            ENDIF
            I2REV=I2REV+IBIT
14       CONTINUE
         IFP1=IP1
2        IF(IFP1.LT.IP2)THEN
            IFP2=2*IFP1
            THETA=ISIGN*6.28318530717959D0/(IFP2/IP1)
            WPR=-2.D0*DSIN(0.5D0*THETA)**2
            WPI=DSIN(THETA)
            WR=1.D0
            WI=0.D0
            DO 17 I3=1,IFP1,IP1
               DO 16 I1=I3,I3+IP1-2,2
                  DO 15 I2=I1,IP3,IFP2
                     K1=I2
                     K2=K1+IFP1
                     TEMPR=SNGL(WR)*DATA(K2)-SNGL(WI)*DATA(K2+1)
```

```
                    TEMPI=SNGL(WR)*DATA(K2+1)+SNGL(WI)*DATA(K2)
                    DATA(K2)=DATA(K1)-TEMPR
                    DATA(K2+1)=DATA(K1+1)-TEMPI
                    DATA(K1)=DATA(K1)+TEMPR
                    DATA(K1+1)=DATA(K1+1)+TEMPI
15              CONTINUE
16            CONTINUE
              WTEMP=WR
              WR=WR*WPR-WI*WPI+WR
              WI=WI*WPR+WTEMP*WPI+WI
17          CONTINUE
            IFP1=IFP2
            GO TO 2
          ENDIF
          NPREV=N*NPREV
18    CONTINUE
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

61.8 Problem 8

Computational Problem 8: Change the potential to that of a Morse oscillator $V(\hat{x}) = De(1 - \exp(-a(\hat{x} - x_e)))^2$, with $x_e = 0$, $De = 8$, and $a = \sqrt{k/(2D_e)}$, where $k = m\omega^2$. Recompute the wave-packet propagation with $x_0 = -0.5$ and $p_0 = 0$ for 100 steps with $\tau = 0.1$ a.u., and compare the expectation values $x(t)$ and $p(t)$ with the corresponding classical trajectories obtained by recursively applying the Velocity-Verlet algorithm.

The output of this program is analogous to Problem 6 but for a Morse potential. Cut the source code attached below, save it in a file named Problem8.f, compile it by typing

```
gfortran Problem8.f -o Problem8
```

run it by typing

```
./Problem8
```

Visualize the output of the time dependent expectation values as compared to classical trajectories as follows: type

```
gnuplot
```

then type

```
set dat sty line
```

then type

```
plot ``traj.0000''
```

That will show the numerical computation of the expectation value $< \Psi_t|\hat{x}|\Psi_t >$ as a function of time. In order to visualize the classical result on top of the quantum mechanical expectation value, type

```
replot ``traj.0000'' u 1:4
```

In order to visualize the output of $< \Psi_t|\hat{p}|\Psi_t >$ as a function of time, type

```
plot ``traj.0000'' u 1:3
```

and to visualize the classical result on top of the quantum mechanical expectation value, type

```
replot ``traj.0000'' u 1:5
```

The plot of $< \Psi_t|\hat{p}|\Psi_t >$ vs. $< \Psi_t|\hat{x}|\Psi_t >$ can be obtained by typing

```
plot ``traj.0000'' u 3:2
```

and the corresponding classical results $p(t)$ vs. $x(t)$

```
plot ``traj.0000'' u 5:4
```

To exit, type

```
quit
```

The snapshots of the time-dependent wave-packet can be visualized as a movie by typing

```
gnuplot<pp_8
```

where the file named

```
pp_8
```

has the following lines:

Download from (http://ursula.chem.yale.edu/~batista/P8/pp_8)

```
set yrange[0:9]
set xrange[-5:25]
set dat sty l
plot "arch.0001" u 1:2 lw 3
pause .1
plot "arch.0002" u 1:2 lw 3
pause .1
plot "arch.0003" u 1:2 lw 3
pause .1
plot "arch.0004" u 1:2 lw 3
pause .1
plot "arch.0005" u 1:2 lw 3
pause .1
plot "arch.0006" u 1:2 lw 3
pause .1
plot "arch.0007" u 1:2 lw 3
pause .1
plot "arch.0008" u 1:2 lw 3
pause .1
plot "arch.0009" u 1:2 lw 3
pause .1
plot "arch.0010" u 1:2 lw 3
pause .1
plot "arch.0011" u 1:2 lw 3
pause .1
plot "arch.0012" u 1:2 lw 3
pause .1
plot "arch.0013" u 1:2 lw 3
pause .1
plot "arch.0014" u 1:2 lw 3
pause .1
plot "arch.0015" u 1:2 lw 3
pause .1
plot "arch.0016" u 1:2 lw 3
pause .1
plot "arch.0017" u 1:2 lw 3
```

```
pause .1
plot "arch.0018" u 1:2 lw 3
pause .1
plot "arch.0019" u 1:2 lw 3
pause .1
plot "arch.0020" u 1:2 lw 3
pause .1
plot "arch.0021" u 1:2 lw 3
pause .1
plot "arch.0022" u 1:2 lw 3
pause .1
plot "arch.0023" u 1:2 lw 3
pause .1
plot "arch.0024" u 1:2 lw 3
pause .1
plot "arch.0025" u 1:2 lw 3
pause .1
plot "arch.0026" u 1:2 lw 3
pause .1
plot "arch.0027" u 1:2 lw 3
pause .1
plot "arch.0028" u 1:2 lw 3
pause .1
plot "arch.0029" u 1:2 lw 3
pause .1
plot "arch.0030" u 1:2 lw 3
pause .1
plot "arch.0031" u 1:2 lw 3
pause .1
plot "arch.0032" u 1:2 lw 3
pause .1
plot "arch.0033" u 1:2 lw 3
pause .1
plot "arch.0034" u 1:2 lw 3
pause .1
plot "arch.0035" u 1:2 lw 3
pause .1
plot "arch.0036" u 1:2 lw 3
pause .1
plot "arch.0037" u 1:2 lw 3
pause .1
plot "arch.0038" u 1:2 lw 3
pause .1
plot "arch.0039" u 1:2 lw 3
pause .1
plot  "arch.0040" u 1:2 lw 3
pause .1
plot "arch.0041" u 1:2 lw 3
pause .1
plot "arch.0042" u 1:2 lw 3
pause .1
```

```
plot "arch.0043" u 1:2 lw 3
pause .1
plot "arch.0044" u 1:2 lw 3
pause .1
plot "arch.0045" u 1:2 lw 3
pause .1
plot "arch.0046" u 1:2 lw 3
pause .1
plot "arch.0047" u 1:2 lw 3
pause .1
plot "arch.0048" u 1:2 lw 3
pause .1
plot "arch.0049" u 1:2 lw 3
pause .1
plot "arch.0050" u 1:2 lw 3
pause .1
plot "arch.0051" u 1:2 lw 3
pause .1
plot "arch.0052" u 1:2 lw 3
pause .1
plot "arch.0053" u 1:2 lw 3
pause .1
plot "arch.0054" u 1:2 lw 3
pause .1
plot "arch.0055" u 1:2 lw 3
pause .1
plot "arch.0056" u 1:2 lw 3
pause .1
plot "arch.0057" u 1:2 lw 3
pause .1
plot "arch.0058" u 1:2 lw 3
pause .1
plot "arch.0059" u 1:2 lw 3
pause .1
plot "arch.0060" u 1:2 lw 3
pause .1
plot "arch.0061" u 1:2 lw 3
pause .1
plot "arch.0062" u 1:2 lw 3
pause .1
plot "arch.0063" u 1:2 lw 3
pause .1
plot "arch.0064" u 1:2 lw 3
pause .1
plot "arch.0065" u 1:2 lw 3
pause .1
plot "arch.0066" u 1:2 lw 3
pause .1
plot "arch.0067" u 1:2 lw 3
pause .1
plot "arch.0068" u 1:2 lw 3
```

```
pause .1
plot "arch.0069" u 1:2 lw 3
pause .1
plot "arch.0070" u 1:2 lw 3
pause .1
plot "arch.0071" u 1:2 lw 3
pause .1
plot "arch.0072" u 1:2 lw 3
pause .1
plot "arch.0073" u 1:2 lw 3
pause .1
plot "arch.0074" u 1:2 lw 3
pause .1
plot "arch.0075" u 1:2 lw 3
pause .1
plot "arch.0076" u 1:2 lw 3
pause .1
plot "arch.0077" u 1:2 lw 3
pause .1
plot "arch.0078" u 1:2 lw 3
pause .1
plot "arch.0079" u 1:2 lw 3
pause .1
plot  "arch.0080" u 1:2 lw 3
pause .1
plot "arch.0081" u 1:2 lw 3
pause .1
plot "arch.0082" u 1:2 lw 3
pause .1
plot "arch.0083" u 1:2 lw 3
pause .1
plot "arch.0084" u 1:2 lw 3
pause .1
plot "arch.0085" u 1:2 lw 3
pause .1
plot "arch.0086" u 1:2 lw 3
pause .1
plot "arch.0087" u 1:2 lw 3
pause .1
plot "arch.0088" u 1:2 lw 3
pause .1
plot "arch.0089" u 1:2 lw 3
pause .1
plot  "arch.0090" u 1:2 lw 3
pause .1
plot "arch.0091" u 1:2 lw 3
pause .1
plot "arch.0092" u 1:2 lw 3
pause .1
plot "arch.0093" u 1:2 lw 3
pause .1
```

```
plot "arch.0094" u 1:2 lw 3
pause .1
plot "arch.0095" u 1:2 lw 3
pause .1
plot "arch.0096" u 1:2 lw 3
pause .1
plot "arch.0097" u 1:2 lw 3
pause .1
plot "arch.0098" u 1:2 lw 3
pause .1
plot "arch.0099" u 1:2 lw 3
pause .1
```

```
      PROGRAM Problem8
c
c     1-D wave packet propagation and Velocity-Verlet propagation
c     on a Morse potential energy surface
c
      IMPLICIT NONE
      INTEGER NN,npts,nptx,ndump
      INTEGER istep,nstep,jj
      REAL dt,xc,pc
      COMPLEX vprop,tprop,x_mean,p_mean
      character*9 Bfile
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      DIMENSION vprop(nptx,NN,NN),tprop(nptx)
      DIMENSION x_mean(NN),p_mean(NN)
      COMMON /class/ xc,pc
c     xo
      jj=0
      write(Bfile, '(A,i4.4)') 'traj.', jj
      OPEN(10,FILE=Bfile)
      CALL ReadParam(nstep,ndump,dt)
      call Initialize()
      CALL SetKinProp(dt,tprop)
      CALL SetPotProp(dt,vprop)
      DO istep=1,nstep+1
         IF(mod(istep-1,10).EQ.0)
     1        PRINT *, "Step=", istep-1,", Final step=", nstep
         IF(istep.GE.1) CALL PROPAGATE(vprop,tprop)
         IF(mod((istep-1),ndump).EQ.0) THEN
            CALL SAVEWF(istep,ndump,dt)
            CALL XM(x_mean)
            CALL PM(p_mean)
            CALL VV(dt)
            WRITE(10,22) (istep-1.)*dt
     1           ,real(x_mean(1)),real(p_mean(1)),xc,pc
         END IF
      END DO
      CLOSE(10)
 22   FORMAT(6(e13.6,2x))
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine ReadParam(nstep,ndump,dt)
c
c     Parameters defining the grid (xmin, xmax), integration time step (dt),
c     rmass (rmass), initial position (xk), initial momentum (pk),
c     number of propagation steps (nstep), and how often to save a pic (ndump)
c
      IMPLICIT NONE
      INTEGER ntype,nstep,nrpt,ireport,ndump,nlit
      REAL xmin,xmax,pk,rmass,xk,dt
```

315

```
      common /packet/  rmass,xk,pk
      common /xy/  xmin,xmax
c
      xmin=-5.0
      xmax=25.0
      dt=0.2
      rmass=1.0
      xk=-.5
      pk=0.0
      nstep=100
      ndump=1
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE VV(dt)
c
c     Velocity Verlet Algorithm J. Chem. Phys. 76, 637 (1982)
c
      IMPLICIT NONE
      REAL v,dx,dt,xc,pc,rmass,xk,pk,acc,xt,VPOT1,VPOT2,F
      COMMON /class/ xc,pc
      common /packet/  rmass,xk,pk
c
c     Compute Force
c
      dx=0.01
      xt=xc+dx
      CALL VA(VPOT1,xt)
      xt=xc-dx
      CALL VA(VPOT2,xt)
      F=-(VPOT1-VPOT2)/(2.0*dx)
      v=pc/rmass
c
c     Advance momenta half a step
c
      pc=pc+0.5*F*dt
c
c     Advance coordinates a step
c
      xc=xc+v*dt+0.5*dt**2*F/rmass
c
c     Compute Force
c
      dx=0.01
      xt=xc+dx
      CALL VA(VPOT1,xt)
      xt=xc-dx
      CALL VA(VPOT2,xt)
      F=-(VPOT1-VPOT2)/(2.0*dx)
c
```

```fortran
c     Advance momenta half a step
c
      pc=pc+0.5*F*dt
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Initialize()

      IMPLICIT NONE
      INTEGER NN,nptx,npts,kk
      COMPLEX chi0,chi,EYE,CRV
      REAL xc,pc,omega,xk2,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha,alpha2
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      DIMENSION CRV(NN,NN)
      common /xy/   xmin,xmax
      common /packet/  rmass,xk,pk
      COMMON / wfunc/ chi(nptx,NN)
      COMMON / iwfunc/ chi0(nptx,NN)
      COMMON /class/ xc,pc

      EYE=(0.0,1.0)
      pi= acos(-1.0)
      omega=1.
      dx=(xmax-xmin)/real(nptx)
      xc=xk
      pc=pk
c
c     Wave Packet Initialization: Gaussian centered at xk, with momentum pk
c
      alpha=rmass*omega
      do kk=1,nptx
         x=xmin+kk*dx
         chi(kk,1)=((alpha/pi)**0.25)
     1        *exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
         chi0(kk,1)=chi(kk,1)
      end do
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE HAMIL(CRV,x)
c
c     Hamiltonian Matrix
c
      IMPLICIT NONE
      INTEGER NN
      REAL x,VPOT1
      COMPLEX CRV
      PARAMETER(NN=1)
      DIMENSION CRV(NN,NN)
c
```

```fortran
      CALL VA(VPOT1,x)
      CRV(1,1)=VPOT1
C
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE VA(V,x)
C
C     Potential Energy Surface: Morse Potential [Phys. Rev. (1929) 34:57]
C
      implicit none
      REAL V,x,rmass,xk,pk,rk,omega,De,xeq,a
      common /packet/ rmass,xk,pk
      xeq=0.0
      omega=1.0
      De=8.0
      rk=rmass*omega**2
      a=sqrt(rk/(2.0*De))
      V=De*(1.0-exp(-a*(x-xeq)))**2
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetKinProp(dt,tprop)
C
C     Kinetic Energy part of the Trotter Expansion: exp(-i p^2 dt/(2 m))
C
      IMPLICIT NONE
      INTEGER nptx,kx,nx,npts
      REAL xsc,xmin,xmax,propfacx,rmass,xk,pi,alenx,dt,pk
      COMPLEX tprop,eye
      parameter(npts=10,nptx=2**npts)
      DIMENSION tprop(nptx)
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
C
      eye=(0.,1.)
      pi = acos(-1.0)
      alenx=xmax-xmin
      propfacx=-dt/2./rmass*(2.*pi)**2
      do kx=1,nptx
         if(kx.le.(nptx/2+1)) then
            nx=kx-1
         else
            nx=kx-1-nptx
         end if
         xsc=0.
         if(nx.ne.0) xsc=real(nx)/alenx
         tprop(kx)=exp(eye*(propfacx*xsc**2))
       end do
C
      return
```

```fortran
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetPotProp(dt,vprop)
c
c     Potential Energy part of the Trotter Expansion: exp(-i V dt/2)
c
      IMPLICIT NONE
      INTEGER NN,ii,nptx,npts
      REAL xmin,xmax,dx,dt,x,VPOT
      COMPLEX vprop,eye
      parameter(npts=10,nptx=2**npts,NN=1)
      DIMENSION vprop(nptx,NN,NN)
      common /xy/  xmin,xmax
      eye=(0.,1.)
      dx=(xmax-xmin)/real(nptx)
c
      do ii=1,nptx
         x=xmin+ii*dx
         CALL VA(VPOT,x)
         vprop(ii,1,1)=exp(-eye*0.5*dt*VPOT)/sqrt(nptx*1.0)
      END DO
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE energies(energy)
      IMPLICIT NONE
      INTEGER j,NN
      COMPLEX energy,RV,RKE
      PARAMETER (NN=1)
      DIMENSION RV(NN),RKE(NN),energy(NN)
      CALL PE(RV)
      CALL KE(RKE)
      DO j=1,NN
         energy(j)=RV(j)+RKE(j)
      END DO
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE SAVEWF(je2,ndump,dt)
c
c     Dump Time Evolved Wave packet
c
      IMPLICIT NONE
      INTEGER je2,nptx,npts,kk,NN,ncount,ndump,jj
      COMPLEX chi,CRV,energy,psi,Psia
      character*9 B
      REAL V,x1,c1,c2,c1a,x,xmin,xmax,dx,EVALUES,dt
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      DIMENSION CRV(NN,NN),EVALUES(NN)
      DIMENSION psi(NN,NN)
      common /xy/  xmin,xmax
```

319

```fortran
      COMMON / wfunc/ chi(nptx,NN)
      COMMON /ENER/ energy(NN)
c
      IF(je2.EQ.1) CALL energies(energy)
      jj=je2/ndump
      write(B, '(A,i4.4)') 'arch.', jj
      OPEN(1,FILE=B)
      dx=(xmax-xmin)/real(nptx)
      ncount=(je2-1)/ndump
c
c     Save Wave-packet components
c
      do kk=1,nptx
         x=xmin+kk*dx
         c1=chi(kk,1)*conjg(chi(kk,1))
         write(1,33) x,sqrt(c1)+real(energy(1))
      end do
      write(1,33)
      do kk=1,nptx
         x=xmin+kk*dx
         write(1,33) x,real(energy(1))
      end do
      write(1,33)
c
c     Save Adiabatic states
c
      do kk=1,nptx
         x=xmin+kk*dx
         CALL HAMIL(CRV,x)
         write(1,33) x,CRV(1,1)
      end do
      CLOSE(1)
 33   format(6(e13.6,2x))
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE XM(RV)
c
c     Expectation Value of the Position
c
      IMPLICIT NONE
      INTEGER nptx,npts,kk,NN,j
      COMPLEX chi,EYE,RV
      REAL Vpot,omega,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      DIMENSION RV(NN)
      COMMON / wfunc/ chi(nptx,NN)
      common /xy/  xmin,xmax
      common /packet/rmass,xk,pk

      dx=(xmax-xmin)/real(nptx)
```

320

```
      DO j=1,NN
         RV(j)=0.0
         do kk=1,nptx
            x=xmin+kk*dx
            IF(j.EQ.1) CALL VA(Vpot,x)
            RV(j)=RV(j)+chi(kk,j)*x*conjg(chi(kk,j))*dx
         end do
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PE(RV)
c
c     Expectation Value of the Potential Enegy
c
      IMPLICIT NONE
      INTEGER nptx,npts,kk,NN,j
      COMPLEX chi,EYE,RV
      REAL Vpot,omega,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      DIMENSION RV(NN)
      COMMON / wfunc/ chi(nptx,NN)
      common /xy/  xmin,xmax
      common /packet/rmass,xk,pk

      dx=(xmax-xmin)/real(nptx)
      DO j=1,NN
         RV(j)=0.0
         do kk=1,nptx
            x=xmin+kk*dx
            IF(j.EQ.1) CALL VA(Vpot,x)
            RV(j)=RV(j)+chi(kk,j)*Vpot*conjg(chi(kk,j))*dx
         end do
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine KE(RKE)
c
c     Expectation value of the kinetic energy
c
      IMPLICIT NONE
      INTEGER NN,kk,nptx,kx,nx,npts,j
      REAL dp,theta,wm,p,xmin,xmax,rmass,xk,pi,alenx,pk,rm,re,ri,dx
      COMPLEX eye,chi,Psip,chic,RKE
      parameter(npts=10,nptx=2**npts,NN=1)
      DIMENSION chic(nptx),RKE(NN)
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
      COMMON / wfunc/ chi(nptx,NN)
c
```

```
        pi = acos(-1.0)
        dx=(xmax-xmin)/nptx
        dp=2.*pi/(xmax-xmin)
c
        DO j=1,NN
          RKE(j)=0.0
          do kk=1,nptx
            chic(kk)=chi(kk,j)
          end do
          CALL fourn(chic,nptx,1,-1)
          do kx=1,nptx
            if(kx.le.(nptx/2+1)) then
              nx=kx-1
            else
              nx=kx-1-nptx
            end if
            p=0.
            if(nx.ne.0) p = real(nx)*dp
            chic(kx)=p**2/(2.0*rmass)*chic(kx)/nptx
          end do
          CALL fourn(chic,nptx,1,1)
          do kk=1,nptx
            RKE(j)=RKE(j)+conjg(chi(kk,j))*chic(kk)*dx
          end do
        END DO
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine PM(RKE)
c
c     Expectation value of the kinetic energy
c
        IMPLICIT NONE
        INTEGER NN,kk,nptx,kx,nx,npts,j
        REAL dp,theta,wm,p,xmin,xmax,rmass,xk,pi,alenx,pk,rm,re,ri,dx
        COMPLEX eye,chi,Psip,chic,RKE
        parameter(npts=10,nptx=2**npts,NN=1)
        DIMENSION chic(nptx),RKE(NN)
        common /xy/   xmin,xmax
        common /packet/   rmass,xk,pk
        COMMON / wfunc/ chi(nptx,NN)
c
        pi = acos(-1.0)
        dx=(xmax-xmin)/nptx
        dp=2.*pi/(xmax-xmin)
c
        DO j=1,NN
          RKE(j)=0.0
          do kk=1,nptx
            chic(kk)=chi(kk,j)
          end do
```

```fortran
         CALL fourn(chic,nptx,1,-1)
         do kx=1,nptx
            if(kx.le.(nptx/2+1)) then
               nx=kx-1
            else
               nx=kx-1-nptx
            end if
            p=0.
            if(nx.ne.0) p = real(nx)*dp
            chic(kx)=p*chic(kx)/nptx
         end do
         CALL fourn(chic,nptx,1,1)
         do kk=1,nptx
            RKE(j)=RKE(j)+conjg(chi(kk,j))*chic(kk)*dx
         end do
      END DO
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PROPAGATE(vprop,tprop)
c
c     Split Operator Fourier Transform Propagation Method
c     J. Comput. Phys. 47, 412 (1982); J. Chem. Phys. 78, 301 (1983)
c
      IMPLICIT NONE
      INTEGER i,j,NN,ii,nptx,npts
      COMPLEX chi,vprop,chin1,chin2,tprop
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      DIMENSION chin1(nptx),chin2(nptx)
      DIMENSION tprop(nptx),vprop(nptx,NN,NN)
      COMMON / wfunc/ chi(nptx,NN)
c
c     Apply potential energy part of the Trotter Expansion
c
      DO i=1,nptx
         chin1(i)=0.0
         DO j=1,NN
            chin1(i)=chin1(i)+vprop(i,1,j)*chi(i,j)
         END DO
      END DO
c
c     Fourier Transform wave-packet to the momentum representation
c
      CALL fourn(chin1,nptx,1,-1)
c
c     Apply kinetic energy part of the Trotter Expansion
c
      DO i=1,nptx
         chin1(i)=tprop(i)*chin1(i)
      END DO
c
```

```
c     Inverse Fourier Transform wave-packet to the coordinate representation
c
      CALL fourn(chin1,nptx,1,1)
c
c     Apply potential energy part of the Trotter Expansion
c
      DO i=1,nptx
         DO j=1,NN
            chi(i,j)=vprop(i,j,1)*chin1(i)
         END DO
      END DO
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutine for FFT from Numerical Recipes
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE FOURN(DATA,NN,NDIM,ISIGN)
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION NN(NDIM),DATA(*)
      NTOT=1
      DO 11 IDIM=1,NDIM
         NTOT=NTOT*NN(IDIM)
 11   CONTINUE
      NPREV=1
      DO 18 IDIM=1,NDIM
         N=NN(IDIM)
         NREM=NTOT/(N*NPREV)
         IP1=2*NPREV
         IP2=IP1*N
         IP3=IP2*NREM
         I2REV=1
         DO 14 I2=1,IP2,IP1
            IF(I2.LT.I2REV)THEN
               DO 13 I1=I2,I2+IP1-2,2
                  DO 12 I3=I1,IP3,IP2
                     I3REV=I2REV+I3-I2
                     TEMPR=DATA(I3)
                     TEMPI=DATA(I3+1)
                     DATA(I3)=DATA(I3REV)
                     DATA(I3+1)=DATA(I3REV+1)
                     DATA(I3REV)=TEMPR
                     DATA(I3REV+1)=TEMPI
 12               CONTINUE
 13            CONTINUE
            ENDIF
            IBIT=IP2/2
 1          IF ((IBIT.GE.IP1).AND.(I2REV.GT.IBIT)) THEN
               I2REV=I2REV-IBIT
               IBIT=IBIT/2
               GO TO 1
            ENDIF
            I2REV=I2REV+IBIT
```

```
14        CONTINUE
          IFP1=IP1
2         IF(IFP1.LT.IP2)THEN
              IFP2=2*IFP1
              THETA=ISIGN*6.28318530717959D0/(IFP2/IP1)
              WPR=-2.D0*DSIN(0.5D0*THETA)**2
              WPI=DSIN(THETA)
              WR=1.D0
              WI=0.D0
              DO 17 I3=1,IFP1,IP1
                  DO 16 I1=I3,I3+IP1-2,2
                      DO 15 I2=I1,IP3,IFP2
                          K1=I2
                          K2=K1+IFP1
                          TEMPR=SNGL(WR)*DATA(K2)-SNGL(WI)*DATA(K2+1)
                          TEMPI=SNGL(WR)*DATA(K2+1)+SNGL(WI)*DATA(K2)
                          DATA(K2)=DATA(K1)-TEMPR
                          DATA(K2+1)=DATA(K1+1)-TEMPI
                          DATA(K1)=DATA(K1)+TEMPR
                          DATA(K1+1)=DATA(K1+1)+TEMPI
15                    CONTINUE
16                CONTINUE
                  WTEMP=WR
                  WR=WR*WPR-WI*WPI+WR
                  WI=WI*WPR+WTEMP*WPI+WI
17            CONTINUE
              IFP1=IFP2
              GO TO 2
          ENDIF
          NPREV=N*NPREV
18    CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

61.9   Problem 9

Computational Problem 9: Simulate the propagation of a wave-packet with $x_0 = -5.5$ and initial momentum $p_0 = 2$ colliding with a barrier potential $V(x) = 3$, if $\text{abs}(x) < 0.5$, and $V(x) = 0$, otherwise. Hint: In order to avoid artificial recurrences you might need to add an absorbing imaginary potential $V_a(x) = i(\text{abs}(x) - 10)^4$, if $\text{abs}(x) > 10$, and $V_a(x) = 0$, otherwise.

   The output of this program can be generated and visualized as follows. Cut the source code attached below, save it in a file named Problem9.f, compile it by typing

```
gfortran Problem9.f -o Problem9
```

run it by typing

```
./Problem9
```

   The snapshots of the time-dependent wave-packet can be visualized as a movie by typing

```
gnuplot<pp_9
```

where the file named

```
 pp_9
```

has the following lines:
   Download from (http://ursula.chem.yale.edu/~batista/P9/pp_9)

```
set yrange[0:4]
set xrange[-10:10]
set dat sty l
plot "arch.0001" u 1:2 lw 3
pause .1
plot "arch.0002" u 1:2 lw 3
pause .1
plot "arch.0003" u 1:2 lw 3
pause .1
plot "arch.0004" u 1:2 lw 3
pause .1
plot "arch.0005" u 1:2 lw 3
pause .1
plot "arch.0006" u 1:2 lw 3
pause .1
plot "arch.0007" u 1:2 lw 3
pause .1
plot "arch.0008" u 1:2 lw 3
pause .1
plot "arch.0009" u 1:2 lw 3
pause .1
plot "arch.0010" u 1:2 lw 3
pause .1
plot "arch.0011" u 1:2 lw 3
```

```
pause .1
plot "arch.0012" u 1:2 lw 3
pause .1
plot "arch.0013" u 1:2 lw 3
pause .1
plot "arch.0014" u 1:2 lw 3
pause .1
plot "arch.0015" u 1:2 lw 3
pause .1
plot "arch.0016" u 1:2 lw 3
pause .1
plot "arch.0017" u 1:2 lw 3
pause .1
plot "arch.0018" u 1:2 lw 3
pause .1
plot "arch.0019" u 1:2 lw 3
pause .1
plot "arch.0020" u 1:2 lw 3
pause .1
plot "arch.0021" u 1:2 lw 3
pause .1
plot "arch.0022" u 1:2 lw 3
pause .1
plot "arch.0023" u 1:2 lw 3
pause .1
plot "arch.0024" u 1:2 lw 3
pause .1
plot "arch.0025" u 1:2 lw 3
pause .1
plot "arch.0026" u 1:2 lw 3
pause .1
plot "arch.0027" u 1:2 lw 3
pause .1
plot "arch.0028" u 1:2 lw 3
pause .1
plot "arch.0029" u 1:2 lw 3
pause .1
plot "arch.0030" u 1:2 lw 3
pause .1
plot "arch.0031" u 1:2 lw 3
pause .1
plot "arch.0032" u 1:2 lw 3
pause .1
plot "arch.0033" u 1:2 lw 3
pause .1
plot "arch.0034" u 1:2 lw 3
pause .1
plot "arch.0035" u 1:2 lw 3
pause .1
plot "arch.0036" u 1:2 lw 3
pause .1
```

```
plot "arch.0037" u 1:2 lw 3
pause .1
plot "arch.0038" u 1:2 lw 3
pause .1
plot "arch.0039" u 1:2 lw 3
pause .1
plot  "arch.0040" u 1:2 lw 3
pause .1
plot "arch.0041" u 1:2 lw 3
pause .1
plot "arch.0042" u 1:2 lw 3
pause .1
plot "arch.0043" u 1:2 lw 3
pause .1
plot "arch.0044" u 1:2 lw 3
pause .1
plot "arch.0045" u 1:2 lw 3
pause .1
plot "arch.0046" u 1:2 lw 3
pause .1
plot "arch.0047" u 1:2 lw 3
pause .1
plot "arch.0048" u 1:2 lw 3
pause .1
plot "arch.0049" u 1:2 lw 3
pause .1
plot "arch.0050" u 1:2 lw 3
pause .1
plot "arch.0051" u 1:2 lw 3
pause .1
plot "arch.0052" u 1:2 lw 3
pause .1
plot "arch.0053" u 1:2 lw 3
pause .1
plot "arch.0054" u 1:2 lw 3
pause .1
plot "arch.0055" u 1:2 lw 3
pause .1
plot "arch.0056" u 1:2 lw 3
pause .1
plot "arch.0057" u 1:2 lw 3
pause .1
plot "arch.0058" u 1:2 lw 3
pause .1
plot "arch.0059" u 1:2 lw 3
pause .1
plot "arch.0060" u 1:2 lw 3
pause .1
plot "arch.0061" u 1:2 lw 3
pause .1
plot "arch.0062" u 1:2 lw 3
```

```
pause .1
plot "arch.0063" u 1:2 lw 3
pause .1
plot "arch.0064" u 1:2 lw 3
pause .1
plot "arch.0065" u 1:2 lw 3
pause .1
plot "arch.0066" u 1:2 lw 3
pause .1
plot "arch.0067" u 1:2 lw 3
pause .1
plot "arch.0068" u 1:2 lw 3
pause .1
plot "arch.0069" u 1:2 lw 3
pause .1
plot "arch.0070" u 1:2 lw 3
pause .1
plot "arch.0071" u 1:2 lw 3
pause .1
plot "arch.0072" u 1:2 lw 3
pause .1
plot "arch.0073" u 1:2 lw 3
pause .1
plot "arch.0074" u 1:2 lw 3
pause .1
plot "arch.0075" u 1:2 lw 3
pause .1
plot "arch.0076" u 1:2 lw 3
pause .1
plot "arch.0077" u 1:2 lw 3
pause .1
plot "arch.0078" u 1:2 lw 3
pause .1
plot "arch.0079" u 1:2 lw 3
pause .1
plot  "arch.0080" u 1:2 lw 3
pause .1
plot "arch.0081" u 1:2 lw 3
pause .1
plot "arch.0082" u 1:2 lw 3
pause .1
plot "arch.0083" u 1:2 lw 3
pause .1
plot "arch.0084" u 1:2 lw 3
pause .1
plot "arch.0085" u 1:2 lw 3
pause .1
plot "arch.0086" u 1:2 lw 3
pause .1
plot "arch.0087" u 1:2 lw 3
pause .1
```

```
plot "arch.0088" u 1:2 lw 3
pause .1
plot "arch.0089" u 1:2 lw 3
pause .1
plot  "arch.0090" u 1:2 lw 3
pause .1
plot "arch.0091" u 1:2 lw 3
pause .1
plot "arch.0092" u 1:2 lw 3
pause .1
plot "arch.0093" u 1:2 lw 3
pause .1
plot "arch.0094" u 1:2 lw 3
pause .1
plot "arch.0095" u 1:2 lw 3
pause .1
plot "arch.0096" u 1:2 lw 3
pause .1
plot "arch.0097" u 1:2 lw 3
pause .1
plot "arch.0098" u 1:2 lw 3
pause .1
plot "arch.0099" u 1:2 lw 3
pause .1
```

```
      PROGRAM Problem9
c
c     1-D wave packet propagation of tunneling through a barrier
c
      IMPLICIT NONE
      INTEGER NN,npts,nptx,ndump
      INTEGER istep,nstep,jj
      REAL dt,xc,pc
      COMPLEX vprop,tprop,x_mean,p_mean
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      DIMENSION vprop(nptx,NN,NN),tprop(nptx)
      DIMENSION x_mean(NN),p_mean(NN)
      COMMON /class/ xc,pc
c
      CALL ReadParam(nstep,ndump,dt)
      call Initialize()
      CALL SetKinProp(dt,tprop)
      CALL SetPotProp(dt,vprop)
      DO istep=1,nstep+1
         IF(mod(istep-1,10).EQ.0)
    1        PRINT *, "Step=", istep-1,", Final step=", nstep
         IF(istep.GE.1) CALL PROPAGATE(vprop,tprop)
         IF(mod((istep-1),ndump).EQ.0) THEN
            CALL SAVEWF(istep,ndump,dt)
         END IF
      END DO
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine ReadParam(nstep,ndump,dt)
c
c     Parameters defining the grid (xmin, xmax), integration time step (dt),
c     rmass (rmass), initial position (xk), initial momentum (pk),
c     number of propagation steps (nstep), and how often to save a pic (ndump)
c
      IMPLICIT NONE
      INTEGER ntype,nstep,nrpt,ireport,ndump,nlit
      REAL xmin,xmax,pk,rmass,xk,dt
      common /packet/  rmass,xk,pk
      common /xy/  xmin,xmax
c
      xmin=-13.0
      xmax=13.0
      dt=0.1
      rmass=1.0
      xk=-4.5
      pk=1.
      nstep=100
      ndump=1
c
```

```fortran
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Initialize()

      IMPLICIT NONE
      INTEGER NN,nptx,npts,kk
      COMPLEX chi0,chi,EYE,CRV
      REAL xc,pc,omega,xk2,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha,alpha2
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      DIMENSION CRV(NN,NN)
      common /xy/   xmin,xmax
      common /packet/  rmass,xk,pk
      COMMON / wfunc/ chi(nptx,NN)
      COMMON / iwfunc/ chi0(nptx,NN)
      COMMON /class/ xc,pc

      EYE=(0.0,1.0)
      pi= acos(-1.0)
      omega=1.
      dx=(xmax-xmin)/real(nptx)
      xc=xk
      pc=pk
c
c     Wave Packet Initialization: Gaussian centered at xk, with momentum pk
c
      alpha=rmass*omega
      do kk=1,nptx
         x=xmin+kk*dx
         chi(kk,1)=((alpha/pi)**0.25)
     1        *exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
         chi0(kk,1)=chi(kk,1)
      end do
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE HAMIL(CRV,x)
c
c     Hamiltonian Matrix
c
      IMPLICIT NONE
      INTEGER NN
      REAL x,VPOT1
      COMPLEX CRV
      PARAMETER(NN=1)
      DIMENSION CRV(NN,NN)
c
      CALL VA(VPOT1,x)
      CRV(1,1)=VPOT1
c
      RETURN
```

332

```
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE VA(V,x)
c
c     Potential Energy Surface: Barrier
c
      implicit none
      REAL V,x,rmass,xk,pk,rk,omega
      common /packet/ rmass,xk,pk
      V=0.0
      IF(abs(x).LE.(.5)) V=3.
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetKinProp(dt,tprop)
c
c     Kinetic Energy part of the Trotter Expansion: exp(-i p^2 dt/(2 m))
c
      IMPLICIT NONE
      INTEGER nptx,kx,nx,npts
      REAL xsc,xmin,xmax,propfacx,rmass,xk,pi,alenx,dt,pk
      COMPLEX tprop,eye
      parameter(npts=10,nptx=2**npts)
      DIMENSION tprop(nptx)
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
c
      eye=(0.,1.)
      pi = acos(-1.0)
      alenx=xmax-xmin
      propfacx=-dt/2./rmass*(2.*pi)**2
      do kx=1,nptx
         if(kx.le.(nptx/2+1)) then
            nx=kx-1
         else
            nx=kx-1-nptx
         end if
         xsc=0.
         if(nx.ne.0) xsc=real(nx)/alenx
         tprop(kx)=exp(eye*(propfacx*xsc**2))
       end do
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetPotProp(dt,vprop)
c
c     Potential Energy part of the Trotter Expansion: exp(-i V dt/2)
c
      IMPLICIT NONE
      INTEGER NN,ii,nptx,npts
```

```fortran
      REAL xmin,xmax,dx,dt,x,VPOT,xa
      COMPLEX vprop,eye
      parameter(npts=10,nptx=2**npts,NN=1,xa=10.)
      DIMENSION vprop(nptx,NN,NN)
      common /xy/  xmin,xmax
      eye=(0.,1.)
      dx=(xmax-xmin)/real(nptx)
c
      do ii=1,nptx
         x=xmin+ii*dx
         CALL VA(VPOT,x)
         vprop(ii,1,1)=exp(-eye*0.5*dt*VPOT)/sqrt(nptx*1.0)
         IF(abs(x).GT.(xa))
     1       vprop(ii,1,1)=vprop(ii,1,1)*exp(-(abs(x)-xa)**4)
      END DO
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE energies(energy)
      IMPLICIT NONE
      INTEGER j,NN
      COMPLEX energy,RV,RKE
      PARAMETER (NN=1)
      DIMENSION RV(NN),RKE(NN),energy(NN)
      CALL PE(RV)
      CALL KE(RKE)
      DO j=1,NN
         energy(j)=RV(j)+RKE(j)
      END DO
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE SAVEWF(je2,ndump,dt)
c
c     Dump Time Evolved Wave packet
c
      IMPLICIT NONE
      INTEGER je2,nptx,npts,kk,NN,ncount,ndump,jj
      COMPLEX chi,CRV,energy,psi,Psia
      character*9 B
      REAL V,x1,c1,c2,c1a,x,xmin,xmax,dx,EVALUES,dt
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      DIMENSION CRV(NN,NN),EVALUES(NN)
      DIMENSION psi(NN,NN)
      common /xy/  xmin,xmax
      COMMON / wfunc/ chi(nptx,NN)
      COMMON /ENER/ energy(NN)
c
      IF(je2.EQ.1) CALL energies(energy)
      jj=je2/ndump
      write(B, '(A,i4.4)') 'arch.', jj
```

334

```fortran
      OPEN(1,FILE=B)
      dx=(xmax-xmin)/real(nptx)
      ncount=(je2-1)/ndump
c
c     Save Wave-packet components
c
      do kk=1,nptx
         x=xmin+kk*dx
         c1=chi(kk,1)*conjg(chi(kk,1))
         write(1,33) x,sqrt(c1)+real(energy(1))
      end do
      write(1,33)
      do kk=1,nptx
         x=xmin+kk*dx
         write(1,33) x
     1          ,real(chi(kk,1))+real(energy(1))
      end do
      write(1,33)
c
c     Save Adiabatic states
c
      do kk=1,nptx
         x=xmin+kk*dx
         CALL HAMIL(CRV,x)
         write(1,33) x,CRV(1,1)
      end do
      CLOSE(1)
 33   format(6(e13.6,2x))
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PE(RV)
c
c     Expectation Value of the Potential Enegy
c
      IMPLICIT NONE
      INTEGER nptx,npts,kk,NN,j
      COMPLEX chi,EYE,RV
      REAL Vpot,omega,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      DIMENSION RV(NN)
      COMMON / wfunc/ chi(nptx,NN)
      common /xy/   xmin,xmax
      common /packet/rmass,xk,pk

      dx=(xmax-xmin)/real(nptx)
      DO j=1,NN
         RV(j)=0.0
         do kk=1,nptx
            x=xmin+kk*dx
            IF(j.EQ.1) CALL VA(Vpot,x)
```

```fortran
            RV(j)=RV(j)+chi(kk,j)*Vpot*conjg(chi(kk,j))*dx
          end do
        END DO
        RETURN
        END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine KE(RKE)
c
c       Expectation value of the kinetic energy
c
        IMPLICIT NONE
        INTEGER NN,kk,nptx,kx,nx,npts,j
        REAL dp,theta,wm,p,xmin,xmax,rmass,xk,pi,alenx,pk,rm,re,ri,dx
        COMPLEX eye,chi,Psip,chic,RKE
        parameter(npts=10,nptx=2**npts,NN=1)
        DIMENSION chic(nptx),RKE(NN)
        common /xy/   xmin,xmax
        common /packet/   rmass,xk,pk
        COMMON / wfunc/ chi(nptx,NN)
c
        pi = acos(-1.0)
        dx=(xmax-xmin)/nptx
        dp=2.*pi/(xmax-xmin)
c
        DO j=1,NN
           RKE(j)=0.0
           do kk=1,nptx
              chic(kk)=chi(kk,j)
           end do
           CALL fourn(chic,nptx,1,-1)
           do kx=1,nptx
              if(kx.le.(nptx/2+1)) then
                 nx=kx-1
              else
                 nx=kx-1-nptx
              end if
              p=0.
              if(nx.ne.0) p = real(nx)*dp
              chic(kx)=p**2/(2.0*rmass)*chic(kx)/nptx
           end do
           CALL fourn(chic,nptx,1,1)
           do kk=1,nptx
              RKE(j)=RKE(j)+conjg(chi(kk,j))*chic(kk)*dx
           end do
        END DO
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        SUBROUTINE PROPAGATE(vprop,tprop)
c
c       Split Operator Fourier Transform Propagation Method
```

```
c      J. Comput. Phys. 47, 412 (1982); J. Chem. Phys. 78, 301 (1983)
c
       IMPLICIT NONE
       INTEGER i,j,NN,ii,nptx,npts
       COMPLEX chi,vprop,chin1,chin2,tprop
       PARAMETER(npts=10,nptx=2**npts,NN=1)
       DIMENSION chin1(nptx),chin2(nptx)
       DIMENSION tprop(nptx),vprop(nptx,NN,NN)
       COMMON / wfunc/ chi(nptx,NN)
c
c      Apply potential energy part of the Trotter Expansion
c
       DO i=1,nptx
          chin1(i)=0.0
          DO j=1,NN
             chin1(i)=chin1(i)+vprop(i,1,j)*chi(i,j)
          END DO
       END DO
c
c      Fourier Transform wave-packet to the momentum representation
c
       CALL fourn(chin1,nptx,1,-1)
c
c      Apply kinetic energy part of the Trotter Expansion
c
       DO i=1,nptx
          chin1(i)=tprop(i)*chin1(i)
       END DO
c
c      Inverse Fourier Transform wave-packet to the coordinate representation
c
       CALL fourn(chin1,nptx,1,1)
c
c      Apply potential energy part of the Trotter Expansion
c
       DO i=1,nptx
          DO j=1,NN
             chi(i,j)=vprop(i,j,1)*chin1(i)
          END DO
       END DO
       END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c      Subroutine for FFT from Numerical Recipes
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       SUBROUTINE FOURN(DATA,NN,NDIM,ISIGN)
       REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
       DIMENSION NN(NDIM),DATA(*)
       NTOT=1
       DO 11 IDIM=1,NDIM
          NTOT=NTOT*NN(IDIM)
  11   CONTINUE
```

```fortran
      NPREV=1
      DO 18 IDIM=1,NDIM
          N=NN(IDIM)
          NREM=NTOT/(N*NPREV)
          IP1=2*NPREV
          IP2=IP1*N
          IP3=IP2*NREM
          I2REV=1
          DO 14 I2=1,IP2,IP1
              IF(I2.LT.I2REV)THEN
                  DO 13 I1=I2,I2+IP1-2,2
                      DO 12 I3=I1,IP3,IP2
                          I3REV=I2REV+I3-I2
                          TEMPR=DATA(I3)
                          TEMPI=DATA(I3+1)
                          DATA(I3)=DATA(I3REV)
                          DATA(I3+1)=DATA(I3REV+1)
                          DATA(I3REV)=TEMPR
                          DATA(I3REV+1)=TEMPI
12                    CONTINUE
13                CONTINUE
              ENDIF
              IBIT=IP2/2
1             IF ((IBIT.GE.IP1).AND.(I2REV.GT.IBIT)) THEN
                  I2REV=I2REV-IBIT
                  IBIT=IBIT/2
                  GO TO 1
              ENDIF
              I2REV=I2REV+IBIT
14        CONTINUE
          IFP1=IP1
2         IF(IFP1.LT.IP2)THEN
              IFP2=2*IFP1
              THETA=ISIGN*6.28318530717959D0/(IFP2/IP1)
              WPR=-2.D0*DSIN(0.5D0*THETA)**2
              WPI=DSIN(THETA)
              WR=1.D0
              WI=0.D0
              DO 17 I3=1,IFP1,IP1
                  DO 16 I1=I3,I3+IP1-2,2
                      DO 15 I2=I1,IP3,IFP2
                          K1=I2
                          K2=K1+IFP1
                          TEMPR=SNGL(WR)*DATA(K2)-SNGL(WI)*DATA(K2+1)
                          TEMPI=SNGL(WR)*DATA(K2+1)+SNGL(WI)*DATA(K2)
                          DATA(K2)=DATA(K1)-TEMPR
                          DATA(K2+1)=DATA(K1+1)-TEMPI
                          DATA(K1)=DATA(K1)+TEMPR
                          DATA(K1+1)=DATA(K1+1)+TEMPI
15                    CONTINUE
16                CONTINUE
```

```fortran
            WTEMP=WR
            WR=WR*WPR-WI*WPI+WR
            WI=WI*WPR+WTEMP*WPI+WI
17        CONTINUE
          IFP1=IFP2
          GO TO 2
        ENDIF
        NPREV=N*NPREV
18   CONTINUE
     RETURN
     END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

61.10   Problem 10

Problem 10: (a) Derive Eq. (36) by considering that,

$$e^{-i\mathbf{V}_c 2\tau} = \mathbf{D}^\dagger \begin{pmatrix} e^{iV_c(\mathbf{x})2\tau} & 0 \\ 0 & e^{-iV_c(\mathbf{x})2\tau} \end{pmatrix} \mathbf{D}, \tag{1194}$$

with

$$\mathbf{D} = \mathbf{D}^\dagger \equiv \begin{pmatrix} -1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}, \tag{1195}$$

since

$$e^{-i\mathbf{V}_c 2\tau} = \mathbf{1} + (-i\mathbf{V}_c 2\tau) + \frac{1}{2!}(-i\mathbf{V}_c 2\tau)^2 + ..., \tag{1196}$$

and

$$\mathbf{V}_c \equiv \begin{pmatrix} 0 & V_c(\mathbf{x}) \\ V_c(\mathbf{x}) & 0 \end{pmatrix} = \mathbf{D}^\dagger \begin{pmatrix} -V_c(\mathbf{x}) & 0 \\ 0 & V_c(\mathbf{x}) \end{pmatrix} \mathbf{D}, \tag{1197}$$

with $\mathbf{DD}^\dagger = 1$.

In order to derive Eq. (28) we need to prove the following equation:

$$e^{-iV_0\tau}e^{-iV_c 2\tau}e^{-iV_0\tau} = \begin{pmatrix} e^{-iV_1(\mathbf{x})2\tau}\cos(2V_c(\mathbf{x})\tau) & -i\sin(2V_c(\mathbf{x})\tau)\,e^{-i(\hat{V}_1(\mathbf{x})+\hat{V}_2(\mathbf{x}))\tau} \\ -i\sin(2V_c(\mathbf{x})\tau)\,e^{-i(V_1(\mathbf{x})+\hat{V}_2(\mathbf{x}))\tau} & \cos(2V_c(\mathbf{x})\tau)\,e^{-iV_2(\mathbf{x})2\tau} \end{pmatrix}. \tag{1198}$$

where

$$e^{-iV_0\tau} = e^{-i\begin{pmatrix} V_1(\mathbf{x}) & 0 \\ 0 & V_2(\mathbf{x}) \end{pmatrix}\tau}. \tag{1199}$$

Expanding the exponential on the r.h.s. of Eq. (1199) gives

$$e^{-i\tau\begin{pmatrix} V_1(\mathbf{x}) & 0 \\ 0 & V_2(\mathbf{x}) \end{pmatrix}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} -i\tau V_1(\mathbf{x}) & 0 \\ 0 & -i\tau V_2(\mathbf{x}) \end{pmatrix} + \begin{pmatrix} \frac{1}{2!}V_1(\mathbf{x})^2(-i\tau)^2 & 0 \\ 0 & \frac{1}{2!}V_2(\mathbf{x})^2(-i\tau)^2 \end{pmatrix} + ... \tag{1200}$$

Therefore,

$$e^{-i\tau\begin{pmatrix} V_1(\mathbf{x}) & 0 \\ 0 & V_2(\mathbf{x}) \end{pmatrix}} = \begin{pmatrix} e^{-iV_1(\mathbf{x})\tau} & 0 \\ 0 & e^{-iV_2(\mathbf{x})\tau} \end{pmatrix}. \tag{1201}$$

In addition, according to Eq. (30),

$$e^{-i\mathbf{V}_c 2\tau} = \mathbf{D}^\dagger \begin{pmatrix} e^{iV_c(\mathbf{x})2\tau} & 0 \\ 0 & e^{-iV_c(\mathbf{x})2\tau} \end{pmatrix} \mathbf{D}, \tag{1202}$$

with
$$\mathbf{D} = \mathbf{D}^\dagger \equiv \begin{pmatrix} -1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}, \tag{1203}$$

Therefore,

$$e^{-iV_0\tau}e^{-iV_c2\tau}e^{-iV_0\tau} = \begin{pmatrix} e^{-iV_1(\mathbf{x})\tau} & 0 \\ 0 & e^{-iV_2(\mathbf{x})\tau} \end{pmatrix} \mathbf{D}^\dagger \begin{pmatrix} e^{iV_c(\mathbf{x})2\tau} & 0 \\ 0 & e^{-iV_c(\mathbf{x})2\tau} \end{pmatrix} \mathbf{D} \begin{pmatrix} e^{-iV_1(\mathbf{x})\tau} & 0 \\ 0 & e^{-iV_2(\mathbf{x})\tau} \end{pmatrix}.$$
$$\tag{1204}$$

The multiplication of the five matrices on the r.h.s. of Eq. (1204) gives the matrix on the r.h.s. of Eq.(1198).

61.11 Problem 11

Problem 11: Derive Eq. (35) by writing the matrix V in the basis of adiabatic eigenstates

$$
\begin{aligned}
\phi_1(x) &= L_{11}(x)|1\rangle + L_{21}(x)|2\rangle, \\
\phi_2(x) &= L_{12}(x)|1\rangle + L_{22}(x)|2\rangle,
\end{aligned}
\tag{1205}
$$

with eigenvalues $E_1(x)$ and $E_2(x)$, respectively. Then, using the expansion

$$
e^{-i\mathbf{V}2\tau} = \mathbf{1} + (-i\mathbf{V}2\tau) + \frac{1}{2!}(-i\mathbf{V}2\tau)^2 + ...,
\tag{1206}
$$

show that in the adiabatic representation

$$
e^{-i\mathbf{V}2\tau} = \begin{pmatrix} e^{-iE_1(x)2\tau} & 0 \\ 0 & e^{-iE_2(x)2\tau} \end{pmatrix}.
\tag{1207}
$$

Finally, show that the diagonal matrix introduced by Eq. (1207) can be rotated to the representation of diabatic states $|1\rangle$, $|2\rangle$ according to the similarity transformation

$$
\mathbf{L}^{-1} \begin{pmatrix} e^{-iE_1(x)\tau} & 0 \\ 0 & e^{-iE_2(x)\tau} \end{pmatrix} \mathbf{L}.
\tag{1208}
$$

According to the definition of the eigenstates of the potential energy matrix, given by Eq. (34),

$$
\begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} = \begin{pmatrix} E_1 & 0 \\ 0 & E_1 \end{pmatrix} \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix},
\tag{1209}
$$

and

$$
\begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \begin{pmatrix} L_{12} \\ L_{22} \end{pmatrix} = \begin{pmatrix} E_2 & 0 \\ 0 & E_2 \end{pmatrix} \begin{pmatrix} L_{12} \\ L_{22} \end{pmatrix}.
\tag{1210}
$$

Therefore,

$$
\begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \begin{pmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix},
\tag{1211}
$$

and

$$
\begin{pmatrix} L_{11} & L_{21} \\ L_{12} & L_{22} \end{pmatrix} \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \begin{pmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{pmatrix} = \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix},
\tag{1212}
$$

or

$$
\begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix} \begin{pmatrix} L_{11} & L_{21} \\ L_{12} & L_{22} \end{pmatrix}.
\tag{1213}
$$

Therefore, defining

$$\mathbf{L} = \begin{pmatrix} L_{11} & L_{21} \\ L_{12} & L_{22} \end{pmatrix}, \tag{1214}$$

gives

$$\begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} = \mathbf{L}^{-1} \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix} \mathbf{L}. \tag{1215}$$

Exponentiating both sides of Eq. (1215), gives

$$e^{-i\tau \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix}} = e^{-i\tau \mathbf{L}^{-1} \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix} \mathbf{L}}. \tag{1216}$$

Expanding the r.h.s. of Eq. (1216) gives,

$$e^{-i\tau \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \mathbf{L}^{-1} \begin{pmatrix} -i\tau E_1 & 0 \\ 0 & -i\tau E_2 \end{pmatrix} \mathbf{L} + \mathbf{L}^{-1} \begin{pmatrix} \frac{1}{2!} E_1^2(-i\tau)^2 & 0 \\ 0 & \frac{1}{2!} E_2^2(-i\tau)^2 \end{pmatrix} \mathbf{L} + ...,$$
$$\tag{1217}$$

since $\mathbf{L}^{-1}\mathbf{L} = 1$. Therefore,

$$e^{-i\tau \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix}} = \mathbf{L}^{-1} \begin{pmatrix} e^{-iE_1(\mathbf{x})\tau} & 0 \\ 0 & e^{-iE_2(\mathbf{x})\tau} \end{pmatrix} \mathbf{L}. \tag{1218}$$

61.12   Problem 12

Computational Problem 12: (a) Write a computer program to implement the SOFT approach described in this section, where step [II] is numerically computed according to Eq. (35). Propagate $|\Psi(\mathbf{x};t)\rangle = \varphi_1(\mathbf{x};t)|1\rangle + \varphi_2(\mathbf{x};t)|2\rangle$, where $\varphi_1(\mathbf{x};0) = \varphi_1(\mathbf{x};0) = \Psi_0(x)$ and $\Psi_0(x)$ as defined in Eq. (11). Use $x_0 = -2.2$, $p_0 = 0$, $m = 1$, $\omega = 1$ and two coupled potential energy surfaces described by the potential energy matrix

$$V = \begin{pmatrix} V_1(x) & \delta \\ \delta & V_2(x) \end{pmatrix}, \tag{1219}$$

where $\delta = 0.3$, $V_1(x) = m\omega^2(x - \bar{x})^2/2$ and $V_2(x) = -x^2/2 + x^4/22$; (b) Propagate $\Psi(\mathbf{x};t)$ according to the potential energy matrix introduced by Eq. (1219), with $\delta = 0$ and compare your results with those obtained in item (a).

The output of this program can be generated and visualized as follows. Cut the source code attached below, save it in a file named Problem12.f, compile it by typing

```
gfortran Problem12.f -o Problem12
```

run it by typing

```
./Problem12
```

That will produce the output for item (a). In order to obtain the output for item (b), modify subroutine Hamil, so that CRV(1,2)=0.0 and CRV(2,1)=0.0, recompile and run.

The snapshots of the time-dependent wave-packet can be visualized as a movie by typing

```
gnuplot<pp_12
```

where the file named

```
pp_12
```

has the following lines:
    Download from (http://ursula.chem.yale.edu/~batista/P12/P12_c/pp_12)

```
set yrange[-2:5]
set dat sty l
plot "arch.0001" u 1:2 lw 3
pause .1
plot "arch.0002" u 1:2 lw 3
pause .1
plot "arch.0003" u 1:2 lw 3
pause .1
plot "arch.0004" u 1:2 lw 3
pause .1
plot "arch.0005" u 1:2 lw 3
pause .1
```

```
plot "arch.0006" u 1:2 lw 3
pause .1
plot "arch.0007" u 1:2 lw 3
pause .1
plot "arch.0008" u 1:2 lw 3
pause .1
plot "arch.0009" u 1:2 lw 3
pause .1
plot "arch.0010" u 1:2 lw 3
pause .1
plot "arch.0011" u 1:2 lw 3
pause .1
plot "arch.0012" u 1:2 lw 3
pause .1
plot "arch.0013" u 1:2 lw 3
pause .1
plot "arch.0014" u 1:2 lw 3
pause .1
plot "arch.0015" u 1:2 lw 3
pause .1
plot "arch.0016" u 1:2 lw 3
pause .1
plot "arch.0017" u 1:2 lw 3
pause .1
plot "arch.0018" u 1:2 lw 3
pause .1
plot "arch.0019" u 1:2 lw 3
pause .1
plot "arch.0020" u 1:2 lw 3
pause .1
plot "arch.0021" u 1:2 lw 3
pause .1
plot "arch.0022" u 1:2 lw 3
pause .1
plot "arch.0023" u 1:2 lw 3
pause .1
plot "arch.0024" u 1:2 lw 3
pause .1
plot "arch.0025" u 1:2 lw 3
pause .1
plot "arch.0026" u 1:2 lw 3
pause .1
plot "arch.0027" u 1:2 lw 3
pause .1
plot "arch.0028" u 1:2 lw 3
pause .1
plot "arch.0029" u 1:2 lw 3
pause .1
plot "arch.0030" u 1:2 lw 3
pause .1
```

```fortran
      PROGRAM Problem12
c
c     1-D nonadiabatic wave-packet propagation
c
      IMPLICIT NONE
      INTEGER NN,npts,nptx,ndump
      INTEGER istep,nstep
      REAL dt
      COMPLEX vprop,tprop
      PARAMETER(npts=9,nptx=2**npts,NN=2)
      DIMENSION vprop(nptx,NN,NN),tprop(nptx)
c
      CALL ReadParam(nstep,ndump,dt)
      call Initialize()
      CALL SetKinProp(dt,tprop)
      CALL SetPotProp(dt,vprop)
      DO istep=1,nstep+1
         IF(mod(istep-1,10).EQ.0)
    1         PRINT *, "Step=", istep-1,", Final step=", nstep
         IF(istep.GT.1) CALL PROPAGATE(vprop,tprop)
         IF(mod((istep-1),ndump).EQ.0) THEN
            CALL SAVEWF(istep,ndump,dt)
         END IF
      END DO
 22   FORMAT(6(e13.6,2x))
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE energies(energy)
      IMPLICIT NONE
      INTEGER j,NN
      COMPLEX energy,RV,RKE
      PARAMETER (NN=2)
      DIMENSION RV(NN),RKE(NN),energy(NN)
      CALL PE(RV)
      CALL KE(RKE)
      DO j=1,NN
         energy(j)=RV(j)+RKE(j)
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine ReadParam(nstep,ndump,dt)
c
c     Parameters defining the grid (xmin, xmax), integration time step (dt),
c     mass (amassx), initial position (xk), initial momentum (pk),
c     number of propagation steps (nstep), and how often to save a pic (ndump)
c
      IMPLICIT NONE
```

346

```fortran
      INTEGER ntype,nstep,nrpt,ireport,ndump,nlit
      REAL xmin,xmax,pk,amassx,xk,dt
      common /packet/  amassx,xk,pk
      common /xy/  xmin,xmax
C
      xmin=-6.0
      xmax=6.0
      dt=0.2
      amassx=1.0
      xk=-2.2
      pk=0.
      nstep=30
      ndump=1
C
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Initialize()

      IMPLICIT NONE
      INTEGER NN,nptx,npts,kk
      COMPLEX chi0,chi,EYE,CRV
      REAL omega,xk2,xmin,xmax,dx,pi,amassx,xk,pk,x,alpha,alpha2
      PARAMETER(npts=9,nptx=2**npts,NN=2)
      DIMENSION CRV(NN,NN)
      common /xy/  xmin,xmax
      common /packet/  amassx,xk,pk
      COMMON / wfunc/ chi(nptx,NN)
      COMMON / iwfunc/ chi0(nptx,NN)

      EYE=(0.0,1.0)
      pi= acos(-1.0)
      omega=1.
      dx=(xmax-xmin)/real(nptx)
C
C     Wave Packet Initialization: Gaussian centered at xk, with momentum pk
C
      alpha=amassx*omega
      do kk=1,nptx
         x=xmin+kk*dx
         chi(kk,1)=((alpha/pi)**0.25)/sqrt(2.)
     1        *exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
         chi(kk,2)=chi(kk,1)*0.
C
         chi0(kk,1)=chi(kk,1)
         chi0(kk,2)=chi(kk,2)
      end do
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE SAVEWF(je2,ndump,dt)
```

```fortran
c
c      Dump Time Evolved Wave packet
c
       IMPLICIT NONE
       INTEGER je2,nptx,npts,kk,NN,ncount,ndump,jj
       COMPLEX chi,CRV,energy,psi,Psia
       character*9 B
       REAL V,x1,c1,c2,c1a,x,xmin,xmax,dx,EVALUES,dt
       PARAMETER(npts=9,nptx=2**npts,NN=2)
       DIMENSION CRV(NN,NN),EVALUES(NN)
       DIMENSION psi(NN,NN)
       common /xy/  xmin,xmax
       COMMON / wfunc/ chi(nptx,NN)
       COMMON /ENER/ energy(NN)
c
       IF(je2.EQ.1) CALL energies(energy)
       jj=je2/ndump
       write(B, '(A,i4.4)') 'arch.', jj
       OPEN(1,FILE=B)
       dx=(xmax-xmin)/real(nptx)
       ncount=(je2-1)/ndump
c
c      Save Wave-packet components
c
       do kk=1,nptx
          x=xmin+kk*dx
          c1=chi(kk,1)*conjg(chi(kk,1))
          c2=chi(kk,2)*conjg(chi(kk,2))
          write(1,33) x,sqrt(c1)+real(energy(1))
       end do
       write(1,33)

       do kk=1,nptx
          x=xmin+kk*dx
          c2=chi(kk,2)*conjg(chi(kk,2))
           write(1,33) x,sqrt(c2)+real(energy(2))
       end do
       write(1,33)

       do kk=1,nptx
          x=xmin+kk*dx
           write(1,33) x,real(energy(2))
       end do
       write(1,33)

       do kk=1,nptx
          x=xmin+kk*dx
           write(1,33) x,real(energy(1))
       end do
       write(1,33)
c
```

```
c      Save Adiabatic states
c
       do kk=1,nptx
          x=xmin+kk*dx
          CALL HAMIL(CRV,x)
          CALL SCHROC1(CRV,psi,EVALUES)
          write(1,33) x,EVALUES(1)
       end do
       write(1,33)
       do kk=1,nptx
          x=xmin+kk*dx
          CALL HAMIL(CRV,x)
          CALL SCHROC1(CRV,psi,EVALUES)
          write(1,33) x,EVALUES(2)
       end do
       CLOSE(1)
 33    format(6(e13.6,2x))
       RETURN
       END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       subroutine SetKinProp(dt,tprop)
c
c      Kinetic Energy part of the Trotter Expansion: exp(-i p^2 dt/(2 m))
c
       IMPLICIT NONE
       INTEGER nptx,kx,nx,npts,NN
       REAL xsc,xmin,xmax,propfacx,amassx,xk,pi,alenx,dt,pk
       COMPLEX tprop,eye
       parameter(npts=9,nptx=2**npts,NN=2)
       DIMENSION tprop(nptx)
       common /xy/   xmin,xmax
       common /packet/  amassx,xk,pk
c
       eye=(0.,1.)
       pi = acos(-1.0)
       alenx=xmax-xmin
       propfacx=-dt/2./amassx*(2.*pi)**2
       do kx=1,nptx
          if(kx.le.(nptx/2+1)) then
             nx=kx-1
          else
             nx=kx-1-nptx
          end if
          xsc=0.
          if(nx.ne.0) xsc=real(nx)/alenx
          tprop(kx)=exp(eye*(propfacx*xsc**2))
        end do
c
       return
       end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
      subroutine SetPotProp(dt,vprop)
c
c     Potential Energy part of the Trotter Expansion: exp(-i V dt/2)
c
      IMPLICIT NONE
      INTEGER NN,ii,kk,jj,nptx,i,j,k,npts
      REAL xmin,xmax,dx,dt,EVALUES,x
      COMPLEX vp,vprop,eye,dummy,psi,CRV
      parameter(npts=9,nptx=2**npts,NN=2)
      DIMENSION vprop(nptx,NN,NN),psi(NN,NN),CRV(NN,NN)
      DIMENSION vp(NN,NN),dummy(NN,NN),EVALUES(NN)
      common /xy/  xmin,xmax
      eye=(0.,1.)
      dx=(xmax-xmin)/real(nptx)
c
      do ii=1,nptx
         x=xmin+ii*dx
         CALL HAMIL(CRV,x)
         CALL SCHROC1(CRV,psi,EVALUES)
         vp(1,1)=exp(-eye*0.5*dt*EVALUES(1))
         vp(1,2)=0.0
         vp(2,1)=0.0
         vp(2,2)=exp(-eye*0.5*dt*EVALUES(2))
         do i=1,2
            do j=1,2
               dummy(i,j)=0.
               do k=1,2
                  dummy(i,j)=dummy(i,j)+vp(i,k)*psi(j,k)
               end do
            end do
         end do
         do i=1,2
            do j=1,2
               vp(i,j)=0.
               do k=1,2
                  vp(i,j)=vp(i,j)+psi(i,k)*dummy(k,j)
               end do
            end do
         end do
         do i=1,2
            do j=1,2
               kk=ii
               vprop(kk,i,j)=vp(i,j)/sqrt(1.0*nptx)
            end do
         end do
      end do
c
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PROPAGATE(vprop,tprop)
```

350

```fortran
c
c       Split Operator Fourier Transform Propagation Method
c       J. Comput. Phys. 47, 412 (1982); J. Chem. Phys. 78, 301 (1983)
c
        IMPLICIT NONE
        INTEGER i,j,kk,NN,in,ii,nptx,npts
        COMPLEX chi,vprop,chin1,chin2,tprop
        PARAMETER(npts=9,nptx=2**npts,NN=2)
        DIMENSION chin1(nptx),chin2(nptx)
        DIMENSION tprop(nptx),vprop(nptx,NN,NN)
        COMMON / wfunc/ chi(nptx,NN)
c
c       Apply potential energy part of the Trotter Expansion
c
        DO ii=1,nptx
           in=ii
           chin1(in)=0.0
           chin2(in)=0.0
           DO j=1,NN
              kk=ii
              chin1(in)=chin1(in)+vprop(kk,1,j)*chi(kk,j)
              chin2(in)=chin2(in)+vprop(kk,2,j)*chi(kk,j)
           END DO
        END DO
c
c       Fourier Transform wave-packet to the momentum representation
c
        CALL fourn(chin1,nptx,1,1)
        CALL fourn(chin2,nptx,1,1)
c
c       Apply kinetic energy part of the Trotter Expansion
c
        DO ii=1,nptx
           in=ii
           kk=ii
           chin1(in)=tprop(kk)*chin1(in)
           chin2(in)=tprop(kk)*chin2(in)
        END DO
c
c       Inverse Fourier Transform wave-packet to the coordinate representation
c
        CALL fourn(chin1,nptx,1,-1)
        CALL fourn(chin2,nptx,1,-1)
c
c       Apply potential energy part of the Trotter Expansion
c
        DO ii=1,nptx
           in=ii
           DO i=1,NN
              kk=ii
              chi(kk,i)=vprop(kk,i,1)*chin1(in)
```

351

```fortran
     1               +vprop(kk,i,2)*chin2(in)
          END DO
        END DO
        END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE HAMIL(CRV,x)
c
c     Hamiltonian Matrix
c
      IMPLICIT NONE
      INTEGER NN
      REAL x,VPOT1,VPOT2
      COMPLEX CRV
      PARAMETER(NN=2)
      DIMENSION CRV(NN,NN)
c
      CALL VA(VPOT1,x)
      CALL VB(VPOT2,x)
      CRV(1,1)=VPOT1
      CRV(2,2)=VPOT2
      CRV(1,2)=0.3    ! comment this line for item (b)
      CRV(2,1)=0.3    ! comment this line for item (b)
c      CRV(1,2)=0.3   ! uncomment this line for item (b)
c      CRV(2,1)=0.3   ! uncomment this line for item (b)
c
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE VA(V,x)
c
c     Potential Energy Surface: Harmonic Oscillator
c
      implicit none
      REAL V,x,amassx,xk,pk,rk,omega
      common /packet/ amassx,xk,pk
      omega=1.0
      rk=amassx*omega**2
      V=0.5*rk*x*x
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE VB(V,x1)
c
c     Potential Energy Surface: Double-Well Potential, tunneling dynamics
c
      implicit none
      REAL V,x1,x
      x=x1
      V=-0.5*x**2+1.0/(16.0*1.3544)*x**4
      RETURN
      END
```

352

```fortran
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PE(RV)
c
c     Expectation Value of the Potential Enegy
c
      IMPLICIT NONE
      INTEGER nptx,npts,kk,NN,j
      COMPLEX chi,EYE,RV
      REAL Vpot,omega,xmin,xmax,dx,pi,amassx,xk,pk,x,alpha
      PARAMETER(npts=9,nptx=2**npts,NN=2)
      DIMENSION RV(NN)
      COMMON / wfunc/ chi(nptx,NN)
      common /xy/  xmin,xmax
      common /packet/amassx,xk,pk

      dx=(xmax-xmin)/real(nptx)
      DO j=1,NN
         RV(j)=0.0
         do kk=1,nptx
            x=xmin+kk*dx
            IF(j.EQ.1) CALL VA(Vpot,x)
            IF(j.EQ.2) CALL VB(Vpot,x)
            RV(j)=RV(j)+chi(kk,j)*Vpot*conjg(chi(kk,j))*dx
         end do
      END DO
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine KE(RKE)
c
c     Expectation value of the kinetic energy
c
      IMPLICIT NONE
      INTEGER NN,kk,nptx,kx,nx,npts,j
      REAL dp,theta,wm,p,xmin,xmax,amassx,xk,pi,alenx,pk,rm,re,ri,dx
      COMPLEX eye,chi,Psip,chic,RKE
      parameter(npts=9,nptx=2**npts,NN=2)
      DIMENSION chic(nptx),RKE(NN)
      common /xy/  xmin,xmax
      common /packet/  amassx,xk,pk
      COMMON / wfunc/ chi(nptx,2)
c
      pi = acos(-1.0)
      dx=(xmax-xmin)/nptx
      dp=2.*pi/(xmax-xmin)
c
      DO j=1,NN
         RKE(j)=0.0
         do kk=1,nptx
            chic(kk)=chi(kk,j)
```

353

```fortran
            end do
            CALL fourn(chic,nptx,1,1)
            do kx=1,nptx
               if(kx.le.(nptx/2+1)) then
                  nx=kx-1
               else
                  nx=kx-1-nptx
               end if
               p=0.
               if(nx.ne.0) p = real(nx)*dp
               chic(kx)=p**2/(2.0*amassx)*chic(kx)/nptx
            end do
            CALL fourn(chic,nptx,1,-1)
            do kk=1,nptx
               RKE(j)=RKE(j)+conjg(chi(kk,j))*chic(kk)*dx
            end do
         END DO
         return
         end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE SCHROC1(CRV,EVECT,EVALUES)
c
c     Hamiltonian Matrix Diagonalization
c
c     CRV: HERMITIAN MATRIX (INPUT)
c     EVECT: EIGENVECTORS (OUTPUT)
c     EVALUES: EIGENVALUES (OUTPUT)
c
      INTEGER N,I,J,NP
      REAL EVALUES,CRV2,EVECT2
      COMPLEX CRV,EVECT
      PARAMETER(N=2,NP=2)
      DIMENSION CRV(N,N),EVECT(N,N),EVALUES(N),E(NP)
      DIMENSION CRV2(N,N),EVECT2(N,N)
C
      DO I=1,N
         EVALUES(I)=0.0
         E(I)=0.0
         DO J=1,N
            CRV2(J,I)=CRV(J,I)
         END DO
      END DO
      CALL TRED2(CRV2,N,NP,EVALUES,E)
      CALL TQLI(EVALUES,E,N,NP,CRV2)
      CALL EIGSRT(EVALUES,CRV2,N,NP)
C
      DO I=1,N
         DO J=1,N
            EVECT(J,I)=CRV2(J,I)
         END DO
      END DO
```

354

```
C
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutines from Numerical Recipes to compute FFT
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE FOURN(DATA,NN,NDIM,ISIGN)
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION NN(NDIM),DATA(*)
      NTOT=1
      DO 11 IDIM=1,NDIM
          NTOT=NTOT*NN(IDIM)
 11   CONTINUE
      NPREV=1
      DO 18 IDIM=1,NDIM
          N=NN(IDIM)
          NREM=NTOT/(N*NPREV)
          IP1=2*NPREV
          IP2=IP1*N
          IP3=IP2*NREM
          I2REV=1
          DO 14 I2=1,IP2,IP1
             IF(I2.LT.I2REV)THEN
                DO 13 I1=I2,I2+IP1-2,2
                   DO 12 I3=I1,IP3,IP2
                       I3REV=I2REV+I3-I2
                       TEMPR=DATA(I3)
                       TEMPI=DATA(I3+1)
                       DATA(I3)=DATA(I3REV)
                       DATA(I3+1)=DATA(I3REV+1)
                       DATA(I3REV)=TEMPR
                       DATA(I3REV+1)=TEMPI
 12                CONTINUE
 13             CONTINUE
             ENDIF
             IBIT=IP2/2
 1           IF ((IBIT.GE.IP1).AND.(I2REV.GT.IBIT)) THEN
                 I2REV=I2REV-IBIT
                 IBIT=IBIT/2
                 GO TO 1
             ENDIF
             I2REV=I2REV+IBIT
 14       CONTINUE
          IFP1=IP1
 2        IF(IFP1.LT.IP2)THEN
             IFP2=2*IFP1
             THETA=ISIGN*6.28318530717959D0/(IFP2/IP1)
             WPR=-2.D0*DSIN(0.5D0*THETA)**2
             WPI=DSIN(THETA)
             WR=1.D0
             WI=0.D0
```

355

```
            DO 17 I3=1,IFP1,IP1
                DO 16 I1=I3,I3+IP1-2,2
                    DO 15 I2=I1,IP3,IFP2
                        K1=I2
                        K2=K1+IFP1
                        TEMPR=SNGL(WR)*DATA(K2)-SNGL(WI)*DATA(K2+1)
                        TEMPI=SNGL(WR)*DATA(K2+1)+SNGL(WI)*DATA(K2)
                        DATA(K2)=DATA(K1)-TEMPR
                        DATA(K2+1)=DATA(K1+1)-TEMPI
                        DATA(K1)=DATA(K1)+TEMPR
                        DATA(K1+1)=DATA(K1+1)+TEMPI
 15                 CONTINUE
 16             CONTINUE
                WTEMP=WR
                WR=WR*WPR-WI*WPI+WR
                WI=WI*WPR+WTEMP*WPI+WI
 17         CONTINUE
            IFP1=IFP2
            GO TO 2
        ENDIF
        NPREV=N*NPREV
 18   CONTINUE
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutines to compute eigenvalues and eigenvectors
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE TRED2(A,N,NP,D,E)
      IMPLICIT NONE
      INTEGER I,J,K,L,N,NP
      REAL A,D,E,H,SCALE,F,G,HH
      DIMENSION A(NP,NP),D(NP),E(NP)
      IF(N.GT.1)THEN
          DO 18 I=N,2,-1
              L=I-1
              H=0.
              SCALE=0.
              IF(L.GT.1)THEN
                  DO 11 K=1,L
                      SCALE=SCALE+ABS(A(I,K))
 11               CONTINUE
                  IF(SCALE.EQ.0.)THEN
                      E(I)=A(I,L)
                  ELSE
                      DO 12 K=1,L
                          A(I,K)=A(I,K)/SCALE
                          H=H+A(I,K)**2
 12                   CONTINUE
                      F=A(I,L)
                      G=-SIGN(SQRT(H),F)
                      E(I)=SCALE*G
```

356

```fortran
                        H=H-F*G
                        A(I,L)=F-G
                        F=0.
                        DO 15 J=1,L
                            A(J,I)=A(I,J)/H
                            G=0.
                            DO 13 K=1,J
                                G=G+A(J,K)*A(I,K)
13                          CONTINUE
                            IF(L.GT.J)THEN
                                DO 14 K=J+1,L
                                    G=G+A(K,J)*A(I,K)
14                              CONTINUE
                            ENDIF
                            E(J)=G/H
                            F=F+E(J)*A(I,J)
15                      CONTINUE
                        HH=F/(H+H)
                        DO 17 J=1,L
                            F=A(I,J)
                            G=E(J)-HH*F
                            E(J)=G
                            DO 16 K=1,J
                                A(J,K)=A(J,K)-F*E(K)-G*A(I,K)
16                          CONTINUE
17                      CONTINUE
                    ENDIF
                ELSE
                    E(I)=A(I,L)
                ENDIF
                D(I)=H
18          CONTINUE
        ENDIF
        D(1)=0.
        E(1)=0.
        DO 23 I=1,N
            L=I-1
            IF(D(I).NE.0.)THEN
                DO 21 J=1,L
                    G=0.
                    DO 19 K=1,L
                        G=G+A(I,K)*A(K,J)
19                  CONTINUE
                    DO 20 K=1,L
                        A(K,J)=A(K,J)-G*A(K,I)
20                  CONTINUE
21              CONTINUE
            ENDIF
            D(I)=A(I,I)
            A(I,I)=1.
            IF(L.GE.1)THEN
```

357

```
              DO 22 J=1,L
                  A(I,J)=0.
                  A(J,I)=0.
 22           CONTINUE
          ENDIF
 23   CONTINUE
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE TQLI(D,E,N,NP,Z)
      IMPLICIT NONE
      INTEGER N,NP,I,K,L,M,ITER
      REAL D,E,Z,DD,G,R,S,C,P,F,B
      DIMENSION D(NP),E(NP),Z(NP,NP)
      IF (N.GT.1) THEN
          DO 11 I=2,N
              E(I-1)=E(I)
 11       CONTINUE
          E(N)=0.
          DO 15 L=1,N
              ITER=0
 1            DO 12 M=L,N-1
                  DD=ABS(D(M))+ABS(D(M+1))
                  IF (ABS(E(M))+DD.EQ.DD) GO TO 2
 12           CONTINUE
              M=N
 2            IF(M.NE.L)THEN
                  IF(ITER.EQ.30) PAUSE 'too many iterations!'
                  ITER=ITER+1
                  G=(D(L+1)-D(L))/(2.*E(L))
                  R=SQRT(G**2+1.)
                  G=D(M)-D(L)+E(L)/(G+SIGN(R,G))
                  S=1.
                  C=1.
                  P=0.
                  DO 14 I=M-1,L,-1
                      F=S*E(I)
                      B=C*E(I)
                      IF(ABS(F).GE.ABS(G))THEN
                          C=G/F
                          R=SQRT(C**2+1.)
                          E(I+1)=F*R
                          S=1./R
                          C=C*S
                      ELSE
                          S=F/G
                          R=SQRT(S**2+1.)
                          E(I+1)=G*R
                          C=1./R
                          S=S*C
                      ENDIF
```

358

```fortran
                         G=D(I+1)-P
                         R=(D(I)-G)*S+2.*C*B
                         P=S*R
                         D(I+1)=G+P
                         G=C*R-B
                         DO 13 K=1,N
                             F=Z(K,I+1)
                             Z(K,I+1)=S*Z(K,I)+C*F
                             Z(K,I)=C*Z(K,I)-S*F
 13                      CONTINUE
 14                 CONTINUE
                    D(L)=D(L)-P
                    E(L)=G
                    E(M)=0.
                    GO TO 1
               ENDIF
 15       CONTINUE
        ENDIF
        RETURN
        END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        SUBROUTINE EIGSRT(D,V,N,NP)
        IMPLICIT NONE
        INTEGER N,NP,I,J,K
        REAL D,V,P
        DIMENSION D(NP),V(NP,NP)
        DO 13 I=1,N-1
          K=I
          P=D(I)
          DO 11 J=I+1,N
            IF(D(J).GE.P)THEN
              K=J
              P=D(J)
            ENDIF
 11       CONTINUE
          IF(K.NE.I)THEN
            D(K)=D(I)
            D(I)=P
            DO 12 J=1,N
              P=V(J,I)
              V(J,I)=V(J,K)
              V(J,K)=P
 12         CONTINUE
          ENDIF
 13     CONTINUE
        RETURN
        END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        SUBROUTINE PIKSRT(N,ARR)
        IMPLICIT NONE
        INTEGER I,J,N
```

359

```
      REAL ARR,A
      DIMENSION ARR(N)
      DO 12 J=2,N
        A=ARR(J)
        DO 11 I=J-1,1,-1
          IF(ARR(I).LE.A)GO TO 10
          ARR(I+1)=ARR(I)
11      CONTINUE
        I=0
10      ARR(I+1)=A
12    CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

61.13   Problem 12SH

Problem 12SH:

The output of this program can be generated and visualized as follows. Cut the source code attached below, save it in a file named Problem12SH.f, compile it by typing

```
gfortran Problem12SH.f -o Problem12SH
```

run it by typing

```
./Problem12SH
```

The snapshots of the time-dependent wave-packet can be visualized as a movie by typing

```
gnuplot<pp_SH
```

where the file named

```
pp_SH
```

has the following lines:

Download from (http://ursula.chem.yale.edu/~batista/P12/P12_SH/pp_SH)

```
set yrange[-2:5]
set dat sty l
plot "arch.0001" u 1:2 lw 3
pause .1
plot "arch.0002" u 1:2 lw 3
pause .1
plot "arch.0003" u 1:2 lw 3
pause .1
plot "arch.0004" u 1:2 lw 3
pause .1
plot "arch.0005" u 1:2 lw 3
pause .1
plot "arch.0006" u 1:2 lw 3
pause .1
plot "arch.0007" u 1:2 lw 3
pause .1
plot "arch.0008" u 1:2 lw 3
pause .1
plot "arch.0009" u 1:2 lw 3
pause .1
plot "arch.0010" u 1:2 lw 3
pause .1
plot "arch.0011" u 1:2 lw 3
pause .1
plot "arch.0012" u 1:2 lw 3
pause .1
plot "arch.0013" u 1:2 lw 3
pause .1
```

```
plot "arch.0014" u 1:2 lw 3
pause .1
plot "arch.0015" u 1:2 lw 3
pause .1
plot "arch.0016" u 1:2 lw 3
pause .1
plot "arch.0017" u 1:2 lw 3
pause .1
plot "arch.0018" u 1:2 lw 3
pause .1
plot "arch.0019" u 1:2 lw 3
pause .1
plot "arch.0020" u 1:2 lw 3
pause .1
plot "arch.0021" u 1:2 lw 3
pause .1
plot "arch.0022" u 1:2 lw 3
pause .1
plot "arch.0023" u 1:2 lw 3
pause .1
plot "arch.0024" u 1:2 lw 3
pause .1
plot "arch.0025" u 1:2 lw 3
pause .1
plot "arch.0026" u 1:2 lw 3
pause .1
plot "arch.0027" u 1:2 lw 3
pause .1
plot "arch.0028" u 1:2 lw 3
pause .1
plot "arch.0029" u 1:2 lw 3
pause .1
plot "arch.0030" u 1:2 lw 3
pause .1
```

```fortran
      PROGRAM SOFTSH
c
c     SOFT Surface Hopping (SOFT/SH) Method (Chen and Batista 2006)
c     1-D nonadiabatic wave-packet propagation
c
      IMPLICIT NONE
      INTEGER NN,npts,nptx,ndump,kt,ntraj,ih1,ih2
      INTEGER istep,nstep,iseed,i,j,k,ik,n1,n2
      REAL dt,rn
      COMPLEX vprop,tprop,energy
      PARAMETER(npts=9,nptx=2**npts,NN=2)
      DIMENSION vprop(nptx,NN,NN),tprop(nptx)
      COMMON /ENER/ energy(NN)
      COMMON /hops/ih1(5051,2),ih2(5051,3)
c
      iseed=912371
      call srand(iseed)
      CALL ReadParam(nstep,ndump,dt)
      ntraj=nstep+1
      n1=ntraj
c
      DO i=2,nstep
         DO j=(i+1),(nstep+1)
            ntraj=ntraj+1
            IF(ntraj.GT.5051) THEN
               PRINT *, "increase dimension of ih1, ih2"
               STOP
            END IF
            ih1(ntraj,1)=i       ! index for hop 1 in trajs with 2 hops
            ih1(ntraj,2)=j       ! index for hop 2 in trajs with 2 hops
         END DO
      END DO
      n2=ntraj
c
c      DO ik=n1+1,ntraj
c         PRINT *, "traj", ik, ih1(ik,1), ih1(ik,2)
c      END DO
c
      DO i=2,nstep-1
         DO j=(i+1),nstep
            DO k=(j+1),(nstep+1)
               ntraj=ntraj+1
               IF(ntraj.GT.5051) THEN
                  PRINT *, "increase dimension of ih1, ih2"
                  STOP
               END IF
               ih2(ntraj,1)=i    ! index for hop 1 in trajs with 3 hops
               ih2(ntraj,2)=j    ! index for hop 2 in trajs with 3 hops
```

363

```fortran
            ih2(ntraj,3)=k     ! index for hop 3 in trajs with 3 hops
         END DO
       END DO
     END DO
c
c     DO ik=n2+1,ntraj
c       PRINT *, "traj", ik, ih2(ik,1), ih2(ik,2), ih2(ik,3)
c     END DO
c
     DO kt=1,ntraj
         IF(mod(kt-1,10).EQ.0)
   1         PRINT *, "Traj = ", kt,", total = ", ntraj
        call Initialize(kt)
        CALL SetKinProp(dt,tprop)
        CALL SetPotProp(dt,vprop)
        CALL energies(energy)
c
        DO istep=1,nstep+1
           IF(istep.GT.1)
   1           CALL PROPAGATE(vprop,tprop,dt,kt,ntraj,istep,nstep)
           IF(mod((istep-1),ndump).EQ.0) THEN
              CALL ACCUM(istep,ndump,dt,kt,nstep)
           END IF
        END DO
     END DO
c
     DO istep=1,nstep+1
        IF(mod((istep-1),ndump).EQ.0) THEN
           CALL SAVEWF(istep,ndump,dt)
        END IF
     END DO
c
 22   FORMAT(6(e13.6,2x))
     END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
     SUBROUTINE energies(energy)
     IMPLICIT NONE
     INTEGER j,NN
     COMPLEX energy,RV,RKE
     PARAMETER (NN=2)
     DIMENSION RV(NN),RKE(NN),energy(NN)
     CALL PE(RV)
     CALL KE(RKE)
     DO j=1,NN
        energy(j)=RV(j)+RKE(j)
     END DO
     RETURN
     END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
     subroutine ReadParam(nstep,ndump,dt)
c
```

364

```fortran
c     Parameters defining the grid (xmin, xmax), integration time step (dt),
c     mass (amassx), initial position (xk), initial momentum (pk),
c     number of propagation steps (nstep), and how often to save a pic (ndump)
c
      IMPLICIT NONE
      INTEGER ntype,nstep,nrpt,ireport,ndump,nlit
      REAL xmin,xmax,pk,amassx,xk,dt
      common /packet/  amassx,xk,pk
      common /xy/  xmin,xmax
c
      xmin=-6.0
      xmax=6.0
      dt=0.2
      amassx=1.0
      xk=-2.2
      pk=0.
      nstep=30
      ndump=1
c
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Initialize(kt)

      IMPLICIT NONE
      INTEGER NN,nptx,npts,kk,j,kt,ns
      COMPLEX chi0,chi,EYE,CRV,c1
      REAL omega,xk2,xmin,xmax,dx,pi,amassx,xk,pk,x,alpha,alpha2
      PARAMETER(npts=9,nptx=2**npts,NN=2)
      DIMENSION CRV(NN,NN)
      common /xy/  xmin,xmax
      common /packet/  amassx,xk,pk
      COMMON / wfunc/ chi(nptx,NN)
      COMMON / iwfunc/ chi0(nptx,NN)
      COMMON/cumul/ c1(nptx,200,2)
      COMMON / OCCUP/ ns
c
      EYE=(0.0,1.0)
      pi= acos(-1.0)
      omega=1.
      dx=(xmax-xmin)/real(nptx)
      ns = 1
c
c     Wave Packet Initialization: Gaussian centered at xk, with momentum pk
c
      alpha=amassx*omega
      do kk=1,nptx
         x=xmin+kk*dx
         chi(kk,1)=((alpha/pi)**0.25)
     1        *exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
         chi(kk,2)=chi(kk,1)*.0
```

```
c
         chi0(kk,1)=chi(kk,1)
         chi0(kk,2)=chi(kk,2)
      end do
c
      IF (kt.EQ.1) THEN
         DO kk=1,200
            DO j=1,nptx
               c1(j,kk,1)=0.0
               c1(j,kk,2)=0.0
            END DO
         END DO
      END IF
c
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE SAVEWF(je2,ndump,dt)
c
c     Dump Time Evolved Wave packet
c
      IMPLICIT NONE
      INTEGER je2,nptx,npts,kk,NN,ncount,ndump,jj
      COMPLEX chi,CRV,energy,psi,Psia,c1,c2
      character*9 B
      REAL V,x1,c1a,x,xmin,xmax,dx,EVALUES,dt,r1,r2
      PARAMETER(npts=9,nptx=2**npts,NN=2)
      DIMENSION CRV(NN,NN),EVALUES(NN)
      DIMENSION psi(NN,NN)
      COMMON/cumul/ c1(nptx,200,2)
      common /xy/  xmin,xmax
      COMMON / wfunc/ chi(nptx,NN)
      COMMON /ENER/ energy(NN)
c
c      IF(je2.EQ.1) CALL energies(energy)
      jj=je2/ndump
      write(B, '(A,i4.4)') 'arch.', jj
      OPEN(1,FILE=B)
      dx=(xmax-xmin)/real(nptx)
      ncount=(je2-1)/ndump
c
c     Save Wave-packet components
c
      do kk=1,nptx
         x=xmin+kk*dx
         r1=abs(c1(kk,jj,1))
         write(1,33) x,r1+real(energy(1))
      end do
      write(1,33)

      do kk=1,nptx
```

366

```fortran
         x=xmin+kk*dx
         r2=abs(c1(kk,jj,2))
         write(1,33) x,r2+real(energy(2))
      end do
      write(1,33)

      do kk=1,nptx
         x=xmin+kk*dx
          write(1,33) x,real(energy(2))
      end do
      write(1,33)

      do kk=1,nptx
         x=xmin+kk*dx
          write(1,33) x,real(energy(1))
      end do
      write(1,33)
c
c     Save Adiabatic states
c
      do kk=1,nptx
         x=xmin+kk*dx
         CALL HAMIL(CRV,x)
         CALL SCHROC1(CRV,psi,EVALUES)
         write(1,33) x,EVALUES(1)
      end do
      write(1,33)
      do kk=1,nptx
         x=xmin+kk*dx
         CALL HAMIL(CRV,x)
         CALL SCHROC1(CRV,psi,EVALUES)
         write(1,33) x,EVALUES(2)
      end do
      CLOSE(1)
 33   format(6(e13.6,2x))
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE ACCUM(istep,ndump,dt,kt,nstep)
c
c     Accumulate Time Evolved Wave packet
c
      IMPLICIT NONE
      INTEGER istep,nptx,npts,kk,NN,ncount,ndump
      INTEGER jj,ns,kt,ih1,ih2,nstep,n1,n2
      COMPLEX chi,CRV,energy,psi,Psia,c1,c2
      character*9 B
      REAL V,x1,c1a,x,xmin,xmax,dx,EVALUES,dt
      PARAMETER(npts=9,nptx=2**npts,NN=2)
      DIMENSION CRV(NN,NN),EVALUES(NN)
      DIMENSION psi(NN,NN)
```

```
      COMMON/cumul/ c1(nptx,200,2)
      common /xy/  xmin,xmax
      COMMON / wfunc/ chi(nptx,NN)
      COMMON /ENER/ energy(NN)
      COMMON / OCCUP/ ns
      COMMON /hops/ih1(5051,2),ih2(5051,3)
c
      jj=istep/ndump
      n1=nstep+1
      n2=n1+nstep*(nstep-1)/2
c
c     Accumulate Wave-packet components
c
      do kk=1,nptx
         IF(kt.LE.jj) c1(kk,jj,ns)=c1(kk,jj,ns)+chi(kk,ns)
         IF((kt.GT.n1).AND.(kt.LE.n2).AND.(istep.GE.ih1(kt,2)))
     1       c1(kk,jj,ns)=c1(kk,jj,ns)+chi(kk,ns)
         IF((kt.GT.n2).AND.(istep.GE.ih2(kt,3)))
     1       c1(kk,jj,ns)=c1(kk,jj,ns)+chi(kk,ns)
      end do
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetKinProp(dt,tprop)
c
c     Kinetic Energy part of the Trotter Expansion: exp(-i p^2 dt/(2 m))
c
      IMPLICIT NONE
      INTEGER nptx,kx,nx,npts,NN
      REAL xsc,xmin,xmax,propfacx,amassx,xk,pi,alenx,dt,pk
      COMPLEX tprop,eye
      parameter(npts=9,nptx=2**npts,NN=2)
      DIMENSION tprop(nptx)
      common /xy/  xmin,xmax
      common /packet/  amassx,xk,pk
c
      eye=(0.,1.)
      pi = acos(-1.0)
      alenx=xmax-xmin
      propfacx=-dt/2./amassx*(2.*pi)**2
      do kx=1,nptx
         if(kx.le.(nptx/2+1)) then
            nx=kx-1
         else
            nx=kx-1-nptx
         end if
         xsc=0.
         if(nx.ne.0) xsc=real(nx)/alenx
         tprop(kx)=exp(eye*(propfacx*xsc**2))
      end do
c
```

368

```fortran
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetPotProp(dt,vprop)
c
c     Potential Energy part of the Trotter Expansion: exp(-i V dt/2)
c
      IMPLICIT NONE
      INTEGER NN,ii,kk,jj,nptx,i,j,k,npts
      REAL xmin,xmax,dx,dt,EVALUES,x,V1,V2,VA
      COMPLEX vp,vprop,eye,dummy,psi,CRV
      parameter(npts=9,nptx=2**npts,NN=2)
      DIMENSION vprop(nptx,NN,NN),psi(NN,NN),CRV(NN,NN)
      DIMENSION vp(NN,NN),dummy(NN,NN),EVALUES(NN)
      common /xy/  xmin,xmax
      eye=(0.,1.)
      dx=(xmax-xmin)/real(nptx)
c
      do ii=1,nptx
         x=xmin+ii*dx
         CALL HAMIL(CRV,x)
         V1=CRV(1,1)
         V2=CRV(2,2)
         VA=0.5*(V1+V2)
         vp(1,1)=exp(-eye*0.5*dt*V1)
         vp(1,2)=exp(-eye*0.5*dt*VA)
         vp(2,1)=exp(-eye*0.5*dt*VA)
         vp(2,2)=exp(-eye*0.5*dt*V2)
         do i=1,2
            do j=1,2
               vprop(ii,i,j)=vp(i,j)/sqrt(1.0*nptx)
            end do
         end do
      end do
c
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PROPAGATE(vprop,tprop,dt,kt,ntraj,istep,nstep)
c
c     SOFT Surface Hopping (SOFT/SH) Method (Chen and Batista 2006)
c
c     SOFT = Split Operator Fourier Transform Propagation Method
c     J. Comput. Phys. 47, 412 (1982); J. Chem. Phys. 78, 301 (1983)
c
      IMPLICIT NONE
      INTEGER i,j,kk,NN,in,ii,nptx,npts,ih1,ih2,IND
      INTEGER NF,ns,ns_n,ns_o,kt,ntraj,istep,nstep,n1,n2
      COMPLEX chi,vprop,chin,tprop,eye,rc,wp,wm
      REAL cs,si,dt,rn,P
      PARAMETER(npts=9,nptx=2**npts,NN=2)
```

369

```
      DIMENSION chin(nptx)
      DIMENSION tprop(nptx),vprop(nptx,NN,NN)
      COMMON / wfunc/ chi(nptx,NN)
      COMMON / OCCUP/ ns
      COMMON /hops/ih1(5051,2),ih2(5051,3)
C
      eye=(0.0,1.0)
      P=(sin(0.3*dt))**2
      wp=1./cos(0.3*dt)
      wm=-eye/sin(0.3*dt)
      n1=nstep+1
      n2=n1+nstep*(nstep-1)/2
C
      IF(kt.EQ.1)  THEN
         NF=1                      ! No hops (i.e., diabatic dynamics)
      ELSE IF(kt.LE.n1) THEN    ! Flux from the other surface with 1 hop
         NF=1
         IF(kt.EQ.istep) THEN
            NF=0                   ! Make a hop at step 2,...,nstep+1
         END IF
      ELSE IF(kt.LE.n2) THEN         ! Flux that hopped to the other surface and back
         NF=1
         IF(istep.EQ.ih1(kt,1)) THEN
            NF=0                   ! Make a hop to the other state at step nstep+2,...,2*nstep
         ELSE IF (istep.EQ.ih1(kt,2)) THEN
            NF=0                   ! Make a hop back to the initial state at step ih+nstep+2,.
         END IF
      ELSE
         NF=1
         IF(istep.EQ.ih2(kt,1)) THEN
            NF=0                   ! Make a hop to the other state
         ELSE IF (istep.EQ.ih2(kt,2)) THEN
            NF=0                   ! Make a hop back to the initial state
         ELSE IF (istep.EQ.ih2(kt,3)) THEN
            NF=0                   ! Make a hop back to the other state
         END IF
      END IF
C
      IF(NF.EQ.1) THEN
         rc=(1.-P)*wp
         ns_n=ns                ! new surface index
         ns_o=ns                ! old surface index
      ELSE
         rc=P*wm
         ns_o = ns
         IF(ns_o.EQ.1) THEN
            ns_n = 2
         ELSE
            ns_n = 1
         END IF
         ns=ns_n
```

370

```fortran
         END IF
C
C        Apply potential energy part of the Trotter Expansion
C
         DO ii=1,nptx
            chin(ii)=vprop(ii,ns_n,ns_o)*chi(ii,ns_o)
         END DO
C
C        Fourier Transform wave-packet to the momentum representation
C
         CALL fourn(chin,nptx,1,1)
C
C        Apply kinetic energy part of the Trotter Expansion
C
         DO ii=1,nptx
            chin(ii)=tprop(ii)*chin(ii)
         END DO
C
C        Inverse Fourier Transform wave-packet to the coordinate representation
C
         CALL fourn(chin,nptx,1,-1)
C
C        Apply potential energy part of the Trotter Expansion
C
         DO ii=1,nptx
            chi(ii,ns_n)=rc*vprop(ii,ns_n,ns_o)*chin(ii)
         END DO
C
         END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
         SUBROUTINE HAMIL(CRV,x)
C
C        Hamiltonian Matrix
C
         IMPLICIT NONE
         INTEGER NN
         REAL x,VPOT1,VPOT2
         COMPLEX CRV
         PARAMETER(NN=2)
         DIMENSION CRV(NN,NN)
C
         CALL VA(VPOT1,x)
         CALL VB(VPOT2,x)
         CRV(1,1)=VPOT1
         CRV(2,2)=VPOT2
         CRV(1,2)=0.3
         CRV(2,1)=0.3
C
         RETURN
         END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```fortran
      SUBROUTINE VA(V,x)
c
c     Potential Energy Surface: Harmonic Oscillator
c
      implicit none
      REAL V,x,amassx,xk,pk,rk,omega
      common /packet/ amassx,xk,pk
      omega=1.0
      rk=amassx*omega**2
      V=0.5*rk*x*x
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE VB(V,x1)
c
c     Potential Energy Surface: Double-Well Potential, tunneling dynamics
c
      implicit none
      REAL V,x1,x
      x=x1
      V=-0.5*x**2+1.0/(16.0*1.3544)*x**4
      RETURN
      END


cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PE(RV)
c
c     Expectation Value of the Potential Enegy
c
      IMPLICIT NONE
      INTEGER nptx,npts,kk,NN,j
      COMPLEX chi,EYE,RV
      REAL Vpot,omega,xmin,xmax,dx,pi,amassx,xk,pk,x,alpha
      PARAMETER(npts=9,nptx=2**npts,NN=2)
      DIMENSION RV(NN)
      COMMON / wfunc/ chi(nptx,NN)
      common /xy/   xmin,xmax
      common /packet/amassx,xk,pk

      dx=(xmax-xmin)/real(nptx)
      DO j=1,NN
         RV(j)=0.0
         do kk=1,nptx
            x=xmin+kk*dx
            IF(j.EQ.1) CALL VA(Vpot,x)
            IF(j.EQ.2) CALL VB(Vpot,x)
            RV(j)=RV(j)+chi(kk,j)*Vpot*conjg(chi(kk,j))*dx
         end do
      END DO
      RETURN
      END
```

372

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine KE(RKE)
c
c     Expectation value of the kinetic energy
c
      IMPLICIT NONE
      INTEGER NN,kk,nptx,kx,nx,npts,j
      REAL dp,theta,wm,p,xmin,xmax,amassx,xk,pi,alenx,pk,rm,re,ri,dx
      COMPLEX eye,chi,Psip,chic,RKE
      parameter(npts=9,nptx=2**npts,NN=2)
      DIMENSION chic(nptx),RKE(NN)
      common /xy/   xmin,xmax
      common /packet/  amassx,xk,pk
      COMMON / wfunc/ chi(nptx,2)
c
      pi = acos(-1.0)
      dx=(xmax-xmin)/nptx
      dp=2.*pi/(xmax-xmin)
c
      DO j=1,NN
         RKE(j)=0.0
         do kk=1,nptx
            chic(kk)=chi(kk,j)
         end do
         CALL fourn(chic,nptx,1,1)
         do kx=1,nptx
            if(kx.le.(nptx/2+1)) then
               nx=kx-1
            else
               nx=kx-1-nptx
            end if
            p=0.
            if(nx.ne.0) p = real(nx)*dp
            chic(kx)=p**2/(2.0*amassx)*chic(kx)/nptx
         end do
         CALL fourn(chic,nptx,1,-1)
         do kk=1,nptx
            RKE(j)=RKE(j)+conjg(chi(kk,j))*chic(kk)*dx
         end do
      END DO
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE SCHROC1(CRV,EVECT,EVALUES)
c
c     Hamiltonian Matrix Diagonalization
c
c     CRV: HERMITIAN MATRIX (INPUT)
c     EVECT: EIGENVECTORS (OUTPUT)
c     EVALUES: EIGENVALUES (OUTPUT)
c
```

```
      INTEGER N,I,J,NP
      REAL EVALUES,CRV2,EVECT2
      COMPLEX CRV,EVECT
      PARAMETER(N=2,NP=2)
      DIMENSION CRV(N,N),EVECT(N,N),EVALUES(N),E(NP)
      DIMENSION CRV2(N,N),EVECT2(N,N)
C
      DO I=1,N
         EVALUES(I)=0.0
         E(I)=0.0
         DO J=1,N
            CRV2(J,I)=CRV(J,I)
         END DO
      END DO
      CALL TRED2(CRV2,N,NP,EVALUES,E)
      CALL TQLI(EVALUES,E,N,NP,CRV2)
      CALL EIGSRT(EVALUES,CRV2,N,NP)
c
      DO I=1,N
         DO J=1,N
            EVECT(J,I)=CRV2(J,I)
         END DO
      END DO
C
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutines from Numerical Recipes to compute FFT
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE FOURN(DATA,NN,NDIM,ISIGN)
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION NN(NDIM),DATA(*)
      NTOT=1
      DO 11 IDIM=1,NDIM
         NTOT=NTOT*NN(IDIM)
 11   CONTINUE
      NPREV=1
      DO 18 IDIM=1,NDIM
         N=NN(IDIM)
         NREM=NTOT/(N*NPREV)
         IP1=2*NPREV
         IP2=IP1*N
         IP3=IP2*NREM
         I2REV=1
         DO 14 I2=1,IP2,IP1
            IF(I2.LT.I2REV)THEN
               DO 13 I1=I2,I2+IP1-2,2
                  DO 12 I3=I1,IP3,IP2
                     I3REV=I2REV+I3-I2
                     TEMPR=DATA(I3)
                     TEMPI=DATA(I3+1)
```

374

```
                    DATA(I3)=DATA(I3REV)
                    DATA(I3+1)=DATA(I3REV+1)
                    DATA(I3REV)=TEMPR
                    DATA(I3REV+1)=TEMPI
12              CONTINUE
13          CONTINUE
          ENDIF
          IBIT=IP2/2
1         IF ((IBIT.GE.IP1).AND.(I2REV.GT.IBIT)) THEN
              I2REV=I2REV-IBIT
              IBIT=IBIT/2
              GO TO 1
          ENDIF
          I2REV=I2REV+IBIT
14      CONTINUE
        IFP1=IP1
2       IF(IFP1.LT.IP2)THEN
            IFP2=2*IFP1
            THETA=ISIGN*6.28318530717959D0/(IFP2/IP1)
            WPR=-2.D0*DSIN(0.5D0*THETA)**2
            WPI=DSIN(THETA)
            WR=1.D0
            WI=0.D0
            DO 17 I3=1,IFP1,IP1
                DO 16 I1=I3,I3+IP1-2,2
                    DO 15 I2=I1,IP3,IFP2
                        K1=I2
                        K2=K1+IFP1
                        TEMPR=SNGL(WR)*DATA(K2)-SNGL(WI)*DATA(K2+1)
                        TEMPI=SNGL(WR)*DATA(K2+1)+SNGL(WI)*DATA(K2)
                        DATA(K2)=DATA(K1)-TEMPR
                        DATA(K2+1)=DATA(K1+1)-TEMPI
                        DATA(K1)=DATA(K1)+TEMPR
                        DATA(K1+1)=DATA(K1+1)+TEMPI
15                  CONTINUE
16              CONTINUE
                WTEMP=WR
                WR=WR*WPR-WI*WPI+WR
                WI=WI*WPR+WTEMP*WPI+WI
17          CONTINUE
            IFP1=IFP2
            GO TO 2
        ENDIF
        NPREV=N*NPREV
18    CONTINUE
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutines to compute eigenvalues and eigenvectors
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE TRED2(A,N,NP,D,E)
```

```fortran
      IMPLICIT NONE
      INTEGER I,J,K,L,N,NP
      REAL A,D,E,H,SCALE,F,G,HH
      DIMENSION A(NP,NP),D(NP),E(NP)
      IF(N.GT.1)THEN
         DO 18 I=N,2,-1
            L=I-1
            H=0.
            SCALE=0.
            IF(L.GT.1)THEN
               DO 11 K=1,L
                  SCALE=SCALE+ABS(A(I,K))
11             CONTINUE
               IF(SCALE.EQ.0.)THEN
                  E(I)=A(I,L)
               ELSE
                  DO 12 K=1,L
                     A(I,K)=A(I,K)/SCALE
                     H=H+A(I,K)**2
12                CONTINUE
                  F=A(I,L)
                  G=-SIGN(SQRT(H),F)
                  E(I)=SCALE*G
                  H=H-F*G
                  A(I,L)=F-G
                  F=0.
                  DO 15 J=1,L
                     A(J,I)=A(I,J)/H
                     G=0.
                     DO 13 K=1,J
                        G=G+A(J,K)*A(I,K)
13                   CONTINUE
                     IF(L.GT.J)THEN
                        DO 14 K=J+1,L
                           G=G+A(K,J)*A(I,K)
14                      CONTINUE
                     ENDIF
                     E(J)=G/H
                     F=F+E(J)*A(I,J)
15                CONTINUE
                  HH=F/(H+H)
                  DO 17 J=1,L
                     F=A(I,J)
                     G=E(J)-HH*F
                     E(J)=G
                     DO 16 K=1,J
                        A(J,K)=A(J,K)-F*E(K)-G*A(I,K)
16                   CONTINUE
17                CONTINUE
               ENDIF
            ELSE
```

```
                  E(I)=A(I,L)
            ENDIF
            D(I)=H
18       CONTINUE
      ENDIF
      D(1)=0.
      E(1)=0.
      DO 23 I=1,N
         L=I-1
         IF(D(I).NE.0.)THEN
            DO 21 J=1,L
               G=0.
               DO 19 K=1,L
                  G=G+A(I,K)*A(K,J)
19             CONTINUE
               DO 20 K=1,L
                  A(K,J)=A(K,J)-G*A(K,I)
20             CONTINUE
21          CONTINUE
         ENDIF
         D(I)=A(I,I)
         A(I,I)=1.
         IF(L.GE.1)THEN
            DO 22 J=1,L
               A(I,J)=0.
               A(J,I)=0.
22          CONTINUE
         ENDIF
23    CONTINUE
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE TQLI(D,E,N,NP,Z)
      IMPLICIT NONE
      INTEGER N,NP,I,K,L,M,ITER
      REAL D,E,Z,DD,G,R,S,C,P,F,B
      DIMENSION D(NP),E(NP),Z(NP,NP)
      IF (N.GT.1) THEN
         DO 11 I=2,N
            E(I-1)=E(I)
11       CONTINUE
         E(N)=0.
         DO 15 L=1,N
            ITER=0
1           DO 12 M=L,N-1
               DD=ABS(D(M))+ABS(D(M+1))
               IF (ABS(E(M))+DD.EQ.DD) GO TO 2
12          CONTINUE
            M=N
2           IF(M.NE.L)THEN
               IF(ITER.EQ.30) PAUSE 'too many iterations!'
```

377

```fortran
                ITER=ITER+1
                G=(D(L+1)-D(L))/(2.*E(L))
                R=SQRT(G**2+1.)
                G=D(M)-D(L)+E(L)/(G+SIGN(R,G))
                S=1.
                C=1.
                P=0.
                DO 14 I=M-1,L,-1
                   F=S*E(I)
                   B=C*E(I)
                   IF(ABS(F).GE.ABS(G))THEN
                      C=G/F
                      R=SQRT(C**2+1.)
                      E(I+1)=F*R
                      S=1./R
                      C=C*S
                   ELSE
                      S=F/G
                      R=SQRT(S**2+1.)
                      E(I+1)=G*R
                      C=1./R
                      S=S*C
                   ENDIF
                   G=D(I+1)-P
                   R=(D(I)-G)*S+2.*C*B
                   P=S*R
                   D(I+1)=G+P
                   G=C*R-B
                   DO 13 K=1,N
                      F=Z(K,I+1)
                      Z(K,I+1)=S*Z(K,I)+C*F
                      Z(K,I)=C*Z(K,I)-S*F
13                 CONTINUE
14              CONTINUE
                D(L)=D(L)-P
                E(L)=G
                E(M)=0.
                GO TO 1
             ENDIF
15       CONTINUE
      ENDIF
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE EIGSRT(D,V,N,NP)
      IMPLICIT NONE
      INTEGER N,NP,I,J,K
      REAL D,V,P
      DIMENSION D(NP),V(NP,NP)
      DO 13 I=1,N-1
        K=I
```

```fortran
            P=D(I)
            DO 11 J=I+1,N
              IF(D(J).GE.P)THEN
                K=J
                P=D(J)
              ENDIF
11          CONTINUE
            IF(K.NE.I)THEN
              D(K)=D(I)
              D(I)=P
              DO 12 J=1,N
                P=V(J,I)
                V(J,I)=V(J,K)
                V(J,K)=P
12            CONTINUE
            ENDIF
13        CONTINUE
          RETURN
          END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
          SUBROUTINE PIKSRT(N,ARR)
          IMPLICIT NONE
          INTEGER I,J,N
          REAL ARR,A
          DIMENSION ARR(N)
          DO 12 J=2,N
            A=ARR(J)
            DO 11 I=J-1,1,-1
              IF(ARR(I).LE.A)GO TO 10
              ARR(I+1)=ARR(I)
11          CONTINUE
            I=0
10          ARR(I+1)=A
12        CONTINUE
          RETURN
          END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

61.14 Problem 15.1

Computational Problem 15.1:

Write a program to solve the time independent Schrödinger equation by using the DVR method and apply it to find the first 4 eigenvalues and eigenfunctions of the Harmonic oscillator introduced by Eq. (17) with $m = 1$ and $\omega = 1$. Verify that the eigenvalues are $E(\nu) = (1/2 + \nu)\hbar\omega$, $\nu = 0$–10.

Download the source code from (http://ursula.chem.yale.edu/~batista/classes/v572/dvrho1.f)

```
      PROGRAM DVR
c
c     This code computes the eigenvalues and eigenvectors of a Harmonic
c     oscillator V(x)=0.5*m*w**2*(x-4.)**2
c     The KE matrix is described according to Eq. (81) of the lecture notes
c     that corresponds to Eq.(A7) of JCP (1991) 96:1982-1991.
c
      CALL READPARAM()
      CALL Hamiltonian()
      CALL EIGV()
      CALL DUMP()
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE READPARAM()
      implicit real *8(a-h,o-z)
      PARAMETER(npt=100,NC=1)
      COMMON /paque/ rmin(NC), rmax(NC), rmass(NC), dx(NC)
      rmin(1)=0.
      rmax(1)=8.
      rmass(1)=1.
      DO I=1,NC
         dx(I)=(rmax(I)-rmin(I)) / (npt-1)
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Hamiltonian()
      implicit real *8(a-h,o-z)
      PARAMETER(npt=100,NC=1)
      COMMON /paque/ rmin(NC), rmax(NC), rmass(NC), dx(NC)
      COMMON/HAM/ VHAM(npt,npt)
      pi=acos(-1.)
      DO i=1, npt
         DO ip=1, npt
            IF(i.EQ.ip) THEN
               VTEMP=pi*pi/3.0d0
c     VTEMP=VTEMP-0.5d0/dfloat(i)**2 ! for radial coord.
            ELSE
               VTEMP=2.d0/(i-ip)**2
c     VTEMP=VTEMP-2.d0/(i+ip)**2    ! for radial coord.
            END IF
            VHAM(i,ip)=VTEMP*(-1)**(i-ip)
```

```
  1               /dx(1)/ dx(1)/ (2.0*rmass(1))
           IF(i.EQ.ip) THEN
              r=rmin(1)+(i-1)*dx(1)
              VHAM(i,ip)=VHAM(i,ip)+V(r)
           END IF
         END DO
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      FUNCTION V(r)
      implicit real *8(a-h,o-z)
      V=0.5*1.*(r-4.)**2
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE EIGV()
C
C     Diagonalization
C
C     VHAM: HERMITIAN MATRIX (INPUT)
C     EVALUES: EIGENVALUES (OUTPUT)
C     EVECT: EIGENVECTORS (OUTPUT)
C
      implicit real *8(a-h,o-z)
      REAL CRV, EVALUES,E,EVECT
      PARAMETER(npt=100,NC=1,npt2=npt**NC)
      COMMON/ HAM/ VHAM(npt2,npt2)
      COMMON /paque/ rmin(NC), rmax(NC), rmass(NC), dx(NC)
      PARAMETER(N=npt2,NP=N)
      COMMON/EIG/ EVALUES(npt2),EVECT(npt2,npt2)
      DIMENSION E(NP)
C
      DO I=1,N
         EVALUES(I)=0.0
         E(I)=0.0
         DO J=1,N
            EVECT(J,I)=VHAM(J,I)
         END DO
      END DO
      CALL TRED22(EVECT,N,NP,EVALUES,E)
      CALL TQLI(EVALUES,E,N,NP,EVECT)
      CALL EIGSRT(EVALUES,EVECT,N,NP)
C
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE DUMP()
C
      implicit real *8(a-h,o-z)
      REAL CRV, EVALUES,EVECT,E
```

```fortran
      character*9 B
      PARAMETER(npt=100,NC=1,npt2=npt**NC)
      PARAMETER(N=npt2,NP=N)
      COMMON/EIG/ EVALUES(npt2),EVECT(npt2,npt2)
      COMMON /paque/ rmin(NC), rmax(NC), rmass(NC), dx(NC)
      DIMENSION r(NC),j(NC)
c
      DO k=1, 10
         IND=npt2-(k-1)
         PRINT *, "E(",k,")=",EVALUES(IND)
      END DO
c
      DO l=1,4
      IND=npt2-(l-1)
      write(B, '(A,i4.4)') 'wave.', l
      OPEN(10,FILE=B)
      rsum=0.0
      DO i=1,npt2
         r(1)=rmin(1)+(i-1)*dx(1)
         WRITE(10,22) r(1),V(r),EVALUES(IND)
    1        ,EVALUES(IND)+EVECT(i,IND)
         rsum=rsum + EVECT(i,IND)**2
      END DO
      PRINT *, "norm(",l,")=",rsum
      END DO
      CLOSE(10)
 22   FORMAT(6(e13.6,2x))
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutines to compute eigenvalues and eigenvectors from NR
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE TRED22(A,N,NP,D,E)
      IMPLICIT NONE
      INTEGER I,J,K,L,N,NP
      REAL A,D,E,H,SCALE,F,G,HH
      DIMENSION A(NP,NP),D(NP),E(NP)
      IF(N.GT.1)THEN
         DO 18 I=N,2,-1
            L=I-1
            H=0.
            SCALE=0.
            IF(L.GT.1)THEN
               DO 11 K=1,L
                  SCALE=SCALE+ABS(A(I,K))
 11            CONTINUE
               IF(SCALE.EQ.0.)THEN
                  E(I)=A(I,L)
               ELSE
                  DO 12 K=1,L
                     A(I,K)=A(I,K)/SCALE
```

```fortran
                        H=H+A(I,K)**2
12                  CONTINUE
                    F=A(I,L)
                    G=-SIGN(SQRT(H),F)
                    E(I)=SCALE*G
                    H=H-F*G
                    A(I,L)=F-G
                    F=0.
                    DO 15 J=1,L
                        A(J,I)=A(I,J)/H
                        G=0.
                        DO 13 K=1,J
                            G=G+A(J,K)*A(I,K)
13                      CONTINUE
                        IF(L.GT.J)THEN
                            DO 14 K=J+1,L
                                G=G+A(K,J)*A(I,K)
14                          CONTINUE
                        ENDIF
                        E(J)=G/H
                        F=F+E(J)*A(I,J)
15                  CONTINUE
                    HH=F/(H+H)
                    DO 17 J=1,L
                        F=A(I,J)
                        G=E(J)-HH*F
                        E(J)=G
                        DO 16 K=1,J
                            A(J,K)=A(J,K)-F*E(K)-G*A(I,K)
16                      CONTINUE
17                  CONTINUE
                ENDIF
            ELSE
                E(I)=A(I,L)
            ENDIF
            D(I)=H
18      CONTINUE
    ENDIF
    D(1)=0.
    E(1)=0.
    DO 23 I=1,N
        L=I-1
        IF(D(I).NE.0.)THEN
            DO 21 J=1,L
                G=0.
                DO 19 K=1,L
                    G=G+A(I,K)*A(K,J)
19              CONTINUE
                DO 20 K=1,L
                    A(K,J)=A(K,J)-G*A(K,I)
20              CONTINUE
```

383

```
21          CONTINUE
         ENDIF
         D(I)=A(I,I)
         A(I,I)=1.
         IF(L.GE.1)THEN
            DO 22 J=1,L
               A(I,J)=0.
               A(J,I)=0.
22          CONTINUE
         ENDIF
23    CONTINUE
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE TQLI(D,E,N,NP,Z)
      IMPLICIT NONE
      INTEGER N,NP,I,K,L,M,ITER
      REAL D,E,Z,DD,G,R,S,C,P,F,B
      DIMENSION D(NP),E(NP),Z(NP,NP)
      IF (N.GT.1) THEN
         DO 11 I=2,N
            E(I-1)=E(I)
11       CONTINUE
         E(N)=0.
         DO 15 L=1,N
            ITER=0
1           DO 12 M=L,N-1
               DD=ABS(D(M))+ABS(D(M+1))
               IF (ABS(E(M))+DD.EQ.DD) GO TO 2
12          CONTINUE
            M=N
2           IF(M.NE.L)THEN
               IF(ITER.EQ.30) PAUSE 'too many iterations!'
               ITER=ITER+1
               G=(D(L+1)-D(L))/(2.*E(L))
               R=SQRT(G**2+1.)
               G=D(M)-D(L)+E(L)/(G+SIGN(R,G))
               S=1.
               C=1.
               P=0.
               DO 14 I=M-1,L,-1
                  F=S*E(I)
                  B=C*E(I)
                  IF(ABS(F).GE.ABS(G))THEN
                     C=G/F
                     R=SQRT(C**2+1.)
                     E(I+1)=F*R
                     S=1./R
                     C=C*S
                  ELSE
                     S=F/G
```

384

```
                          R=SQRT(S**2+1.)
                          E(I+1)=G*R
                          C=1./R
                          S=S*C
                    ENDIF
                    G=D(I+1)-P
                    R=(D(I)-G)*S+2.*C*B
                    P=S*R
                    D(I+1)=G+P
                    G=C*R-B
                    DO 13 K=1,N
                        F=Z(K,I+1)
                        Z(K,I+1)=S*Z(K,I)+C*F
                        Z(K,I)=C*Z(K,I)-S*F
 13                 CONTINUE
 14             CONTINUE
                D(L)=D(L)-P
                E(L)=G
                E(M)=0.
                GO TO 1
            ENDIF
 15     CONTINUE
      ENDIF
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE EIGSRT(D,V,N,NP)
      IMPLICIT NONE
      INTEGER N,NP,I,J,K
      REAL D,V,P
      DIMENSION D(NP),V(NP,NP)
      DO 13 I=1,N-1
        K=I
        P=D(I)
        DO 11 J=I+1,N
          IF(D(J).GE.P)THEN
            K=J
            P=D(J)
          ENDIF
 11     CONTINUE
        IF(K.NE.I)THEN
          D(K)=D(I)
          D(I)=P
          DO 12 J=1,N
            P=V(J,I)
            V(J,I)=V(J,K)
            V(J,K)=P
 12       CONTINUE
        ENDIF
 13   CONTINUE
      RETURN
```

```
        END
ccccccccccccccccccccccccccccccccccccccccccccccc
        SUBROUTINE PIKSRT(N,ARR)
        IMPLICIT NONE
        INTEGER I,J,N
        REAL ARR,A
        DIMENSION ARR(N)
        DO 12 J=2,N
          A=ARR(J)
          DO 11 I=J-1,1,-1
            IF(ARR(I).LE.A)GO TO 10
            ARR(I+1)=ARR(I)
11        CONTINUE
          I=0
10        ARR(I+1)=A
12    CONTINUE
        RETURN
        END
ccccccccccccccccccccccccccccccccccccccccccccccc
```

To visualize the output of the program listed above, save it in a file named dvrho1.f, compile it by typing

```
 gfortran dvrho1.f -o dvrho1
```

and run it by typing

```
./dvrho1
```

Then, cut the script attached below, save it with the name scr_1 in the same directory where you run your code, and visualize the 4 eigenstates by typing

```
 gnuplot<scr_1
```

where the file named scr_1 has the following lines:
    Download from (http://ursula.chem.yale.edu/∼batista/classes/v572/scr_1)

```
set yrange[0:6]
set xrange[-10:10]
set dat sty l
plot "wave.0001"  u 1:2 lw 3
pause 1.
replot "wave.0001" u 1:3  lw 3
pause 1.
replot "wave.0001" u 1:4  lw 3
pause 1.
replot "wave.0002" u 1:3 lw 3
pause 1.
replot "wave.0002" u 1:4 lw 3
pause 1.
```

```
replot "wave.0003" u 1:3 lw 3
pause 1.
replot "wave.0003" u 1:4 lw 3
pause 1.
replot "wave.0004" u 1:3 lw 3
pause 1.
replot "wave.0004" u 1:4 lw 3
pause 5.0
```

61.15   Problem 15.2

Computational Problem 15.2:

Change the potential of the code written in 15.1 to that of a Morse oscillator $V(\hat{x}) = De(1 - \exp(-a(\hat{x} - x_e)))^2$, with $x_e = 0$, $De = 8$, and $a = \sqrt{k/(2D_e)}$, where $k = m\omega^2$, and recompute the eigenvalues and eigenfunctions.

Download the source code from (http://ursula.chem.yale.edu/~batista/classes/v572/dvrmo1.f)

```
      PROGRAM DVR
c
c     This code computes the eigenvalues and eigenvectors of a Morse
c     oscillator V(r)=De*(1.0-exp(-a*(r-re)))**2
c     The KE matrix is described according to Eq. (82) of the lecture notes
c     that corresponds to Eq.(A8) of JCP (1991) 96:1982-1991.
c
      CALL READPARAM()
      CALL Hamiltonian()
      CALL EIGV()
      CALL DUMP()
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE READPARAM()
      implicit real *8(a-h,o-z)
      PARAMETER(npt=100,NC=1)
      COMMON /paque/ rmin(NC), rmax(NC), rmass(NC), dx(NC)
      rmin(1)=0.
      rmax(1)=10.
      rmass(1)=1.
      DO I=1,NC
         dx(I)=(rmax(I)-rmin(I)) / (npt-1)
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Hamiltonian()
      implicit real *8(a-h,o-z)
      PARAMETER(npt=100,NC=1)
      COMMON /paque/ rmin(NC), rmax(NC), rmass(NC), dx(NC)
      COMMON/HAM/ VHAM(npt,npt)
      pi=acos(-1.)
      DO i=1, npt
         DO ip=1, npt
            IF(i.EQ.ip) THEN
               VTEMP=pi*pi/3.0d0
               VTEMP=VTEMP-0.5d0/dfloat(i)**2 ! for radial coord.
            ELSE
               VTEMP=2.d0/(i-ip)**2
               VTEMP=VTEMP-2.d0/(i+ip)**2     ! for radial coord.
            END IF
            VHAM(i,ip)=VTEMP*(-1)**(i-ip)
```

```
     1               /dx(1)/ dx(1)/ (2.0*rmass(1))
            IF(i.EQ.ip) THEN
                r=rmin(1)+(i-1)*dx(1)
                VHAM(i,ip)=VHAM(i,ip)+V(r)
            END IF
          END DO
       END DO
       RETURN
       END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       FUNCTION V(r)
       implicit real *8(a-h,o-z)
c       V=0.5*1.*(r-3.)**2
       De=8.0
       re=3.0
       rk=1.
       a=sqrt(rk/(2.0*De))
       V=De*(1.0-exp(-a*(r-re)))**2
       RETURN
       END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       SUBROUTINE EIGV()
C
c     Diagonalization
c
c     VHAM: HERMITIAN MATRIX (INPUT)
c     EVALUES: EIGENVALUES (OUTPUT)
c     EVECT: EIGENVECTORS (OUTPUT)
c
       implicit real *8(a-h,o-z)
       REAL CRV, EVALUES,E,EVECT
       PARAMETER(npt=100,NC=1,npt2=npt**NC)
       COMMON/ HAM/ VHAM(npt2,npt2)
       COMMON /paque/ rmin(NC), rmax(NC), rmass(NC), dx(NC)
       PARAMETER(N=npt2,NP=N)
       COMMON/EIG/ EVALUES(npt2),EVECT(npt2,npt2)
       DIMENSION E(NP)
C
       DO I=1,N
          EVALUES(I)=0.0
          E(I)=0.0
          DO J=1,N
             EVECT(J,I)=VHAM(J,I)
          END DO
       END DO
       CALL TRED22(EVECT,N,NP,EVALUES,E)
       CALL TQLI(EVALUES,E,N,NP,EVECT)
       CALL EIGSRT(EVALUES,EVECT,N,NP)
C
       RETURN
       END
```

389

```fortran
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE DUMP()
c
      implicit real *8(a-h,o-z)
      REAL CRV, EVALUES,EVECT,E
      character*9 B
      PARAMETER(npt=100,NC=1,npt2=npt**NC)
      PARAMETER(N=npt2,NP=N)
      COMMON/EIG/ EVALUES(npt2),EVECT(npt2,npt2)
      COMMON /paque/ rmin(NC), rmax(NC), rmass(NC), dx(NC)
      DIMENSION r(NC),j(NC)
c
      DO k=1, 10
         IND=npt2-(k-1)
         PRINT *, "E(",k,")=",EVALUES(IND)
      END DO
c
      DO l=1,4
      IND=npt2-(l-1)
      write(B, '(A,i4.4)') 'wave.', l
      OPEN(10,FILE=B)
      rsum=0.0
      DO i=1,npt2
         r(1)=rmin(1)+(i-1)*dx(1)
         WRITE(10,22) r(1),V(r),EVALUES(IND)
    1         ,EVALUES(IND)+EVECT(i,IND)
         rsum=rsum + EVECT(i,IND)**2
      END DO
      PRINT *, "norm(",l,")=",rsum
      END DO
      CLOSE(10)
 22   FORMAT(6(e13.6,2x))
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutines to compute eigenvalues and eigenvectors from NR
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE TRED22(A,N,NP,D,E)
      IMPLICIT NONE
      INTEGER I,J,K,L,N,NP
      REAL A,D,E,H,SCALE,F,G,HH
      DIMENSION A(NP,NP),D(NP),E(NP)
      IF(N.GT.1)THEN
         DO 18 I=N,2,-1
            L=I-1
            H=0.
            SCALE=0.
            IF(L.GT.1)THEN
               DO 11 K=1,L
                  SCALE=SCALE+ABS(A(I,K))
 11            CONTINUE
```

```
                     IF(SCALE.EQ.0.)THEN
                         E(I)=A(I,L)
                     ELSE
                         DO 12 K=1,L
                             A(I,K)=A(I,K)/SCALE
                             H=H+A(I,K)**2
12                       CONTINUE
                         F=A(I,L)
                         G=-SIGN(SQRT(H),F)
                         E(I)=SCALE*G
                         H=H-F*G
                         A(I,L)=F-G
                         F=0.
                         DO 15 J=1,L
                             A(J,I)=A(I,J)/H
                             G=0.
                             DO 13 K=1,J
                                 G=G+A(J,K)*A(I,K)
13                           CONTINUE
                             IF(L.GT.J)THEN
                                 DO 14 K=J+1,L
                                     G=G+A(K,J)*A(I,K)
14                               CONTINUE
                             ENDIF
                             E(J)=G/H
                             F=F+E(J)*A(I,J)
15                       CONTINUE
                         HH=F/(H+H)
                         DO 17 J=1,L
                             F=A(I,J)
                             G=E(J)-HH*F
                             E(J)=G
                             DO 16 K=1,J
                                 A(J,K)=A(J,K)-F*E(K)-G*A(I,K)
16                           CONTINUE
17                       CONTINUE
                     ENDIF
                 ELSE
                     E(I)=A(I,L)
                 ENDIF
                 D(I)=H
18           CONTINUE
         ENDIF
         D(1)=0.
         E(1)=0.
         DO 23 I=1,N
             L=I-1
             IF(D(I).NE.0.)THEN
                 DO 21 J=1,L
                     G=0.
                     DO 19 K=1,L
```

391

```
                      G=G+A(I,K)*A(K,J)
 19              CONTINUE
                DO 20 K=1,L
                   A(K,J)=A(K,J)-G*A(K,I)
 20              CONTINUE
 21          CONTINUE
          ENDIF
          D(I)=A(I,I)
          A(I,I)=1.
          IF(L.GE.1)THEN
             DO 22 J=1,L
                A(I,J)=0.
                A(J,I)=0.
 22          CONTINUE
          ENDIF
 23    CONTINUE
       RETURN
       END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       SUBROUTINE TQLI(D,E,N,NP,Z)
       IMPLICIT NONE
       INTEGER N,NP,I,K,L,M,ITER
       REAL D,E,Z,DD,G,R,S,C,P,F,B
       DIMENSION D(NP),E(NP),Z(NP,NP)
       IF (N.GT.1) THEN
          DO 11 I=2,N
             E(I-1)=E(I)
 11       CONTINUE
          E(N)=0.
          DO 15 L=1,N
             ITER=0
 1           DO 12 M=L,N-1
                DD=ABS(D(M))+ABS(D(M+1))
                IF (ABS(E(M))+DD.EQ.DD) GO TO 2
 12          CONTINUE
             M=N
 2           IF(M.NE.L)THEN
                IF(ITER.EQ.30) PAUSE 'too many iterations!'
                ITER=ITER+1
                G=(D(L+1)-D(L))/(2.*E(L))
                R=SQRT(G**2+1.)
                G=D(M)-D(L)+E(L)/(G+SIGN(R,G))
                S=1.
                C=1.
                P=0.
                DO 14 I=M-1,L,-1
                   F=S*E(I)
                   B=C*E(I)
                   IF(ABS(F).GE.ABS(G))THEN
                      C=G/F
                      R=SQRT(C**2+1.)
```

392

```fortran
                        E(I+1)=F*R
                        S=1./R
                        C=C*S
                    ELSE
                        S=F/G
                        R=SQRT(S**2+1.)
                        E(I+1)=G*R
                        C=1./R
                        S=S*C
                    ENDIF
                    G=D(I+1)-P
                    R=(D(I)-G)*S+2.*C*B
                    P=S*R
                    D(I+1)=G+P
                    G=C*R-B
                    DO 13 K=1,N
                        F=Z(K,I+1)
                        Z(K,I+1)=S*Z(K,I)+C*F
                        Z(K,I)=C*Z(K,I)-S*F
13                  CONTINUE
14              CONTINUE
                D(L)=D(L)-P
                E(L)=G
                E(M)=0.
                GO TO 1
            ENDIF
15      CONTINUE
      ENDIF
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE EIGSRT(D,V,N,NP)
      IMPLICIT NONE
      INTEGER N,NP,I,J,K
      REAL D,V,P
      DIMENSION D(NP),V(NP,NP)
      DO 13 I=1,N-1
        K=I
        P=D(I)
        DO 11 J=I+1,N
          IF(D(J).GE.P)THEN
            K=J
            P=D(J)
          ENDIF
11      CONTINUE
        IF(K.NE.I)THEN
          D(K)=D(I)
          D(I)=P
          DO 12 J=1,N
            P=V(J,I)
            V(J,I)=V(J,K)
```

393

```
          V(J,K)=P
12        CONTINUE
       ENDIF
13    CONTINUE
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PIKSRT(N,ARR)
      IMPLICIT NONE
      INTEGER I,J,N
      REAL ARR,A
      DIMENSION ARR(N)
      DO 12 J=2,N
        A=ARR(J)
        DO 11 I=J-1,1,-1
          IF(ARR(I).LE.A)GO TO 10
          ARR(I+1)=ARR(I)
11        CONTINUE
        I=0
10      ARR(I+1)=A
12    CONTINUE
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccc
```

To visualize the output of the program listed above, save it in a file named dvrmo1.f, compile it by typing

```
 gfortran dvrmo1.f -o dvrmo1
```

and run it by typing

```
./dvrmo1
```

Then, cut the script attached below, save it with the name scr_1 in the same directory where you run your code, and visualize the 4 eigenstates by typing

```
 gnuplot<scr_1
```

where the file named scr_1 has the following lines:

Download from (http://ursula.chem.yale.edu/∼batista/classes/v572/scr_1)

```
set yrange[0:6]
set xrange[-10:10]
set dat sty l
plot "wave.0001"  u 1:2 lw 3
pause 1.
replot "wave.0001" u 1:3  lw 3
pause 1.
replot "wave.0001" u 1:4  lw 3
```

```
pause 1.
replot "wave.0002" u 1:3 lw 3
pause 1.
replot "wave.0002" u 1:4 lw 3
pause 1.
replot "wave.0003" u 1:3 lw 3
pause 1.
replot "wave.0003" u 1:4 lw 3
pause 1.
replot "wave.0004" u 1:3 lw 3
pause 1.
replot "wave.0004" u 1:4 lw 3
pause 5.0
```

61.16   Problem 15.3

Computational Problem 15.3:

Generalize the program developed in 15.1 to solve the 2-dimensional Harmonic oscillator $V(x, y) = 1/2m\omega^2(x^2 + y^2)$ and apply it to find the first 4 eigenvalues and eigenfunctions of the Harmonic oscillator introduced by Eq. (17) with $m = 1$ and $\omega = 1$. Verify that the eigenvalues are $E(\nu) = (1 + \nu_1 + \nu_2)\hbar\omega$.

Download the source code from (http://ursula.chem.yale.edu/~batista/classes/v572/dvrho2.f)

```
      PROGRAM DVR
c
c     This code computes the eigenvalues and eigenvectors of the
c     2 dimensional harmonic oscillator V(x,y)= 0.5*(x.-5)**2 +0.5*(y.-5)**2
c     The KE matrix is described according to Eq. (82) of the lecture notes
c     that corresponds to Eq.(A8) of JCP (1991) 96:1982-1991.
c
      CALL READPARAM()
      CALL Hamiltonian()
      CALL EIGV()
      CALL DUMP()
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE READPARAM()
      implicit real *8(a-h,o-z)
      COMMON /paque/ rmin(2), rmax(2), rmass(2), dx(2)
      PARAMETER(npt=20)
      rmin(1)=0.
      rmax(1)=10.
      rmin(2)=0.
      rmax(2)=10
      rmass(1)=1.
      rmass(2)=1.
      DO I=1,2
         dx(I)=(rmax(I)-rmin(I)) / (npt-1)
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Hamiltonian()
      implicit real *8(a-h,o-z)
      COMMON /paque/ rmin(2), rmax(2), rmass(2), dx(2)
      PARAMETER(npt=20,npt2=npt*npt)
      DIMENSION r(2),j(2),jp(2)
      COMMON/HAM/ VHAM(npt2,npt2)
      pi=acos(-1.)
      DO i=1, npt2                ! i = i(1) + npt * (i(2)-1)
         DO ip=1, npt2           ! ip = ip(1) + npt * (ip(2)-1)
            j(1)=1+mod((i-1),npt)
            j(2)=1+abs((i-1)/npt)
            jp(1)=1+mod((ip-1),npt)
```

```
            jp(2)=1+abs((ip-1)/npt)
            VHAM(i,ip)=0.0
            DO k=1,2
                l=1
                IF(k.EQ.1) l=2
                IF(j(l).EQ.jp(l)) THEN
                    IF(j(k).EQ.jp(k)) THEN
                        VTEMP=pi*pi/3.0d0
c     VTEMP=VTEMP-0.5d0/dfloat(j(k))**2 ! radial
                    ELSE
                        VTEMP=2.d0/(j(k)-jp(k))**2
c     VTEMP=VTEMP-2.d0/(j(k)+jp(k))**2 ! radial
                    END IF
                    VHAM(i,ip)=VHAM(i,ip)+VTEMP*(-1)**(j(k)-jp(k))
     1                /dx(k)/ dx(k)/ (2.0*rmass(k))
                END IF
            END DO
            IF((j(1).EQ.jp(1)).AND.(j(2).EQ.jp(2))) THEN
                r(1)=rmin(1)+(j(1)-1)*dx(1)
                r(2)=rmin(2)+(j(2)-1)*dx(2)
                VHAM(i,ip)=VHAM(i,ip)+V(r)
            END IF
          END DO
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      FUNCTION V(r)
      implicit real *8(a-h,o-z)
      DIMENSION r(2)
      rOH=r(1)
      rHO=r(2)
      V = rhosc(rHO)+rhosc(rOH)
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE EIGV()
c
c     Diagonalization
c
c     VHAM: HERMITIAN MATRIX (INPUT)
c     EVALUES: EIGENVALUES (OUTPUT)
c     EVECT: EIGENVECTORS (OUTPUT)
c
      implicit real *8(a-h,o-z)
      REAL CRV, EVALUES,E,EVECT
      COMMON /paque/ rmin(2), rmax(2), rmass(2), dx(2)
      PARAMETER(npt=20,npt2=npt*npt)
      COMMON/ HAM/ VHAM(npt2,npt2)
      PARAMETER(N=npt2,NP=N)
      COMMON/EIG/ EVALUES(npt2),EVECT(npt2,npt2)
```

```
      DIMENSION E(NP)
C
      DO I=1,N
         EVALUES(I)=0.0
         E(I)=0.0
         DO J=1,N
            EVECT(J,I)=VHAM(J,I)
         END DO
      END DO
      CALL TRED22(EVECT,N,NP,EVALUES,E)
      CALL TQLI(EVALUES,E,N,NP,EVECT)
      CALL EIGSRT(EVALUES,EVECT,N,NP)
C
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE DUMP()
C
      implicit real *8(a-h,o-z)
      REAL CRV, EVALUES,EVECT,E
      character*9 B
      COMMON /paque/ rmin(2), rmax(2), rmass(2), dx(2)
      PARAMETER(npt=20,npt2=npt*npt)
      PARAMETER(N=npt2,NP=N)
      COMMON/EIG/ EVALUES(npt2),EVECT(npt2,npt2)
      DIMENSION r(2),j(2)
C
      DO k=1, 10
         IND=npt2-(k-1)
         PRINT *, "E(",k,")=",EVALUES(IND)
      END DO
C
      DO l=1,4
      IND=npt2-(l-1)
      write(B, '(A,i4.4)') 'wave.', l
      OPEN(10,FILE=B)
      DO i=1,npt2                 ! i = i(1) + npt * (i(2)-1)
         j(1)=1+mod((i-1),npt)
         j(2)=1+abs((i-1)/npt)
         r(1)=rmin(1)+(j(1)-1)*dx(1)
         r(2)=rmin(2)+(j(2)-1)*dx(2)
         WRITE(10,22) r(1),r(2), EVECT(i,IND),V(r)
         IF(j(1).EQ.npt) WRITE(10,22)
      END DO
      END DO
      CLOSE(10)
 22   FORMAT(6(e13.6,2x))
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      double precision function rhosc(r)
```

398

```fortran
      implicit real *8(a-h,o-z)
      rhosc=0.5*1.*(r-5.)**2
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutines to compute eigenvalues and eigenvectors from NR
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE TRED22(A,N,NP,D,E)
      IMPLICIT NONE
      INTEGER I,J,K,L,N,NP
      REAL A,D,E,H,SCALE,F,G,HH
      DIMENSION A(NP,NP),D(NP),E(NP)
      IF(N.GT.1)THEN
         DO 18 I=N,2,-1
            L=I-1
            H=0.
            SCALE=0.
            IF(L.GT.1)THEN
               DO 11 K=1,L
                  SCALE=SCALE+ABS(A(I,K))
 11            CONTINUE
               IF(SCALE.EQ.0.)THEN
                  E(I)=A(I,L)
               ELSE
                  DO 12 K=1,L
                     A(I,K)=A(I,K)/SCALE
                     H=H+A(I,K)**2
 12               CONTINUE
                  F=A(I,L)
                  G=-SIGN(SQRT(H),F)
                  E(I)=SCALE*G
                  H=H-F*G
                  A(I,L)=F-G
                  F=0.
                  DO 15 J=1,L
                     A(J,I)=A(I,J)/H
                     G=0.
                     DO 13 K=1,J
                        G=G+A(J,K)*A(I,K)
 13                  CONTINUE
                     IF(L.GT.J)THEN
                        DO 14 K=J+1,L
                           G=G+A(K,J)*A(I,K)
 14                     CONTINUE
                     ENDIF
                     E(J)=G/H
                     F=F+E(J)*A(I,J)
 15               CONTINUE
                  HH=F/(H+H)
                  DO 17 J=1,L
                     F=A(I,J)
```

```fortran
                        G=E(J)-HH*F
                        E(J)=G
                        DO 16 K=1,J
                           A(J,K)=A(J,K)-F*E(K)-G*A(I,K)
16                      CONTINUE
17                  CONTINUE
                 ENDIF
            ELSE
                E(I)=A(I,L)
            ENDIF
            D(I)=H
18      CONTINUE
      ENDIF
      D(1)=0.
      E(1)=0.
      DO 23 I=1,N
         L=I-1
         IF(D(I).NE.0.)THEN
             DO 21 J=1,L
                G=0.
                DO 19 K=1,L
                   G=G+A(I,K)*A(K,J)
19              CONTINUE
                DO 20 K=1,L
                   A(K,J)=A(K,J)-G*A(K,I)
20              CONTINUE
21           CONTINUE
         ENDIF
         D(I)=A(I,I)
         A(I,I)=1.
         IF(L.GE.1)THEN
             DO 22 J=1,L
                A(I,J)=0.
                A(J,I)=0.
22           CONTINUE
         ENDIF
23    CONTINUE
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE TQLI(D,E,N,NP,Z)
      IMPLICIT NONE
      INTEGER N,NP,I,K,L,M,ITER
      REAL D,E,Z,DD,G,R,S,C,P,F,B
      DIMENSION D(NP),E(NP),Z(NP,NP)
      IF (N.GT.1) THEN
         DO 11 I=2,N
             E(I-1)=E(I)
11       CONTINUE
         E(N)=0.
         DO 15 L=1,N
```

```
          ITER=0
1         DO 12 M=L,N-1
            DD=ABS(D(M))+ABS(D(M+1))
            IF (ABS(E(M))+DD.EQ.DD) GO TO 2
12        CONTINUE
          M=N
2         IF(M.NE.L)THEN
            IF(ITER.EQ.30) PAUSE 'too many iterations!'
            ITER=ITER+1
            G=(D(L+1)-D(L))/(2.*E(L))
            R=SQRT(G**2+1.)
            G=D(M)-D(L)+E(L)/(G+SIGN(R,G))
            S=1.
            C=1.
            P=0.
            DO 14 I=M-1,L,-1
              F=S*E(I)
              B=C*E(I)
              IF(ABS(F).GE.ABS(G))THEN
                C=G/F
                R=SQRT(C**2+1.)
                E(I+1)=F*R
                S=1./R
                C=C*S
              ELSE
                S=F/G
                R=SQRT(S**2+1.)
                E(I+1)=G*R
                C=1./R
                S=S*C
              ENDIF
              G=D(I+1)-P
              R=(D(I)-G)*S+2.*C*B
              P=S*R
              D(I+1)=G+P
              G=C*R-B
              DO 13 K=1,N
                F=Z(K,I+1)
                Z(K,I+1)=S*Z(K,I)+C*F
                Z(K,I)=C*Z(K,I)-S*F
13            CONTINUE
14          CONTINUE
            D(L)=D(L)-P
            E(L)=G
            E(M)=0.
            GO TO 1
          ENDIF
15      CONTINUE
      ENDIF
      RETURN
      END
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE EIGSRT(D,V,N,NP)
      IMPLICIT NONE
      INTEGER N,NP,I,J,K
      REAL D,V,P
      DIMENSION D(NP),V(NP,NP)
      DO 13 I=1,N-1
        K=I
        P=D(I)
        DO 11 J=I+1,N
          IF(D(J).GE.P)THEN
            K=J
            P=D(J)
          ENDIF
11      CONTINUE
        IF(K.NE.I)THEN
          D(K)=D(I)
          D(I)=P
          DO 12 J=1,N
            P=V(J,I)
            V(J,I)=V(J,K)
            V(J,K)=P
12        CONTINUE
        ENDIF
13    CONTINUE
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PIKSRT(N,ARR)
      IMPLICIT NONE
      INTEGER I,J,N
      REAL ARR,A
      DIMENSION ARR(N)
      DO 12 J=2,N
        A=ARR(J)
        DO 11 I=J-1,1,-1
          IF(ARR(I).LE.A)GO TO 10
          ARR(I+1)=ARR(I)
11      CONTINUE
        I=0
10      ARR(I+1)=A
12    CONTINUE
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

To visualize the output of the program listed above, save it in a file named dvrho2.f, compile it by typing

```
 gfortran dvrho2.f -o dvrho2
```

and run it by typing

```
./dvrho2
```

Then, cut the script attached below, save it with the name scr_2 in the same directory where you run your code, and visualize the 4 eigenstates by typing

```
 gnuplot<scr_2
```

where the file named scr_2 has the following lines:

Download from (http://ursula.chem.yale.edu/∼batista/classes/v572/scr_2)

```
set dat sty l
set param
set hidden3d
set contour
set cntrparam level 10
splot "wave.0001" title "Ground State"
pause 5.
splot "wave.0002" title "1st Excited State"
pause 5.
splot "wave.0003" title "2nd Excited  State"
pause 5.
splot "wave.0004" title "3rd Excited State"
pause 5.
```

61.17   Problem 15.4

Computational Problem 15.4:

15.4 Change the potential of the code written in 15.3 to that of a 2-dimensional Morse oscillator $V(\hat{x}, \hat{y}) = De(1 - \exp(-a(\hat{x} - x_e)))^2 + De(1 - \exp(-a(\hat{y} - x_e)))^2$, with $x_e = 0$, $De = 8$, and $a = \sqrt{k/(2D_e)}$, where $k = m\omega^2$, and recompute the eigenvalues and eigenfunctions.

Download the source code from (http://ursula.chem.yale.edu/~batista/classes/v572/dvrmo2.f)

```
      PROGRAM DVR
C
C     This code computes the eigenvalues and eigenvectors of the
C     2 dimensional Morse oscillator
C     V(x,y)=De*(1.0-exp(-a*(x-re)))**2+De*(1.0-exp(-a*(y-re)))**2
C     The KE matrix is described according to Eq. (82) of the lecture notes
C     that corresponds to Eq.(A8) of JCP (1991) 96:1982-1991.
C
      CALL READPARAM()
      CALL Hamiltonian()
      CALL EIGV()
      CALL DUMP()
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE READPARAM()
      implicit real *8(a-h,o-z)
      COMMON /paque/ rmin(2), rmax(2), rmass(2), dx(2)
      PARAMETER(npt=20)
      rmin(1)=0.
      rmax(1)=10.
      rmin(2)=0.
      rmax(2)=10
      rmass(1)=1.
      rmass(2)=1.
      DO I=1,2
         dx(I)=(rmax(I)-rmin(I)) / (npt-1)
      END DO
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Hamiltonian()
      implicit real *8(a-h,o-z)
      COMMON /paque/ rmin(2), rmax(2), rmass(2), dx(2)
      PARAMETER(npt=20,npt2=npt*npt)
      DIMENSION r(2),j(2),jp(2)
      COMMON/HAM/ VHAM(npt2,npt2)
      pi=acos(-1.)
      DO i=1, npt2                    ! i = i(1) + npt * (i(2)-1)
         DO ip=1, npt2                ! ip = ip(1) + npt * (ip(2)-1)
            j(1)=1+mod((i-1),npt)
            j(2)=1+abs((i-1)/npt)
            jp(1)=1+mod((ip-1),npt)
```

404

```fortran
            jp(2)=1+abs((ip-1)/npt)
            VHAM(i,ip)=0.0
            DO k=1,2
                l=1
                IF(k.EQ.1) l=2
                IF(j(l).EQ.jp(l)) THEN
                    IF(j(k).EQ.jp(k)) THEN
                        VTEMP=pi*pi/3.0d0-0.5d0/dfloat(j(k))**2
                    ELSE
                        VTEMP=2.d0/(j(k)-jp(k))**2-2.d0/(j(k)+jp(k))**2
                    END IF
                    VHAM(i,ip)=VHAM(i,ip)+VTEMP*(-1)**(j(k)-jp(k))
     1                   /dx(k)/ dx(k)/ (2.0*rmass(k))
                END IF
            END DO
            IF((j(1).EQ.jp(1)).AND.(j(2).EQ.jp(2))) THEN
                r(1)=rmin(1)+(j(1)-1)*dx(1)
                r(2)=rmin(2)+(j(2)-1)*dx(2)
                VHAM(i,ip)=VHAM(i,ip)+V(r)
            END IF
          END DO
        END DO
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      FUNCTION V(r)
      implicit real *8(a-h,o-z)
      DIMENSION r(2)
      rOH=r(1)
      rHO=r(2)
      V = rmorse(rHO)+rmorse(rOH)
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE EIGV()
c
c     Diagonalization
c
c     VHAM: HERMITIAN MATRIX (INPUT)
c     EVALUES: EIGENVALUES (OUTPUT)
c     EVECT: EIGENVECTORS (OUTPUT)
c
      implicit real *8(a-h,o-z)
      REAL CRV, EVALUES,E,EVECT
      COMMON /paque/ rmin(2), rmax(2), rmass(2), dx(2)
      PARAMETER(npt=20,npt2=npt*npt)
      COMMON/ HAM/ VHAM(npt2,npt2)
      PARAMETER(N=npt2,NP=N)
      COMMON/EIG/ EVALUES(npt2),EVECT(npt2,npt2)
      DIMENSION E(NP)
C
```

```
      DO I=1,N
         EVALUES(I)=0.0
         E(I)=0.0
         DO J=1,N
            EVECT(J,I)=VHAM(J,I)
         END DO
      END DO
      CALL TRED22(EVECT,N,NP,EVALUES,E)
      CALL TQLI(EVALUES,E,N,NP,EVECT)
      CALL EIGSRT(EVALUES,EVECT,N,NP)
C
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE DUMP()
C
      implicit real *8(a-h,o-z)
      REAL CRV, EVALUES,EVECT,E
      character*9 B
      COMMON /paque/ rmin(2), rmax(2), rmass(2), dx(2)
      PARAMETER(npt=20,npt2=npt*npt)
      PARAMETER(N=npt2,NP=N)
      COMMON/EIG/ EVALUES(npt2),EVECT(npt2,npt2)
      DIMENSION r(2),j(2)
C
      DO k=1, 10
         IND=npt2-(k-1)
         PRINT *, "E(",k,")=",EVALUES(IND)
      END DO
C
      DO l=1,4
      IND=npt2-(l-1)
      write(B, '(A,i4.4)') 'wave.', l
      OPEN(10,FILE=B)
      DO i=1,npt2               ! i = i(1) + npt * (i(2)-1)
         j(1)=1+mod((i-1),npt)
         j(2)=1+abs((i-1)/npt)
         r(1)=rmin(1)+(j(1)-1)*dx(1)
         r(2)=rmin(2)+(j(2)-1)*dx(2)
         WRITE(10,22) r(1),r(2), EVECT(i,IND),V(r)
         IF(j(1).EQ.npt) WRITE(10,22)
      END DO
      END DO
      CLOSE(10)
 22   FORMAT(6(e13.6,2x))
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      double precision function rmorse(r)
      implicit real *8(a-h,o-z)
      De=8.0
```

```
      re=3.0
      rk=1.
      a=sqrt(rk/(2.0*De))
      rmorse=De*(1.0-exp(-a*(r-re)))**2
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutines to compute eigenvalues and eigenvectors from NR
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE TRED22(A,N,NP,D,E)
      IMPLICIT NONE
      INTEGER I,J,K,L,N,NP
      REAL A,D,E,H,SCALE,F,G,HH
      DIMENSION A(NP,NP),D(NP),E(NP)
      IF(N.GT.1)THEN
         DO 18 I=N,2,-1
            L=I-1
            H=0.
            SCALE=0.
            IF(L.GT.1)THEN
               DO 11 K=1,L
                  SCALE=SCALE+ABS(A(I,K))
 11            CONTINUE
               IF(SCALE.EQ.0.)THEN
                  E(I)=A(I,L)
               ELSE
                  DO 12 K=1,L
                     A(I,K)=A(I,K)/SCALE
                     H=H+A(I,K)**2
 12               CONTINUE
                  F=A(I,L)
                  G=-SIGN(SQRT(H),F)
                  E(I)=SCALE*G
                  H=H-F*G
                  A(I,L)=F-G
                  F=0.
                  DO 15 J=1,L
                     A(J,I)=A(I,J)/H
                     G=0.
                     DO 13 K=1,J
                        G=G+A(J,K)*A(I,K)
 13                  CONTINUE
                     IF(L.GT.J)THEN
                        DO 14 K=J+1,L
                           G=G+A(K,J)*A(I,K)
 14                     CONTINUE
                     ENDIF
                     E(J)=G/H
                     F=F+E(J)*A(I,J)
 15               CONTINUE
                  HH=F/(H+H)
```

```
                    DO 17 J=1,L
                        F=A(I,J)
                        G=E(J)-HH*F
                        E(J)=G
                        DO 16 K=1,J
                            A(J,K)=A(J,K)-F*E(K)-G*A(I,K)
16                      CONTINUE
17                  CONTINUE
                ENDIF
            ELSE
                E(I)=A(I,L)
            ENDIF
            D(I)=H
18      CONTINUE
      ENDIF
      D(1)=0.
      E(1)=0.
      DO 23 I=1,N
          L=I-1
          IF(D(I).NE.0.)THEN
              DO 21 J=1,L
                  G=0.
                  DO 19 K=1,L
                      G=G+A(I,K)*A(K,J)
19                CONTINUE
                  DO 20 K=1,L
                      A(K,J)=A(K,J)-G*A(K,I)
20                CONTINUE
21            CONTINUE
          ENDIF
          D(I)=A(I,I)
          A(I,I)=1.
          IF(L.GE.1)THEN
              DO 22 J=1,L
                  A(I,J)=0.
                  A(J,I)=0.
22            CONTINUE
          ENDIF
23    CONTINUE
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE TQLI(D,E,N,NP,Z)
      IMPLICIT NONE
      INTEGER N,NP,I,K,L,M,ITER
      REAL D,E,Z,DD,G,R,S,C,P,F,B
      DIMENSION D(NP),E(NP),Z(NP,NP)
      IF (N.GT.1) THEN
          DO 11 I=2,N
              E(I-1)=E(I)
11        CONTINUE
```

```fortran
          E(N)=0.
          DO 15 L=1,N
            ITER=0
1           DO 12 M=L,N-1
                DD=ABS(D(M))+ABS(D(M+1))
                IF (ABS(E(M))+DD.EQ.DD) GO TO 2
12          CONTINUE
            M=N
2           IF(M.NE.L)THEN
                IF(ITER.EQ.30) PAUSE 'too many iterations!'
                ITER=ITER+1
                G=(D(L+1)-D(L))/(2.*E(L))
                R=SQRT(G**2+1.)
                G=D(M)-D(L)+E(L)/(G+SIGN(R,G))
                S=1.
                C=1.
                P=0.
                DO 14 I=M-1,L,-1
                    F=S*E(I)
                    B=C*E(I)
                    IF(ABS(F).GE.ABS(G))THEN
                        C=G/F
                        R=SQRT(C**2+1.)
                        E(I+1)=F*R
                        S=1./R
                        C=C*S
                    ELSE
                        S=F/G
                        R=SQRT(S**2+1.)
                        E(I+1)=G*R
                        C=1./R
                        S=S*C
                    ENDIF
                    G=D(I+1)-P
                    R=(D(I)-G)*S+2.*C*B
                    P=S*R
                    D(I+1)=G+P
                    G=C*R-B
                    DO 13 K=1,N
                        F=Z(K,I+1)
                        Z(K,I+1)=S*Z(K,I)+C*F
                        Z(K,I)=C*Z(K,I)-S*F
13                  CONTINUE
14              CONTINUE
                D(L)=D(L)-P
                E(L)=G
                E(M)=0.
                GO TO 1
            ENDIF
15        CONTINUE
      ENDIF
```

```fortran
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE EIGSRT(D,V,N,NP)
      IMPLICIT NONE
      INTEGER N,NP,I,J,K
      REAL D,V,P
      DIMENSION D(NP),V(NP,NP)
      DO 13 I=1,N-1
        K=I
        P=D(I)
        DO 11 J=I+1,N
          IF(D(J).GE.P)THEN
            K=J
            P=D(J)
          ENDIF
11      CONTINUE
        IF(K.NE.I)THEN
          D(K)=D(I)
          D(I)=P
          DO 12 J=1,N
            P=V(J,I)
            V(J,I)=V(J,K)
            V(J,K)=P
12        CONTINUE
        ENDIF
13    CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE PIKSRT(N,ARR)
      IMPLICIT NONE
      INTEGER I,J,N
      REAL ARR,A
      DIMENSION ARR(N)
      DO 12 J=2,N
        A=ARR(J)
        DO 11 I=J-1,1,-1
          IF(ARR(I).LE.A)GO TO 10
          ARR(I+1)=ARR(I)
11      CONTINUE
        I=0
10      ARR(I+1)=A
12    CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

To visualize the output of the program listed above, save it in a file named dvrmo2.f, compile it by typing

```
gfortran dvrmo2.f -o dvrmo2
```

and run it by typing

```
./dvrmo2
```

Then, cut the script attached below, save it with the name scr_2 in the same directory where you run your code, and visualize the 4 eigenstates by typing

```
gnuplot<scr_2
```

where the file named scr_2 has the following lines:
Download from (http://ursula.chem.yale.edu/~batista/classes/v572/scr_2)

```
set dat sty l
set param
set hidden3d
set contour
set cntrparam level 10
splot "wave.0001" title "Ground State"
pause 5.
splot "wave.0002" title "1st Excited State"
pause 5.
splot "wave.0003" title "2nd Excited  State"
pause 5.
splot "wave.0004" title "3rd Excited State"
pause 5.
```

61.18   Problem 15.5

15.5 Propagate a 1-d or 2-d wavepacket, initialized as a superposition of ground ψ_0 and first excited state ψ_1, $\psi(0) = 1/\sqrt{2}\psi_0 - 1/\sqrt{2}\psi_1$ of the 1-d or 2-d Morse oscillator defined in 15.2 and 15.3, by using the DVR Hamiltonian of item 15.4 and the Lanczos propagation scheme in the Krylov basis as implemented in *Expokit*. Compare your results with the analytic solution, based on the eigenvalues and eigenvectors: $\psi(t) = 1/\sqrt{2}\psi_0\exp(-iE_0t) - 1/\sqrt{2}\psi_1\exp(-iE_1t)$.

Download the 1D source code from here and the 2D source code from here

Untar the tarball files by typing

tar -xvf 1DMorse_expokit.tar

or

tar -xvf 2DMorse_expokit.tar

Compile by typing

make 1DMorse

or

make 2DMorse

and run the program by typing

./1DMorse

or

./1DMorse

Visualize the numerical and analytical time-dependent wavepackets by typing

gnuplot

61.19   Problem 16

Computational Problem 16: Modify the program for wave-packet propagation developed in Problem 12 and simulate the propagation of a wave packet in the symmetric double well

$$V(x) = -0.5x^2 + 1.0/(16.0 * 1.3544)x^4, \tag{1220}$$

for the initial state

$$\Phi_0(x) = \pi^{-1/4}e^{-0.5(x-x0)^2}, \tag{1221}$$

with $x_0 = -2.1$.

16.1: Propagate Φ_0 for 1000 a.u., using a propagation step $\tau = 0.1$ a.u. and compute $|\xi(t)|^2$.
16.2: Propagate Φ_0 for 1000 a.u. applying a sequence of 2-π pulses as described by Eq. (368) in the time-window $t = 305$–500 a.u. Compare $|\xi(t)|^2$ with the results obtained in 16.1.

Download the source code from
(http://ursula.chem.yale.edu/~batista/classes/v572/dw_cc.f)
Download the script for visualizing the results of $|\xi(t)|^2$ from
(http://ursula.chem.yale.edu/~batista/classes/v572/scr_prob)

The output of this program can be generated and visualized as follows. Cut the source code attached below, save it in a file named dw_cc.f, compile it by typing

```
gfortran dw_cc.f -o dw_cc
```

run it by typing

```
./dw_cc
```

The results for $|\xi(t)|^2$ can be visualized by typing

```
gnuplot<scr_prob
```

. The evolution of the wavepacket with 2-π pulses applied can be visualized by downloading the script from (http://ursula.chem.yale.edu/~batista/classes/v572/pp_p) and typing

```
gnuplot<pp_p
```

. You will see that the wave-packet gets trapped on left of the dividing barrier during the time-window $t = 3050$–5000 (snapshot frames 61–100 spaced at time intervals of 50 a.u.). The analogous results in the absence of perturbational pulses can be visualized by downloading the script from (http://ursula.chem.yale.edu/~batista/classes/v572/pp_n) and typing

```
gnuplot<pp_n
```

.

```fortran
      PROGRAM main
c
c     Coherent control of tunneling in a symmetric 1-dimensional double well
c     by using a sequence of 2-pi pulses that repetitively affect the phase
c     the wave packet component associated with the initial state relative
c     to the other terms in the expansion
c
      IMPLICIT NONE
      INTEGER NN,igammax,nptx,ndump,NFLAG,istep,nstep,ii,i
      REAL dt,p1g,p2g,p1l,p2l,rr,ra,re0
      COMPLEX vprop,tprop1,vprop_e,tprop1_e,energy
      PARAMETER(igammax=9,nptx=2**igammax,NN=2)
      DIMENSION vprop(nptx,NN,NN),tprop1(nptx),energy(NN)
      DIMENSION vprop_e(nptx,NN,NN),tprop1_e(nptx)
      COMMON/e00/re0
c
      NFLAG=1
      CALL ReadParam(nstep,ndump,dt)
      call inithr()
      CALL SetKinProp1(dt,tprop1)
      CALL SetPotProp(dt,vprop)
      CALL SetKinProp1_e(tprop1_e)
      CALL SetPotProp_e(vprop_e)
      DO istep=1,nstep+1
         IF(istep.GE.1) CALL PROPAGATE(vprop,tprop1)
         CALL ENERGY_s(vprop_e,tprop1_e,energy)
         IF(istep.EQ.1) re0=energy(2)
         IF(istep.GT.5000) NFLAG=0
         IF(mod((istep-1),ndump).EQ.0) THEN
            CALL DUMPWF(istep,ndump,p1g,p2g,p1l,p2l,energy)
            WRITE(10,22) dt*istep*2.4189E-2,p1g
     1           ,p1l,p2g,p2l,real(energy(1))
         END IF
         IF(((istep-1).GE.3050).AND.(NFLAG.EQ.1)) THEN
            CALL pulse()
         END IF
         IF(mod((istep-1),ndump).EQ.0) THEN
            PRINT *,"# steps", (istep-1)
            CALL DUMPWF(istep,ndump,p1g,p2g,p1l,p2l,energy)
         END IF
      END DO
 22   FORMAT(6(e13.6,2x))
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine ReadParam(nstep,ndump,dt)
c
c     Parameters defining the grid (xmin, xmax), integration time step (dt),
c     mass (amassx), initial position (xk), initial momentum (pk),
c     number of propagation steps (nstep), and how often to save a pic (ndump)
c
      IMPLICIT NONE
```

```fortran
      INTEGER ntype,nstep,nrpt,ireport,ndump,nlit
      REAL xmin,xmax,pk,amassx,xk,dt
      common /packet/  amassx,xk,pk
      common /xy/  xmin,xmax
c
      xmin=-10.0
      xmax=10.0
      dt=.1
      amassx=1.0
      xk=-2.1
      pk=0.0
      nstep=10000
      ndump=50
c
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE INITHR()

      IMPLICIT NONE
      INTEGER NN,nptx,igammax,kk
      COMPLEX chi0,chi,EYE,CRV
      REAL xk2,xmin,xmax,dx,pi,amassx,xk,pk,x,beta,beta2
      PARAMETER(igammax=9,nptx=2**igammax,NN=2)
      DIMENSION CRV(NN,NN)
      common /xy/  xmin,xmax
      common /packet/  amassx,xk,pk
      COMMON / wfunc/ chi(nptx,NN)
      COMMON / iwfunc/ chi0(nptx,NN)

      EYE=(0.0,1.0)
      pi= acos(-1.0)
      dx=(xmax-xmin)/real(nptx)
c
c     Wave Packet Initialization: Gaussian centered at xk, with momentum pk
c
      xk2=xk
      beta=.5
      beta2=.5
      do kk=1,nptx
         x=xmin+kk*dx
         chi(kk,1)=((2.0*beta/pi)**0.25)
     1        *exp(-beta*(x-xk)**2+EYE*pk*(x-xk))
         chi(kk,2)=chi(kk,1)
         chi0(kk,1)=((2.0*beta2/pi)**0.25)
     1        *exp(-beta2*(x-xk2)**2+EYE*pk*(x-xk2))
         chi0(kk,2)=chi0(kk,1)
      end do
c
c     Hamiltonian Matrix CRV
c
```

415

```
      do kk=1,nptx
         x=xmin+kk*dx
         CALL HAMIL(CRV,x)
c          WRITE(11,22) x,real(CRV(1,1)),real(CRV(2,2))
c     1          ,real(chi0(kk,1))
      END DO
 22   FORMAT(6(e13.6,2x))
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE DUMPWF(istep,ndump,p1g,p2g,p1l,p2l,energy)
c
c     Dump Time Evolved Wave packet
c
      IMPLICIT NONE
      INTEGER nptx,igammax,kk,NN,ncount,ndump,jj,istep
      COMPLEX chi,CRV,energy
      character*9 B,BB
      REAL re0,V,x1,c1,c2,x,xmin,xmax,dx,p1g,p2g,p1l,p2l
      PARAMETER(igammax=9,nptx=2**igammax,NN=2)
      DIMENSION CRV(NN,NN),energy(NN)
      common /xy/  xmin,xmax
      COMMON / wfunc/ chi(nptx,NN)
      COMMON/e00/re0
c
      jj=istep/ndump
      x1=2.1
      Call EXCITEDB(V,x1)
      write(B, '(A,i4.4)') 'arch.', jj
      write(BB, '(A,i4.4)') 'rrch.', jj
      OPEN(1,FILE=B)
      OPEN(2,FILE=BB)
      dx=(xmax-xmin)/real(nptx)
      ncount=jj
      p1g=0.0
      p2g=0.0
      p1l=0.0
      p2l=0.0
      do kk=1,nptx
         x=xmin+kk*dx
         c1=chi(kk,1)*conjg(chi(kk,1))
         c2=chi(kk,2)*conjg(chi(kk,2))
         IF(x.GE.0) THEN
            p1g=p1g+c1*dx
            p2g=p2g+c2*dx
         ELSE
            p1l=p1l+c1*dx
            p2l=p2l+c2*dx
         END IF
c          write(1,33) x,sqrt(c1)+real(energy(1)),sqrt(c2)+real(energy(2))
         write(1,33) x,sqrt(c1)+re0,sqrt(c2)+re0
```

416

```fortran
      end do
      write(1,33)

      do kk=1,nptx
         x=xmin+kk*dx
c         write(1,33) x,re0
c     1          ,re0
         write(2,33) x,real(chi(kk,1))+re0
     1          ,real(chi(kk,2))+re0
       end do
       write(2,33)

       do kk=1,nptx
          x=xmin+kk*dx
          write(1,33) x,re0
     1          ,re0
       end do
       write(1,33)

       do kk=1,nptx
          x=xmin+kk*dx
          CALL HAMIL(CRV,x)
          write(1,33) x,real(CRV(1,1)),real(CRV(2,2))
     1          ,real(CRV(1,1)),real(CRV(2,2))
       end do
       CLOSE(1)
       CLOSE(2)
 33    format(6(e13.6,2x))
c
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE pulse()
c
c     2 pi pulse
c
      IMPLICIT NONE
      INTEGER je2,nptx,igammax,kk,NN,ncount,ndump
      COMPLEX chi,pj2,chi0,EYE
      REAL c1,c2,x,xmin,xmax,dx,p1,p2,pi,phase
      PARAMETER(igammax=9,nptx=2**igammax,NN=2)
      common /xy/  xmin,xmax
      COMMON / wfunc/ chi(nptx,NN)
      COMMON / iwfunc/ chi0(nptx,NN)
c
      pi=acos(-1.0)
      phase=pi
      EYE=(0.0,1.0)
      dx=(xmax-xmin)/real(nptx)
      pj2=0.0
      do kk=1,nptx
```

```
           pj2=pj2+chi(kk,2)*chi0(kk,2)
        end do
        pj2=pj2*dx
c        PRINT *, pj2
        do kk=1,nptx
           chi(kk,2)=chi(kk,2)-chi0(kk,2)*pj2+
     1         chi0(kk,2)*pj2*exp(EYE*phase)
        end do
c
        RETURN
        END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine SetKinProp1(dt,tprop1)
c
c       Kinetic Energy part of the Trotter Expansion: exp(-i p^2 dt/(2 m))
c
        IMPLICIT NONE
        INTEGER nptx,kx,nx,igammax,NN
        REAL xsc,xmin,xmax,propfacx,amassx,xk,pi,alenx,dt,pk
        COMPLEX tprop1,eye
        parameter(igammax=9,nptx=2**igammax,NN=2)
        DIMENSION tprop1(nptx)
        common /xy/  xmin,xmax
        common /packet/  amassx,xk,pk
c
        eye=(0.,1.)
        pi = acos(-1.0)
        alenx=xmax-xmin
        propfacx=-dt/2./amassx*(2.*pi)**2
        do kx=1,nptx
           if(kx.le.(nptx/2+1)) then
              nx=kx-1
           else
              nx=kx-1-nptx
           end if
           xsc=0.
           if(nx.ne.0) xsc=real(nx)/alenx
           tprop1(kx)=exp(eye*(propfacx*xsc**2))
         end do
c
        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine SetPotProp(dt,vprop)
c
c       Potential Energy part of the Trotter Expansion: exp(-i V dt/2)
c
        IMPLICIT NONE
        INTEGER NN,ii,kk,jj,nptx,i,j,k,igammax
        REAL xmin,xmax,dx,dt,rsqnx,EVALUES,x
        COMPLEX vp,vprop,eye,dummy,psi,CRV
```

```
        parameter(igammax=9,nptx=2**igammax,NN=2)
        DIMENSION vprop(nptx,NN,NN),psi(NN,NN),CRV(NN,NN)
        DIMENSION vp(NN,NN),dummy(NN,NN),EVALUES(NN)
        common /xy/  xmin,xmax
        eye=(0.,1.)
        dx=(xmax-xmin)/real(nptx)
c
        do ii=1,nptx
           x=xmin+ii*dx
           CALL HAMIL(CRV,x)
           CALL SCHROC1(CRV,psi,EVALUES)
           vp(1,1)=exp(-eye*0.5*dt*EVALUES(1))
           vp(1,2)=0.0
           vp(2,1)=0.0
           vp(2,2)=exp(-eye*0.5*dt*EVALUES(2))
           do i=1,2
              do j=1,2
                 dummy(i,j)=0.
                 do k=1,2
                    dummy(i,j)=dummy(i,j)+vp(i,k)*psi(j,k)
                 end do
              end do
           end do
           do i=1,2
              do j=1,2
                 vp(i,j)=0.
                 do k=1,2
                    vp(i,j)=vp(i,j)+psi(i,k)*dummy(k,j)
                 end do
              end do
           end do
           rsqnx=1.0/sqrt(1.0*nptx)
           do i=1,2
              do j=1,2
                 kk=ii
                 vprop(kk,i,j)=vp(i,j)*rsqnx
              end do
           end do
        end do
c
        RETURN
        END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine SetKinProp1_e(tprop1)
c
c       Kinetic Energy part of the Trotter Expansion: exp(-i p^2 dt/(2 m))
c
        IMPLICIT NONE
        INTEGER nptx,kx,nx,igammax,NN
        REAL xsc,xmin,xmax,propfacx,amassx,xk,pi,alenx,dt,pk
        COMPLEX tprop1,eye
```

419

```
       parameter(igammax=9,nptx=2**igammax,NN=2)
       DIMENSION tprop1(nptx)
       common /xy/   xmin,xmax
       common /packet/   amassx,xk,pk
c
       eye=(0.,1.)
       pi = acos(-1.0)
       alenx=xmax-xmin
       propfacx=1./2./amassx*(2.*pi)**2
       do kx=1,nptx
          if(kx.le.(nptx/2+1)) then
             nx=kx-1
          else
             nx=kx-1-nptx
          end if
          xsc=0.
          if(nx.ne.0) xsc=real(nx)/alenx
          tprop1(kx)=propfacx*xsc**2
       end do
c
       return
       end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       subroutine SetPotProp_e(vprop)
c
c      Potential Energy part of the Trotter Expansion: exp(-i V dt/2)
c
       IMPLICIT NONE
       INTEGER NN,ii,kk,jj,nptx,i,j,k,igammax
       REAL xmin,xmax,dx,dt,rsqnx,EVALUES,x
       COMPLEX vp,vprop,eye,dummy,psi,CRV
       parameter(igammax=9,nptx=2**igammax,NN=2)
       DIMENSION vprop(nptx,NN,NN),psi(NN,NN),CRV(NN,NN)
       DIMENSION vp(NN,NN),dummy(NN,NN),EVALUES(NN)
       common /xy/   xmin,xmax
       eye=(0.,1.)
       dx=(xmax-xmin)/real(nptx)
c
       do ii=1,nptx
          x=xmin+ii*dx
          CALL HAMIL(CRV,x)
          CALL SCHROC1(CRV,psi,EVALUES)
          vp(1,1)=EVALUES(1)
          vp(1,2)=0.0
          vp(2,1)=0.0
          vp(2,2)=EVALUES(2)
          do i=1,2
             do j=1,2
                dummy(i,j)=0.
                do k=1,2
                   dummy(i,j)=dummy(i,j)+vp(i,k)*psi(j,k)
```

420

```
            end do
         end do
      end do
      do i=1,2
         do j=1,2
            vp(i,j)=0.
            do k=1,2
               vp(i,j)=vp(i,j)+psi(i,k)*dummy(k,j)
            end do
         end do
      end do
      rsqnx=1.0/sqrt(1.0*nptx)
      do i=1,2
         do j=1,2
            kk=ii
            vprop(kk,i,j)=vp(i,j)*rsqnx
         end do
      end do
      end do
c
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PROPAGATE(vprop,tprop1)
c
c     Split Operator Fourier Transform Propagation Method
c     J. Comput. Phys. 47, 412 (1982); J. Chem. Phys. 78, 301 (1983)
c
      IMPLICIT NONE
      INTEGER i,j,kk,NN,in,ii,nptx,igammax
      COMPLEX chi,vprop,chin1,chin2,tprop1
      PARAMETER(igammax=9,nptx=2**igammax,NN=2)
      DIMENSION chin1(nptx),chin2(nptx)
      DIMENSION tprop1(nptx),vprop(nptx,NN,NN)
      COMMON / wfunc/ chi(nptx,NN)
c
c     Apply potential energy part of the Trotter Expansion
c
      DO ii=1,nptx
         in=ii
         chin1(in)=0.0
         chin2(in)=0.0
         DO j=1,NN
            kk=ii
            chin1(in)=chin1(in)+vprop(kk,1,j)*chi(kk,j)
            chin2(in)=chin2(in)+vprop(kk,2,j)*chi(kk,j)
         END DO
      END DO
c
c     Fourier Transform wave-packet to the momentum representation
c
```

```fortran
      CALL fourn(chin1,nptx,1,1)
      CALL fourn(chin2,nptx,1,1)
c
c     Apply kinetic energy part of the Trotter Expansion
c
      DO ii=1,nptx
         in=ii
         kk=ii
         chin1(in)=tprop1(kk)*chin1(in)
         chin2(in)=tprop1(kk)*chin2(in)
      END DO
c
c     Inverse Fourier Transform wave-packet to the coordinate representation
c
      CALL fourn(chin1,nptx,1,-1)
      CALL fourn(chin2,nptx,1,-1)
c
c     Apply potential energy part of the Trotter Expansion
c
      DO ii=1,nptx
         in=ii
         DO i=1,NN
            kk=ii
            chi(kk,i)=vprop(kk,i,1)*chin1(in)
     1                +vprop(kk,i,2)*chin2(in)
         END DO
      END DO
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE ENERGY_s(vprop,tprop1,energy)
c
c     Split Operator Fourier Transform Propagation Method
c     J. Comput. Phys. 47, 412 (1982); J. Chem. Phys. 78, 301 (1983)
c
      IMPLICIT NONE
      INTEGER i,j,kk,NN,in,ii,nptx,igammax
      COMPLEX chi,vprop,chin1,chin2,tprop1,energy
      REAL xmin,xmax,dx,rsqnx
      PARAMETER(igammax=9,nptx=2**igammax,NN=2)
      DIMENSION chin1(nptx),chin2(nptx),energy(NN)
      DIMENSION tprop1(nptx),vprop(nptx,NN,NN)
      COMMON / wfunc/ chi(nptx,NN)
      common /xy/  xmin,xmax
c
c     Apply potential energy part of the Trotter Expansion
c
      rsqnx=1.0/sqrt(1.0*nptx)
      dx=(xmax-xmin)/real(nptx)
c
      DO in=1,nptx
         chin1(in)=chi(in,1)
```

```
         chin2(in)=chi(in,2)
      END DO
c
c     Fourier Transform wave-packet to the momentum representation
c
      CALL fourn(chin1,nptx,1,1)
      CALL fourn(chin2,nptx,1,1)
c
c     Apply kinetic energy part of the Trotter Expansion
c
      DO in=1,nptx
         chin1(in)=tprop1(in)*chin1(in)*rsqnx**2
         chin2(in)=tprop1(in)*chin2(in)*rsqnx**2
      END DO
c
c     Inverse Fourier Transform wave-packet to the coordinate representation
c
      CALL fourn(chin1,nptx,1,-1)
      CALL fourn(chin2,nptx,1,-1)
c
c      DO in=1,nptx
c         DO j=1,NN
c            chin1(in)=chin1(in)+vprop(in,1,j)*chi(in,j)
c            chin2(in)=chin2(in)+vprop(in,2,j)*chi(in,j)
c         END DO
c       END DO
c
      DO in=1,nptx
         chin1(in)=chin1(in)+vprop(in,1,1)*chi(in,1)/rsqnx
         chin2(in)=chin2(in)+vprop(in,2,2)*chi(in,2)/rsqnx
      END DO
c
      energy(1)=0.0
      energy(2)=0.0
      DO in=1,nptx
         energy(1)=energy(1)+conjg(chi(in,1))*chin1(in)*dx
         energy(2)=energy(2)+conjg(chi(in,2))*chin2(in)*dx
      END DO
c
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE HAMIL(CRV,x)
c
c     Hamiltonian Matrix
c
      IMPLICIT NONE
      INTEGER NN
      REAL x,VPOT1,VPOT2
      COMPLEX CRV
      PARAMETER(NN=2)
```

```
      DIMENSION CRV(NN,NN)
C
C        CALL EXCITEDA(VPOT1,x)
      CALL EXCITEDB(VPOT2,x)
      CRV(1,1)=VPOT2
      CRV(2,2)=VPOT2
      CRV(1,2)=0.00
      CRV(2,1)=0.00
C
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE EXCITEDA(V,x)
C
C      Potential Energy Surface: Harmonic Oscillator
C
      implicit none
      REAL V,x
      v=0.5*x*x
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE EXCITEDB(V,x1)
C
C      Potential Energy Surface: Double-Well Potential, tunneling dynamics
C
      implicit none
      REAL V,x1,x
      x=x1
C        if(abs(x).LE.(2.34)) x=2.34

      V=-0.5*x**2+1.0/(16.0*1.3544)*x**4
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE SCHROC1(CRV,EVECT,EVALUES)
C
C      Hamiltonian Matrix Diagonalization
C
C      CRV: HERMITIAN MATRIX (INPUT)
C      EVECT: EIGENVECTORS (OUTPUT)
C      EVALUES: EIGENVALUES (OUTPUT)
C
      INTEGER N,I,J,NP
      REAL EVALUES,CRV2,EVECT2
      COMPLEX CRV,EVECT
      PARAMETER(N=2,NP=2)
      DIMENSION CRV(N,N),EVECT(N,N),EVALUES(N),E(NP)
      DIMENSION CRV2(N,N),EVECT2(N,N)
C
      DO I=1,N
```

424

```
         EVALUES(I)=0.0
         E(I)=0.0
         DO J=1,N
            CRV2(J,I)=CRV(J,I)
         END DO
      END DO
      CALL TRED2(CRV2,N,NP,EVALUES,E)
      CALL TQLI(EVALUES,E,N,NP,CRV2)
      CALL EIGSRT(EVALUES,CRV2,N,NP)
C
      DO I=1,N
         DO J=1,N
            EVECT(J,I)=CRV2(J,I)
         END DO
      END DO
C
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutines from Numerical Recipes
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE FOURN(DATA,NN,NDIM,ISIGN)
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION NN(NDIM),DATA(*)
      NTOT=1
      DO 11 IDIM=1,NDIM
         NTOT=NTOT*NN(IDIM)
 11   CONTINUE
      NPREV=1
      DO 18 IDIM=1,NDIM
         N=NN(IDIM)
         NREM=NTOT/(N*NPREV)
         IP1=2*NPREV
         IP2=IP1*N
         IP3=IP2*NREM
         I2REV=1
         DO 14 I2=1,IP2,IP1
            IF(I2.LT.I2REV)THEN
               DO 13 I1=I2,I2+IP1-2,2
                  DO 12 I3=I1,IP3,IP2
                     I3REV=I2REV+I3-I2
                     TEMPR=DATA(I3)
                     TEMPI=DATA(I3+1)
                     DATA(I3)=DATA(I3REV)
                     DATA(I3+1)=DATA(I3REV+1)
                     DATA(I3REV)=TEMPR
                     DATA(I3REV+1)=TEMPI
 12               CONTINUE
 13            CONTINUE
            ENDIF
            IBIT=IP2/2
```

```
1              IF ((IBIT.GE.IP1).AND.(I2REV.GT.IBIT)) THEN
                  I2REV=I2REV-IBIT
                  IBIT=IBIT/2
                  GO TO 1
               ENDIF
               I2REV=I2REV+IBIT
14        CONTINUE
          IFP1=IP1
2         IF(IFP1.LT.IP2)THEN
              IFP2=2*IFP1
              THETA=ISIGN*6.28318530717959D0/(IFP2/IP1)
              WPR=-2.D0*DSIN(0.5D0*THETA)**2
              WPI=DSIN(THETA)
              WR=1.D0
              WI=0.D0
              DO 17 I3=1,IFP1,IP1
                 DO 16 I1=I3,I3+IP1-2,2
                    DO 15 I2=I1,IP3,IFP2
                       K1=I2
                       K2=K1+IFP1
                       TEMPR=SNGL(WR)*DATA(K2)-SNGL(WI)*DATA(K2+1)
                       TEMPI=SNGL(WR)*DATA(K2+1)+SNGL(WI)*DATA(K2)
                       DATA(K2)=DATA(K1)-TEMPR
                       DATA(K2+1)=DATA(K1+1)-TEMPI
                       DATA(K1)=DATA(K1)+TEMPR
                       DATA(K1+1)=DATA(K1+1)+TEMPI
15                  CONTINUE
16               CONTINUE
                 WTEMP=WR
                 WR=WR*WPR-WI*WPI+WR
                 WI=WI*WPR+WTEMP*WPI+WI
17            CONTINUE
              IFP1=IFP2
              GO TO 2
           ENDIF
           NPREV=N*NPREV
18      CONTINUE
        RETURN
        END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        SUBROUTINE TRED2(A,N,NP,D,E)
        IMPLICIT NONE
        INTEGER I,J,K,L,N,NP
        REAL A,D,E,H,SCALE,F,G,HH
        DIMENSION A(NP,NP),D(NP),E(NP)
        IF(N.GT.1)THEN
           DO 18 I=N,2,-1
              L=I-1
              H=0.
              SCALE=0.
              IF(L.GT.1)THEN
```

```
                DO 11 K=1,L
                    SCALE=SCALE+ABS(A(I,K))
11              CONTINUE
                IF(SCALE.EQ.0.)THEN
                    E(I)=A(I,L)
                ELSE
                    DO 12 K=1,L
                        A(I,K)=A(I,K)/SCALE
                        H=H+A(I,K)**2
12                  CONTINUE
                    F=A(I,L)
                    G=-SIGN(SQRT(H),F)
                    E(I)=SCALE*G
                    H=H-F*G
                    A(I,L)=F-G
                    F=0.
                    DO 15 J=1,L
                        A(J,I)=A(I,J)/H
                        G=0.
                        DO 13 K=1,J
                            G=G+A(J,K)*A(I,K)
13                      CONTINUE
                        IF(L.GT.J)THEN
                            DO 14 K=J+1,L
                                G=G+A(K,J)*A(I,K)
14                          CONTINUE
                        ENDIF
                        E(J)=G/H
                        F=F+E(J)*A(I,J)
15                  CONTINUE
                    HH=F/(H+H)
                    DO 17 J=1,L
                        F=A(I,J)
                        G=E(J)-HH*F
                        E(J)=G
                        DO 16 K=1,J
                            A(J,K)=A(J,K)-F*E(K)-G*A(I,K)
16                      CONTINUE
17                  CONTINUE
                ENDIF
            ELSE
                E(I)=A(I,L)
            ENDIF
            D(I)=H
18      CONTINUE
    ENDIF
    D(1)=0.
    E(1)=0.
    DO 23 I=1,N
        L=I-1
        IF(D(I).NE.0.)THEN
```

427

```
              DO 21 J=1,L
                  G=0.
                  DO 19 K=1,L
                      G=G+A(I,K)*A(K,J)
 19               CONTINUE
                  DO 20 K=1,L
                      A(K,J)=A(K,J)-G*A(K,I)
 20               CONTINUE
 21           CONTINUE
          ENDIF
          D(I)=A(I,I)
          A(I,I)=1.
          IF(L.GE.1)THEN
              DO 22 J=1,L
                  A(I,J)=0.
                  A(J,I)=0.
 22           CONTINUE
          ENDIF
 23   CONTINUE
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE TQLI(D,E,N,NP,Z)
      IMPLICIT NONE
      INTEGER N,NP,I,K,L,M,ITER
      REAL D,E,Z,DD,G,R,S,C,P,F,B
      DIMENSION D(NP),E(NP),Z(NP,NP)
      IF (N.GT.1) THEN
          DO 11 I=2,N
              E(I-1)=E(I)
 11       CONTINUE
          E(N)=0.
          DO 15 L=1,N
              ITER=0
 1            DO 12 M=L,N-1
                  DD=ABS(D(M))+ABS(D(M+1))
                  IF (ABS(E(M))+DD.EQ.DD) GO TO 2
 12           CONTINUE
              M=N
 2            IF(M.NE.L)THEN
                  IF(ITER.EQ.30) PAUSE 'too many iterations!'
                  ITER=ITER+1
                  G=(D(L+1)-D(L))/(2.*E(L))
                  R=SQRT(G**2+1.)
                  G=D(M)-D(L)+E(L)/(G+SIGN(R,G))
                  S=1.
                  C=1.
                  P=0.
                  DO 14 I=M-1,L,-1
                      F=S*E(I)
                      B=C*E(I)
```

428

```fortran
                      IF(ABS(F).GE.ABS(G))THEN
                          C=G/F
                          R=SQRT(C**2+1.)
                          E(I+1)=F*R
                          S=1./R
                          C=C*S
                      ELSE
                          S=F/G
                          R=SQRT(S**2+1.)
                          E(I+1)=G*R
                          C=1./R
                          S=S*C
                      ENDIF
                      G=D(I+1)-P
                      R=(D(I)-G)*S+2.*C*B
                      P=S*R
                      D(I+1)=G+P
                      G=C*R-B
                      DO 13 K=1,N
                          F=Z(K,I+1)
                          Z(K,I+1)=S*Z(K,I)+C*F
                          Z(K,I)=C*Z(K,I)-S*F
 13                   CONTINUE
 14                 CONTINUE
                    D(L)=D(L)-P
                    E(L)=G
                    E(M)=0.
                    GO TO 1
                ENDIF
 15       CONTINUE
        ENDIF
        RETURN
        END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        SUBROUTINE EIGSRT(D,V,N,NP)
        IMPLICIT NONE
        INTEGER N,NP,I,J,K
        REAL D,V,P
        DIMENSION D(NP),V(NP,NP)
        DO 13 I=1,N-1
          K=I
          P=D(I)
          DO 11 J=I+1,N
            IF(D(J).GE.P)THEN
              K=J
              P=D(J)
            ENDIF
 11       CONTINUE
          IF(K.NE.I)THEN
            D(K)=D(I)
            D(I)=P
```

429

```fortran
          DO 12 J=1,N
            P=V(J,I)
            V(J,I)=V(J,K)
            V(J,K)=P
12        CONTINUE
        ENDIF
13    CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE PIKSRT(N,ARR)
      IMPLICIT NONE
      INTEGER I,J,N
      REAL ARR,A
      DIMENSION ARR(N)
      DO 12 J=2,N
        A=ARR(J)
        DO 11 I=J-1,1,-1
          IF(ARR(I).LE.A)GO TO 10
          ARR(I+1)=ARR(I)
11      CONTINUE
        I=0
10      ARR(I+1)=A
12    CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

61.20 Problem OCT

Computational Problem OCT: Using the code for propagation a particle in a Morse potential (Problem 8), modify the code to write an optimal control loop to find an electric field that approximately evolves an initial state $\psi(t_i) = \phi_0$ into a state $\psi(t_f) = \phi_1$, where phi_0 and phi_1 are the ground and first excited states to the harmonic oscillator that approximates the Morse potential.

Solution to Problem on Optimal Control Theory:

In order to visualize the output of this program, cut the source code attached below save it in a file named OCT.f, compile it by typing

```
 gfortran OCT.f -o OCT
```

run it by typing

```
./OCT
```

Visualize the output as follows: type

```
gnuplot
```

then type

```
plot ''arch.0000''
```

That will show the representation of the initial state, in terms of an array of numbers associated with a grid in coordinate space. Then type

```
replot ''arch.0062''
```

, then

```
set yrange [0:1]
```

and finally

```
replot
```

to see the superposition of initial and final states. To exit, type

```
quit
```

The following code can be downloaded from here:

```
      PROGRAM OCT
c
c     1-D wave packet propagation on a Morse potential energy surface
c     Optimal Control Theory to drive excitation nu=0->1
c
      IMPLICIT NONE
      INTEGER nstep,NN,npts,nptx,ndump
```

```fortran
      INTEGER istep,jj,optiter
      REAL pi,dt,xc,pc,alpha,yield
      COMPLEX vprop,tprop,x_mean,p_mean
      character*9 Bfile
      PARAMETER(nstep=3142,npts=10,nptx=2**npts,NN=1)
      DIMENSION vprop(nptx,NN,NN),tprop(nptx)
      DIMENSION x_mean(NN),p_mean(NN)
      real efield(nstep), efieldnew(nstep)
      COMMON /class/ xc,pc
c
      pi=acos(-1.0)
      jj=0
      write(Bfile, '(A,i4.4)') 'traj.', jj
      OPEN(10,FILE=Bfile)
      CALL ReadParam(ndump,dt)
      call Initialize()
      call SetChi()
      call SetEF(dt,efield,nstep)
      CALL SetKinProp(dt,tprop)
      CALL SetPotProp(dt,vprop)
      call dumpEF(dt,efield,0,nstep)
c ___ run optimal control iterations
      alpha = 1.0
      call SetChi()
      DO istep=1,nstep
 CALL ONLYPROP(vprop,tprop,efield(istep),dt)
      end do
      call SetChi0()

      do optiter=1,15
 write(*,'(a,i4$)')'OCT iteration ',optiter
        DO istep=nstep,1,-1
    CALL BACKPROP(vprop,tprop,efield(istep),dt)
 end do
 call SetChi()
 DO istep=1,nstep
    CALL PROPAGATE(vprop,tprop,efield(istep),dt)
    efield(istep)=efield(istep)
     1          *( sin((pi*istep)/nstep)**2 )/alpha
 end do
 alpha=alpha*1.05
        call dumpEF(dt,efield,optiter,nstep)
        call SetChi0()
        write(*,'(a,e16.8)') ' yield before iteration: ',yield()
      end do
      write(*,*)'OCT done.'
c ___ run once to write observables
      call SetChi()
      CALL SAVEWF(0,ndump,dt)
      DO istep=1,nstep
         IF(mod(istep,ndump).EQ.0) then
```

432

```
            PRINT *, "Step ", istep," of ", nstep
            CALL SAVEWF(istep,ndump,dt)
         end if
         CALL ONLYPROP(vprop,tprop,efield(istep),dt)
         CALL XM(x_mean)
         CALL PM(p_mean)
         CALL VV(dt)
         WRITE(10,22) istep*dt,real(x_mean(1)),real(p_mean(1)),xc,pc
      END DO
      CLOSE(10)
      call SetChi0()
      write(*,'(a,e16.8)') ' yield: ', yield()
22    FORMAT(6(e13.6,2x))
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine ReadParam(ndump,dt)
c
c     Parameters defining the grid (xmin, xmax), integration time step (dt),
c     rmass (rmass), initial position (xk), initial momentum (pk),
c     number of propagation steps (nstep), and how often to save a pic (ndump)
c
      IMPLICIT NONE
      INTEGER ntype,nrpt,ireport,ndump,nlit
      REAL xmin,xmax,pk,rmass,xk,dt
      common /packet/  rmass,xk,pk
      common /xy/  xmin,xmax
c
      xmin=-5.0
      xmax=20.0
      dt=0.2
      rmass=1.0
      xk=0.0
      pk=0.0
      ndump=50
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE VV(dt)
c
c     Velocity Verlet Algorithm J. Chem. Phys. 76, 637 (1982)
c
      IMPLICIT NONE
      REAL v,dx,dt,xc,pc,rmass,xk,pk,acc,xt,VPOT1,VPOT2,F
      COMMON /class/ xc,pc
      common /packet/ rmass,xk,pk
c
c     Compute Force
c
      dx=0.01
      xt=xc+dx
```

```
         CALL VA(VPOT1,xt)
         xt=xc-dx
         CALL VA(VPOT2,xt)
         F=-(VPOT1-VPOT2)/(2.0*dx)
         v=pc/rmass
c
c        Advance momenta half a step
c
         pc=pc+0.5*F*dt
c
c        Advance coordinates a step
c
         xc=xc+v*dt+0.5*dt**2*F/rmass
c
c        Compute Force
c
         dx=0.01
         xt=xc+dx
         CALL VA(VPOT1,xt)
         xt=xc-dx
         CALL VA(VPOT2,xt)
         F=-(VPOT1-VPOT2)/(2.0*dx)
c
c        Advance momenta half a step
c
         pc=pc+0.5*F*dt
c
         return
         end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
         SUBROUTINE Initialize()

         IMPLICIT NONE
         INTEGER NN,nptx,npts,kk
         COMPLEX EYE,CRV,chi0,chi,states
         REAL xc,pc,omega,xk2,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha,alpha2
         PARAMETER(npts=10,nptx=2**npts,NN=1)
         DIMENSION CRV(NN,NN)
         common /xy/   xmin,xmax
         common /packet/ rmass,xk,pk
         COMMON /states/ states(nptx,2)
         COMMON /wfunc/  chi(nptx,NN)
         COMMON /iwfunc/ chi0(nptx,NN)
         COMMON /class/ xc,pc

         EYE=(0.0,1.0)
         pi= acos(-1.0)
         omega=1.
         dx=(xmax-xmin)/real(nptx)
         xc=xk
         pc=pk
```

434

```fortran
c
c      Wave Packet Initialization: Gaussian centered at xk, with momentum pk
c
       alpha=rmass*omega
       do kk=1,nptx
          x=xmin+kk*dx
          states(kk,1)=((alpha/pi)**0.25)
     1          *exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
          states(kk,2)=sqrt(2.0)*(x-xk)*states(kk,1)
       end do
       RETURN
       END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       real function yield()
       IMPLICIT NONE
       INTEGER NN,nptx,npts,kk
       COMPLEX chi0,chi,states,collect
       REAL dx,xmin,xmax
       PARAMETER(npts=10,nptx=2**npts,NN=1)
       common /xy/  xmin,xmax
       COMMON /wfunc/  chi(nptx,NN)
       COMMON /iwfunc/ chi0(nptx,NN)
       COMMON /states/ states(nptx,2)

       dx=(xmax-xmin)/real(nptx)
       collect=(0.0, 0.0)
       do kk=1,nptx
          collect=collect+conjg(states(kk,2))*chi(kk,1)
       end do
       yield=abs(collect*dx)**2

       RETURN
       END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       SUBROUTINE SetEF(dt,efield,nstep)
       IMPLICIT NONE
       integer ii,nstep
       real dt,pi,omega,AA
       real efield(nstep)

       pi=acos(-1.0)
       omega=1.0
       AA=1.0e-2
c
c      Field initialization
c
       do ii=1,nstep
          efield(ii)=AA*( sin((pi*ii)/real(nstep))**2 )*sin(omega*ii*dt)
       end do
c       write(*,*)"norm: ", norm*(xmax-xmin)/real(nptx)
```

```fortran
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE dumpEF(dt,efield,optiter,nstep)
      IMPLICIT NONE
      integer ii,optiter,nstep
      real dt,efield(nstep)
      character*32 BB

      write(BB, '(A,i4.4)') 'ef.', optiter
      OPEN(1,FILE=BB)

      do ii=1,nstep
         write(1,'(2(e16.8))') ii*dt, efield(ii)
      end do

      close(1)

      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE SetChi()
      IMPLICIT NONE
      INTEGER NN,nptx,npts,kk
      COMPLEX chi0,chi,states
      REAL norm,xmin,xmax
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      common /xy/  xmin,xmax
      COMMON /states/ states(nptx,2)
      COMMON /wfunc/  chi(nptx,NN)
      COMMON /iwfunc/ chi0(nptx,NN)


c
c     Wave Packet Initialization: Gaussian centered at xk, with momentum pk
c
      norm=0.0
      do kk=1,nptx
         chi(kk,1)=states(kk,1)
         norm=norm+real(conjg(chi(kk,1))*chi(kk,1))
      end do
c     write(*,*)"norm: ", norm*(xmax-xmin)/real(nptx)

      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE SetChi0()
      IMPLICIT NONE
      INTEGER NN,nptx,npts,kk
      COMPLEX chi0,chi,states,collect
      REAL norm,dx,xmin,xmax
      PARAMETER(npts=10,nptx=2**npts,NN=1)
```

436

```
      common /xy/   xmin,xmax
      COMMON /states/ states(nptx,2)
      COMMON /wfunc/   chi(nptx,NN)
      COMMON /iwfunc/ chi0(nptx,NN)

c
c     Wave Packet Initialization: Gaussian centered at xk, with momentum pk
c
      dx=(xmax-xmin)/nptx
      norm=0.0
      collect=(0.0, 0.0)
      do kk=1,nptx
         collect=collect+conjg(states(kk,2))*chi(kk,1)
      end do
      collect=collect*dx
      do kk=1,nptx
         chi0(kk,1)=states(kk,2)*collect
         norm=norm+real(conjg(chi(kk,1))*chi(kk,1))
      end do
c      write(*,*)"norm: ", norm*(xmax-xmin)/real(nptx)


      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE HAMIL(CRV,x)
c
c     Hamiltonian Matrix
c
      IMPLICIT NONE
      INTEGER NN
      REAL x,VPOT1
      COMPLEX CRV
      PARAMETER(NN=1)
      DIMENSION CRV(NN,NN)
c
      CALL VA(VPOT1,x)
      CRV(1,1)=VPOT1
c
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE VA(V,x)
c
c     Potential Energy Surface: Morse Potential [Phys. Rev. (1929) 34:57]
c
      implicit none
      REAL V,x,rmass,xk,pk,rk,omega,De,xeq,a
      common /packet/ rmass,xk,pk
      xeq=0.0
      omega=1.0
```

```
      De=8.0
      rk=rmass*omega**2
      a=sqrt(rk/(2.0*De))
      V=De*(1.0-exp(-a*(x-xeq)))**2
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      real function dipole(xx)
      implicit none
      real xx

      dipole = 1.0 * xx

      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetKinProp(dt,tprop)
c
c     Kinetic Energy part of the Trotter Expansion: exp(-i p^2 dt/(2 m))
c
      IMPLICIT NONE
      INTEGER nptx,kx,nx,npts
      REAL xsc,xmin,xmax,propfacx,rmass,xk,pi,alenx,dt,pk
      COMPLEX tprop,eye
      parameter(npts=10,nptx=2**npts)
      DIMENSION tprop(nptx)
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
c
      eye=(0.,1.)
      pi = acos(-1.0)
      alenx=xmax-xmin
      propfacx=-dt/2./rmass*(2.*pi)**2
      do kx=1,nptx
         if(kx.le.(nptx/2+1)) then
            nx=kx-1
         else
            nx=kx-1-nptx
         end if
         xsc=0.
         if(nx.ne.0) xsc=real(nx)/alenx
         tprop(kx)=exp(eye*(propfacx*xsc**2))/dble(nptx)
       end do
c
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetPotProp(dt,vprop)
c
c     Potential Energy part of the Trotter Expansion: exp(-i V dt/2)
c
```

```
      IMPLICIT NONE
      INTEGER NN,ii,nptx,npts
      REAL xmin,xmax,dx,dt,x,VPOT
      COMPLEX vprop,eye
      parameter(npts=10,nptx=2**npts,NN=1)
      DIMENSION vprop(nptx,NN,NN)
      common /xy/  xmin,xmax
      eye=(0.,1.)
      dx=(xmax-xmin)/real(nptx)
c
      do ii=1,nptx
         x=xmin+ii*dx
         CALL VA(VPOT,x)
         vprop(ii,1,1)=exp(-eye*0.5*dt*VPOT)
      END DO
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE energies(energy)
      IMPLICIT NONE
      INTEGER j,NN
      COMPLEX energy,RV,RKE
      PARAMETER (NN=1)
      DIMENSION RV(NN),RKE(NN),energy(NN)
      CALL PE(RV)
      CALL KE(RKE)
      DO j=1,NN
         energy(j)=RV(j)+RKE(j)
      END DO
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE SAVEWF(je2,ndump,dt)
c
c     Dump Time Evolved Wave packet
c
      IMPLICIT NONE
      INTEGER je2,nptx,npts,kk,NN,ncount,ndump,jj
      COMPLEX chi,CRV,energy,psi,Psia
      character*9 B
      REAL V,x1,c1,c2,c1a,x,xmin,xmax,dx,EVALUES,dt,cutoff
      PARAMETER(npts=10,nptx=2**npts,NN=1,cutoff=1.0E-5)
      DIMENSION CRV(NN,NN),EVALUES(NN)
      DIMENSION psi(NN,NN)
      common /xy/  xmin,xmax
      COMMON /wfunc/ chi(nptx,NN)
      COMMON /ENER/ energy(NN)
c
      IF(je2.EQ.1) CALL energies(energy)
      jj=je2/ndump
      write(B, '(A,i4.4)') 'arch.', jj
```

```
         OPEN(1,FILE=B)
         dx=(xmax-xmin)/real(nptx)
         ncount=(je2-1)/ndump
c
c        Save Wave-packet components
c
         do kk=1,nptx
            x=xmin+kk*dx
            c1=chi(kk,1)*conjg(chi(kk,1))
            write(1,33) x,max(sqrt(c1)+real(energy(1)),cutoff)
         end do
         write(1,33)
         write(1,33)
         do kk=1,nptx
            x=xmin+kk*dx
            write(1,33) x,real(energy(1))
         end do
         write(1,33)
         write(1,33)
c
c        Save Adiabatic states
c
         do kk=1,nptx
            x=xmin+kk*dx
            CALL HAMIL(CRV,x)
            write(1,33) x,CRV(1,1)
         end do
         CLOSE(1)
 33      format(6(e13.6,2x))
         RETURN
         END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
         SUBROUTINE XM(RV)
c
c        Expectation Value of the Position
c
         IMPLICIT NONE
         INTEGER nptx,npts,kk,NN,j
         COMPLEX chi,EYE,RV
         REAL Vpot,omega,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha
         PARAMETER(npts=10,nptx=2**npts,NN=1)
         DIMENSION RV(NN)
         COMMON /wfunc/ chi(nptx,NN)
         common /xy/   xmin,xmax
         common /packet/rmass,xk,pk

         dx=(xmax-xmin)/real(nptx)
         DO j=1,NN
            RV(j)=0.0
            do kk=1,nptx
               x=xmin+kk*dx
```

440

```
          IF(j.EQ.1) CALL VA(Vpot,x)
          RV(j)=RV(j)+chi(kk,j)*x*conjg(chi(kk,j))*dx
        end do
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PE(RV)
c
c     Expectation Value of the Potential Enegy
c
      IMPLICIT NONE
      INTEGER nptx,npts,kk,NN,j
      COMPLEX chi,EYE,RV
      REAL Vpot,omega,xmin,xmax,dx,pi,rmass,xk,pk,x,alpha
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      DIMENSION RV(NN)
      COMMON /wfunc/ chi(nptx,NN)
      common /xy/  xmin,xmax
      common /packet/rmass,xk,pk

      dx=(xmax-xmin)/real(nptx)
      DO j=1,NN
         RV(j)=0.0
         do kk=1,nptx
            x=xmin+kk*dx
            IF(j.EQ.1) CALL VA(Vpot,x)
            RV(j)=RV(j)+chi(kk,j)*Vpot*conjg(chi(kk,j))*dx
         end do
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine KE(RKE)
c
c     Expectation value of the kinetic energy
c
      IMPLICIT NONE
      INTEGER NN,kk,nptx,kx,nx,npts,j
      REAL dp,theta,wm,p,xmin,xmax,rmass,xk,pi,alenx,pk,rm,re,ri,dx
      COMPLEX eye,chi,Psip,chic,RKE
      parameter(npts=10,nptx=2**npts,NN=1)
      DIMENSION chic(nptx),RKE(NN)
      common /xy/  xmin,xmax
      common /packet/  rmass,xk,pk
      COMMON /wfunc/ chi(nptx,NN)
c
      pi = acos(-1.0)
      dx=(xmax-xmin)/nptx
      dp=2.*pi/(xmax-xmin)
c
```

441

```fortran
      DO j=1,NN
         RKE(j)=0.0
         do kk=1,nptx
            chic(kk)=chi(kk,j)
         end do
         CALL fourn(chic,nptx,1,-1)
         do kx=1,nptx
            if(kx.le.(nptx/2+1)) then
               nx=kx-1
            else
               nx=kx-1-nptx
            end if
            p=0.
            if(nx.ne.0) p = real(nx)*dp
            chic(kx)=p**2/(2.0*rmass)*chic(kx)/nptx
         end do
         CALL fourn(chic,nptx,1,1)
         do kk=1,nptx
            RKE(j)=RKE(j)+conjg(chi(kk,j))*chic(kk)*dx
         end do
      END DO
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine PM(RKE)
c
c     Expectation value of the kinetic energy
c
      IMPLICIT NONE
      INTEGER NN,kk,nptx,kx,nx,npts,j
      REAL dp,theta,wm,p,xmin,xmax,rmass,xk,pi,alenx,pk,rm,re,ri,dx
      COMPLEX eye,chi,Psip,chic,RKE
      parameter(npts=10,nptx=2**npts,NN=1)
      DIMENSION chic(nptx),RKE(NN)
      common /xy/   xmin,xmax
      common /packet/  rmass,xk,pk
      COMMON /wfunc/ chi(nptx,NN)
c
      pi = acos(-1.0)
      dx=(xmax-xmin)/nptx
      dp=2.*pi/(xmax-xmin)
c
      DO j=1,NN
         RKE(j)=0.0
         do kk=1,nptx
            chic(kk)=chi(kk,j)
         end do
         CALL fourn(chic,nptx,1,-1)
         do kx=1,nptx
            if(kx.le.(nptx/2+1)) then
               nx=kx-1
```

```
            else
                nx=kx-1-nptx
            end if
            p=0.
            if(nx.ne.0) p = real(nx)*dp
            chic(kx)=p*chic(kx)/nptx
        end do
        CALL fourn(chic,nptx,1,1)
        do kk=1,nptx
            RKE(j)=RKE(j)+conjg(chi(kk,j))*chic(kk)*dx
        end do
      END DO
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE ONLYPROP(vprop,tprop,ef,dt)
c
c     Split Operator Fourier Transform Propagation Method
c     J. Comput. Phys. 47, 412 (1982); J. Chem. Phys. 78, 301 (1983)
c
      IMPLICIT NONE
      INTEGER i,j,NN,ii,nptx,npts
      COMPLEX eye,chi,chi0,vprop,chin1,chin2,tprop,collect1,collect2
      real ef,dt,dx,xx,xmin,xmax
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      parameter (eye=(0.0,1.0))
      DIMENSION chin1(nptx),chin2(nptx)
      DIMENSION tprop(nptx),vprop(nptx,NN,NN)
      common /xy/  xmin,xmax
      COMMON /wfunc/ chi(nptx,NN)
      COMMON /iwfunc/ chi0(nptx,NN)

      real dipole

      dx = (xmax - xmin)/nptx
c
c     Apply potential energy part of the Trotter Expansion
c
      DO i=1,nptx
         xx = xmin + i*dx
         chin1(i)=exp(eye*dipole(xx)*ef*dt/2)*
     1      vprop(i,1,1)*chi(i,1)
      END DO
c
c     Fourier Transform wave-packet to the momentum representation
c
      CALL fourn(chin1,nptx,1,-1)
c
c     Apply kinetic energy part of the Trotter Expansion
c
      DO i=1,nptx
```

443

```fortran
            chin1(i)=tprop(i)*chin1(i)
         END DO
c
c     Inverse Fourier Transform wave-packet to the coordinate representation
c
         CALL fourn(chin1,nptx,1,1)
c
c     Apply potential energy part of the Trotter Expansion
c
         DO i=1,nptx
            xx = xmin + i*dx
            chi(i,1)=exp(eye*dipole(xx)*ef*dt/2)*
      1        vprop(i,1,1)*chin1(i)
         END DO

         END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
         SUBROUTINE BACKPROP(vprop,tprop,ef,dt)
c
c     Split Operator Fourier Transform Propagation Method
c     J. Comput. Phys. 47, 412 (1982); J. Chem. Phys. 78, 301 (1983)
c
         IMPLICIT NONE
         INTEGER i,j,NN,ii,nptx,npts
         COMPLEX eye,chi0,vprop,chin1,chin2,tprop
         real ef,dt,dx,xx,xmin,xmax
         PARAMETER(npts=10,nptx=2**npts,NN=1)
         parameter (eye=(0.0,1.0))
         DIMENSION chin1(nptx),chin2(nptx)
         DIMENSION tprop(nptx),vprop(nptx,NN,NN)
         common /xy/  xmin,xmax
         COMMON /iwfunc/ chi0(nptx,NN)

         real dipole

         dx = (xmax - xmin)/nptx
c
c     Apply potential energy part of the Trotter Expansion
c
c ___ linear field term assumes linear dipole moment
         DO i=1,nptx
            xx = xmin + i*dx
            chin1(i)=exp(-eye*dipole(xx)*ef*dt/2)*
      1        conjg(vprop(i,1,1))*chi0(i,1)
         END DO
c
c     Fourier Transform wave-packet to the momentum representation
c
         CALL fourn(chin1,nptx,1,-1)
c
c     Apply kinetic energy part of the Trotter Expansion
```

444

```
c
      DO i=1,nptx
         chin1(i)=conjg(tprop(i))*chin1(i)
      END DO
c
c     Inverse Fourier Transform wave-packet to the coordinate representation
c
      CALL fourn(chin1,nptx,1,1)
c
c     Apply potential energy part of the Trotter Expansion
c
      DO i=1,nptx
         xx = xmin + i*dx
         chi0(i,1)=exp(-eye*dipole(xx)*ef*dt/2)*
     1      conjg(vprop(i,1,1))*chin1(i)
      END DO
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PROPAGATE(vprop,tprop,ef,dt)
c
c     Split Operator Fourier Transform Propagation Method
c     J. Comput. Phys. 47, 412 (1982); J. Chem. Phys. 78, 301 (1983)
c
      IMPLICIT NONE
      INTEGER i,j,NN,ii,nptx,npts
      COMPLEX eye,chi,chi0,vprop,chin1,chin2,tprop,collect1,collect2
      real ef,dt,dx,xx,xmin,xmax
      PARAMETER(npts=10,nptx=2**npts,NN=1)
      parameter (eye=(0.0,1.0))
      DIMENSION chin1(nptx),chin2(nptx)
      DIMENSION tprop(nptx),vprop(nptx,NN,NN)
      common /xy/  xmin,xmax
      COMMON /wfunc/ chi(nptx,NN)
      COMMON /iwfunc/ chi0(nptx,NN)

      real dipole

      dx = (xmax - xmin)/nptx
c
c     Apply potential energy part of the Trotter Expansion
c
      DO i=1,nptx
         xx = xmin + i*dx
         chin1(i)=exp(eye*dipole(xx)*ef*dt/2)*
     1      vprop(i,1,1)*chi(i,1)
      END DO
c
c     Fourier Transform wave-packet to the momentum representation
c
      CALL fourn(chin1,nptx,1,-1)
c
```

445

```fortran
c     Apply kinetic energy part of the Trotter Expansion
c
      DO i=1,nptx
         chin1(i)=tprop(i)*chin1(i)
      END DO
c
c     Inverse Fourier Transform wave-packet to the coordinate representation
c
      CALL fourn(chin1,nptx,1,1)
c
c     Apply potential energy part of the Trotter Expansion
c
      DO i=1,nptx
         xx = xmin + i*dx
         chi(i,1)=exp(eye*dipole(xx)*ef*dt/2)*
     1      vprop(i,1,1)*chin1(i)
      END DO

c ___ now the same thing for the Lagrange multiplier

c
c     Apply potential energy part of the Trotter Expansion
c
      DO i=1,nptx
         xx = xmin + i*dx
         chin1(i)=exp(eye*dipole(xx)*ef*dt/2)*
     1      vprop(i,1,1)*chi0(i,1)
      END DO
c
c     Fourier Transform wave-packet to the momentum representation
c
      CALL fourn(chin1,nptx,1,-1)
c
c     Apply kinetic energy part of the Trotter Expansion
c
      DO i=1,nptx
         chin1(i)=tprop(i)*chin1(i)
      END DO
c
c     Inverse Fourier Transform wave-packet to the coordinate representation
c
      CALL fourn(chin1,nptx,1,1)
c
c     Apply potential energy part of the Trotter Expansion
c
      DO i=1,nptx
         xx = xmin + i*dx
         chi0(i,1)=exp(eye*dipole(xx)*ef*dt/2)*
     1      vprop(i,1,1)*chin1(i)
      END DO
```

```fortran
c ___ evaluate overlap integral for new field

      collect1=(0.0, 0.0)
      collect2=(0.0, 0.0)
      DO i=1,nptx
         xx = xmin + i*dx
c         collect1=collect1+conjg(chi(i,1))*chi0(i,1)
         collect2=collect2+conjg(chi0(i,1))*dipole(xx)*chi(i,1)
      END DO
      ef=-aimag(collect2*dx)


      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutine for FFT from Numerical Recipes
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE FOURN(DATA,NN,NDIM,ISIGN)
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION NN(NDIM),DATA(*)
      NTOT=1
      DO 11 IDIM=1,NDIM
         NTOT=NTOT*NN(IDIM)
 11   CONTINUE
      NPREV=1
      DO 18 IDIM=1,NDIM
         N=NN(IDIM)
         NREM=NTOT/(N*NPREV)
         IP1=2*NPREV
         IP2=IP1*N
         IP3=IP2*NREM
         I2REV=1
         DO 14 I2=1,IP2,IP1
            IF(I2.LT.I2REV)THEN
               DO 13 I1=I2,I2+IP1-2,2
                  DO 12 I3=I1,IP3,IP2
                     I3REV=I2REV+I3-I2
                     TEMPR=DATA(I3)
                     TEMPI=DATA(I3+1)
                     DATA(I3)=DATA(I3REV)
                     DATA(I3+1)=DATA(I3REV+1)
                     DATA(I3REV)=TEMPR
                     DATA(I3REV+1)=TEMPI
 12               CONTINUE
 13            CONTINUE
            ENDIF
            IBIT=IP2/2
 1          IF ((IBIT.GE.IP1).AND.(I2REV.GT.IBIT)) THEN
               I2REV=I2REV-IBIT
               IBIT=IBIT/2
               GO TO 1
            ENDIF
            I2REV=I2REV+IBIT
```

```
14        CONTINUE
          IFP1=IP1
2         IF(IFP1.LT.IP2)THEN
              IFP2=2*IFP1
              THETA=ISIGN*6.28318530717959D0/(IFP2/IP1)
              WPR=-2.D0*DSIN(0.5D0*THETA)**2
              WPI=DSIN(THETA)
              WR=1.D0
              WI=0.D0
              DO 17 I3=1,IFP1,IP1
                  DO 16 I1=I3,I3+IP1-2,2
                      DO 15 I2=I1,IP3,IFP2
                          K1=I2
                          K2=K1+IFP1
                          TEMPR=SNGL(WR)*DATA(K2)-SNGL(WI)*DATA(K2+1)
                          TEMPI=SNGL(WR)*DATA(K2+1)+SNGL(WI)*DATA(K2)
                          DATA(K2)=DATA(K1)-TEMPR
                          DATA(K2+1)=DATA(K1+1)-TEMPI
                          DATA(K1)=DATA(K1)+TEMPR
                          DATA(K1+1)=DATA(K1+1)+TEMPI
15                    CONTINUE
16                CONTINUE
                  WTEMP=WR
                  WR=WR*WPR-WI*WPI+WR
                  WI=WI*WPR+WTEMP*WPI+WI
17            CONTINUE
              IFP1=IFP2
              GO TO 2
          ENDIF
          NPREV=N*NPREV
18    CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

61.21 Problem FGH

Computational Problem FGH:

Write a program to solve the time independent Schrödinger equation by using the FGH method and apply it to find the first 4 eigenvalues and eigenfunctions of the Harmonic oscillator introduced by Eq. (17) with $m = 1$ and $\omega = 1$. Verify that the eigenvalues are $E(\nu) = (1/2 + \nu)\hbar\omega$, $\nu = 0$–10.

Download the source code from (http://ursula.chem.yale.edu/~batista/classes/v572/FGH.f)

```
      PROGRAM FGH
c
c     This code computes the eigenvalues and eigenvectors of a Harmonic
c     oscillator V(x)=0.5*m*w**2*(x-2.)**2
c     The KE matrix is described according to Eq. (321) of the lecture notes
c     that corresponds to Eq.(21) of JCP (1989) 91:3571-3576.
c
      CALL READPARAM()
      CALL FGH_Ham()
      CALL EIGV()
      CALL DUMP()
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE READPARAM()
      implicit real *8(a-h,o-z)
      PARAMETER(npt=64,NC=1)
      COMMON /paque/ rmin(NC), rmax(NC), rmass(NC), dx(NC)
      rmax(1)=8.
      rmin(1)=-rmax(1)
      rmass(1)=1.
      DO I=1,NC
         dx(I)=(rmax(I)-rmin(I)) / npt
      END DO
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE FGH_Ham()
      implicit real *8(a-h,o-z)
      PARAMETER(npt=64,NC=1)
      COMMON /paque/ rmin(NC), rmax(NC), rmass(NC), dx(NC)
      COMMON/HAM/ VHAM(npt,npt)
      pi=acos(-1.)
      dp=2.0*pi/(rmax(1)-rmin(1))
      DO i=1,npt
         r=(i-npt/2)*dx(1)
         DO ip=1, npt
            rp=(ip-npt/2)*dx(1)
            VHAM(i,ip) = 0.0
            IF(ip.EQ.i) VHAM(i,ip)=VHAM(i,ip)+V(r)
            DO k=1,npt/2
               pk=dp*(k-npt/2)
               VHAM(i,ip)=VHAM(i,ip)+2.0*cos((rp-r)*pk)*
```

449

```
     1               pk**2/(2.0*rmass(1))/npt
           END DO
         END DO
       END DO
       RETURN
       END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       FUNCTION V(r)
       implicit real *8(a-h,o-z)
       x0=0.0
       V=0.5*1.*(r-x0)**2
       RETURN
       END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       SUBROUTINE DVR_Ham()
       implicit real *8(a-h,o-z)
      PARAMETER(npt=64,NC=1)
       COMMON /paque/ rmin(NC), rmax(NC), rmass(NC), dx(NC)
       COMMON/HAM/ VHAM(npt,npt)
       pi=acos(-1.)
       DO i=1, npt
          DO ip=1, npt
             IF(i.EQ.ip) THEN
               VTEMP=pi*pi/3.0d0
c     VTEMP=VTEMP-0.5d0/dfloat(i)**2 ! for radial coord.
             ELSE
               VTEMP=2.d0/(i-ip)**2
c     VTEMP=VTEMP-2.d0/(i+ip)**2     ! for radial coord.
             END IF
             VHAM(i,ip)=VTEMP*(-1)**(i-ip)
     1          /dx(1)/ dx(1)/ (2.0*rmass(1))
             IF(i.EQ.ip) THEN
                r=rmin(1)+(i-1)*dx(1)
                VHAM(i,ip)=VHAM(i,ip)+V(r)
             END IF
          END DO
       END DO
       RETURN
       END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       SUBROUTINE EIGV()
c
c     Diagonalization
c
c     VHAM: HERMITIAN MATRIX (INPUT)
c     EVALUES: EIGENVALUES (OUTPUT)
c     EVECT: EIGENVECTORS (OUTPUT)
c
       implicit real *8(a-h,o-z)
       REAL CRV, EVALUES,E,EVECT
       PARAMETER(npt=32,NC=1,npt2=npt**NC)
```

450

```
      COMMON/ HAM/ VHAM(npt2,npt2)
      COMMON /paque/ rmin(NC), rmax(NC), rmass(NC), dx(NC)
      PARAMETER(N=npt2,NP=N)
      COMMON/EIG/ EVALUES(npt2),EVECT(npt2,npt2)
      DIMENSION E(NP)
C
      DO I=1,N
         EVALUES(I)=0.0
         E(I)=0.0
         DO J=1,N
            EVECT(J,I)=VHAM(J,I)
         END DO
      END DO
      CALL TRED22(EVECT,N,NP,EVALUES,E)
      CALL TQLI(EVALUES,E,N,NP,EVECT)
      CALL EIGSRT(EVALUES,EVECT,N,NP)
C
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE DUMP()
C
      implicit real *8(a-h,o-z)
   REAL CRV, EVALUES,EVECT,E
      character*9 B
      PARAMETER(npt=64,NC=1,npt2=npt**NC)
      PARAMETER(N=npt2,NP=N)
      COMMON/EIG/ EVALUES(npt2),EVECT(npt2,npt2)
      COMMON /paque/ rmin(NC), rmax(NC), rmass(NC), dx(NC)
      DIMENSION r(NC),j(NC)
C
      DO k=1, 10
         IND=npt2-(k-1)
         PRINT *, "E(",k,")=",EVALUES(IND)
      END DO
C
      DO l=1,4
      IND=npt2-(l-1)
      write(B, '(A,i4.4)') 'wave.', l
      OPEN(10,FILE=B)
      rsum=0.0
      DO i=1,npt2
         r(1)=(i-npt2/2)*dx(1)
         WRITE(10,22) r(1),V(r),EVALUES(IND)
    1        ,EVALUES(IND)+EVECT(i,IND)
         rsum=rsum + EVECT(i,IND)**2
      END DO
      PRINT *, "norm(",l,")=",rsum
      END DO
      CLOSE(10)
 22   FORMAT(6(e13.6,2x))
```

451

```
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c      Subroutines to compute eigenvalues and eigenvectors from NR
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE TRED22(A,N,NP,D,E)
      IMPLICIT NONE
      INTEGER I,J,K,L,N,NP
      REAL A,D,E,H,SCALE,F,G,HH
      DIMENSION A(NP,NP),D(NP),E(NP)
      IF(N.GT.1)THEN
         DO 18 I=N,2,-1
            L=I-1
            H=0.
            SCALE=0.
            IF(L.GT.1)THEN
               DO 11 K=1,L
                  SCALE=SCALE+ABS(A(I,K))
 11            CONTINUE
               IF(SCALE.EQ.0.)THEN
                  E(I)=A(I,L)
               ELSE
                  DO 12 K=1,L
                     A(I,K)=A(I,K)/SCALE
                     H=H+A(I,K)**2
 12               CONTINUE
                  F=A(I,L)
                  G=-SIGN(SQRT(H),F)
                  E(I)=SCALE*G
                  H=H-F*G
                  A(I,L)=F-G
                  F=0.
                  DO 15 J=1,L
                     A(J,I)=A(I,J)/H
                     G=0.
                     DO 13 K=1,J
                        G=G+A(J,K)*A(I,K)
 13                  CONTINUE
                     IF(L.GT.J)THEN
                        DO 14 K=J+1,L
                           G=G+A(K,J)*A(I,K)
 14                     CONTINUE
                     ENDIF
                     E(J)=G/H
                     F=F+E(J)*A(I,J)
 15               CONTINUE
                  HH=F/(H+H)
                  DO 17 J=1,L
                     F=A(I,J)
                     G=E(J)-HH*F
                     E(J)=G
```

```fortran
                        DO 16 K=1,J
                            A(J,K)=A(J,K)-F*E(K)-G*A(I,K)
16                      CONTINUE
17                  CONTINUE
                ENDIF
            ELSE
                E(I)=A(I,L)
            ENDIF
            D(I)=H
18      CONTINUE
      ENDIF
      D(1)=0.
      E(1)=0.
      DO 23 I=1,N
          L=I-1
          IF(D(I).NE.0.)THEN
              DO 21 J=1,L
                  G=0.
                  DO 19 K=1,L
                      G=G+A(I,K)*A(K,J)
19                CONTINUE
                  DO 20 K=1,L
                      A(K,J)=A(K,J)-G*A(K,I)
20                CONTINUE
21            CONTINUE
          ENDIF
          D(I)=A(I,I)
          A(I,I)=1.
          IF(L.GE.1)THEN
              DO 22 J=1,L
                  A(I,J)=0.
                  A(J,I)=0.
22            CONTINUE
          ENDIF
23    CONTINUE
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE TQLI(D,E,N,NP,Z)
      IMPLICIT NONE
      INTEGER N,NP,I,K,L,M,ITER
      REAL D,E,Z,DD,G,R,S,C,P,F,B
      DIMENSION D(NP),E(NP),Z(NP,NP)
      IF (N.GT.1) THEN
          DO 11 I=2,N
              E(I-1)=E(I)
11        CONTINUE
          E(N)=0.
          DO 15 L=1,N
              ITER=0
1             DO 12 M=L,N-1
```

453

```
                    DD=ABS(D(M))+ABS(D(M+1))
                    IF (ABS(E(M))+DD.EQ.DD) GO TO 2
12              CONTINUE
                M=N
2               IF(M.NE.L)THEN
                    IF(ITER.EQ.30) PAUSE 'too many iterations!'
                    ITER=ITER+1
                    G=(D(L+1)-D(L))/(2.*E(L))
                    R=SQRT(G**2+1.)
                    G=D(M)-D(L)+E(L)/(G+SIGN(R,G))
                    S=1.
                    C=1.
                    P=0.
                    DO 14 I=M-1,L,-1
                        F=S*E(I)
                        B=C*E(I)
                        IF(ABS(F).GE.ABS(G))THEN
                            C=G/F
                            R=SQRT(C**2+1.)
                            E(I+1)=F*R
                            S=1./R
                            C=C*S
                        ELSE
                            S=F/G
                            R=SQRT(S**2+1.)
                            E(I+1)=G*R
                            C=1./R
                            S=S*C
                        ENDIF
                        G=D(I+1)-P
                        R=(D(I)-G)*S+2.*C*B
                        P=S*R
                        D(I+1)=G+P
                        G=C*R-B
                        DO 13 K=1,N
                            F=Z(K,I+1)
                            Z(K,I+1)=S*Z(K,I)+C*F
                            Z(K,I)=C*Z(K,I)-S*F
13                      CONTINUE
14                  CONTINUE
                    D(L)=D(L)-P
                    E(L)=G
                    E(M)=0.
                    GO TO 1
                ENDIF
15          CONTINUE
        ENDIF
        RETURN
        END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        SUBROUTINE EIGSRT(D,V,N,NP)
```

```
      IMPLICIT NONE
      INTEGER N,NP,I,J,K
      REAL D,V,P
      DIMENSION D(NP),V(NP,NP)
      DO 13 I=1,N-1
        K=I
        P=D(I)
        DO 11 J=I+1,N
          IF(D(J).GE.P)THEN
            K=J
            P=D(J)
          ENDIF
11      CONTINUE
        IF(K.NE.I)THEN
          D(K)=D(I)
          D(I)=P
          DO 12 J=1,N
            P=V(J,I)
            V(J,I)=V(J,K)
            V(J,K)=P
12        CONTINUE
        ENDIF
13    CONTINUE
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PIKSRT(N,ARR)
      IMPLICIT NONE
      INTEGER I,J,N
      REAL ARR,A
      DIMENSION ARR(N)
      DO 12 J=2,N
        A=ARR(J)
        DO 11 I=J-1,1,-1
          IF(ARR(I).LE.A)GO TO 10
          ARR(I+1)=ARR(I)
11      CONTINUE
        I=0
10      ARR(I+1)=A
12    CONTINUE
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

To visualize the output of the program listed above, save it in a file named FGH.f, compile it by typing

```
 gfortran FGH.f -o FGH
```

and run it by typing

455

```
./FGH
```

Then, cut the script attached below, save it with the name scr FGH in the same directory where you run your code, and visualize the 4 eigenstates by typing

```
gnuplot<scr_FGH
```

where the file named scr FGH has the following lines:
  Download from (http://ursula.chem.yale.edu/~batista/classes/v572/scr FGH)

```
set yrange[0:7]
set xrange[-6:6]
set dat sty l
plot "wave.0001" u 1:2 w l lw 3
pause 1.
replot "wave.0001" u 1:3 w l  lw 3
pause 1.
replot "wave.0001" u 1:4 w l  lw 3
pause 1.
replot "wave.0002" u 1:3 w l lw 3
pause 1.
replot "wave.0002" u 1:4 w l lw 3
pause 1.
replot "wave.0003" u 1:3 w l lw 3
pause 1.
replot "wave.0003" u 1:4 w l lw 3
pause 1.
replot "wave.0004" u 1:3 w l lw 3
pause 1.
replot "wave.0004" u 1:4 w l lw 3
pause 5.0
```

61.22 Problem SBH

Solution to Problem SBH:

1. Write a program to solve the time independent Schrödinger equation by using the Single Boson Hamiltonian (SBH) method and apply it to find the first 4 eigenvalues and eigenfunctions of the DVR Hamiltonian, introduced by Eq. (992), with $H_{11} = 1$, $H_{22} = -1$ and $H_{12} = 0.1$.

Download the source code from (http://ursula.chem.yale.edu/~batista/classes/v572/SBH.f)

```fortran
      PROGRAM Program_SBH
c
c     This code computes the eigenvalues and eigenvectors of the 2x2
c     Single Boson Hamiltonian (SBH) described according
c     to Eq. (871) of the lecture notes.
c
      CALL READPARAM()
      CALL SBH()
      CALL EIGV()
      CALL DUMP()
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE READPARAM()
      implicit real *8(a-h,o-z)
      PARAMETER(npt=128,NC=1)
      COMMON /paque/ rmin(NC), rmax(NC), rmass(NC), dx(NC)
      rmax(1)=6.
      rmin(1)=-rmax(1)
      rmass(1)=1.
      DO I=1,NC
         dx(I)=(rmax(I)-rmin(I)) / npt
      END DO
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE SBH()
      implicit real *8(a-h,o-z)
      PARAMETER(npt=128,NC=1)
      COMMON /paque/ rmin(NC), rmax(NC), rmass(NC), dx(NC)
      COMMON/HAM/ VHAM(npt,npt)
      pi=acos(-1.)
      dp=2.0*pi/(rmax(1)-rmin(1))
      H11=1.0
      H22=-1.0
      H12=0.1
      DO i=1,npt
         r=(i-npt/2)*dx(1)
         DO ip=1, npt
            rp=(ip-npt/2)*dx(1)
            VHAM(i,ip) = 0.0
            IF(ip.EQ.i) VHAM(i,ip)=VHAM(i,ip)+Vd(r)
            DO k=1,npt/2
```

```fortran
                pk=dp*(k-npt/2)
                VHAM(i,ip)=VHAM(i,ip)+2.0*cos((rp-r)*pk)/npt*
     1               (Vp(r,rp)*pk**2-H22/4.0*pk**4)
            END DO
          END DO
       END DO
       RETURN
       END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       FUNCTION Vd(r)
       implicit real *8(a-h,o-z)
       x0=0.0
       H11=1.0
       H22=-1.0
       H12=0.1
       Vd=0.5*H11*(3.0-r**2)+H12/sqrt(2.0)*(4.0*r-r**3)+0.5*H22*(3.0*r**2
     1      -5.0*0.5-0.5*r**4)
       RETURN
       END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       FUNCTION Vp(r,rp)
       implicit real *8(a-h,o-z)
       x0=0.0
       H11=1.0
       H22=-1.0
       H12=0.1
       Vp=1.5*H22-0.5*H11-H12*(r+rp)/sqrt(8.0)-H22*(r**2+rp**2)/4.0
       RETURN
       END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       SUBROUTINE EIGV()
C
c     Diagonalization
c
c     VHAM: HERMITIAN MATRIX (INPUT)
c     EVALUES: EIGENVALUES (OUTPUT)
c     EVECT: EIGENVECTORS (OUTPUT)
c
       implicit real *8(a-h,o-z)
       REAL CRV, EVALUES,E,EVECT
       PARAMETER(npt=128,NC=1,npt2=npt**NC)
       COMMON/ HAM/ VHAM(npt2,npt2)
       COMMON /paque/ rmin(NC), rmax(NC), rmass(NC), dx(NC)
       PARAMETER(N=npt2,NP=N)
       COMMON/EIG/ EVALUES(npt2),EVECT(npt2,npt2)
       DIMENSION E(NP)
C
       DO I=1,N
          EVALUES(I)=0.0
          E(I)=0.0
          DO J=1,N
```

```fortran
         EVECT(J,I)=VHAM(J,I)
      END DO
      END DO
      CALL TRED22(EVECT,N,NP,EVALUES,E)
      CALL TQLI(EVALUES,E,N,NP,EVECT)
      CALL EIGSRT(EVALUES,EVECT,N,NP)
c
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE DUMP()
c
      implicit real *8(a-h,o-z)
      REAL CRV, EVALUES,EVECT,E
      character*9 B
      PARAMETER(npt=128,NC=1,npt2=npt**NC)
      PARAMETER(N=npt2,NP=N)
      COMMON/EIG/ EVALUES(npt2),EVECT(npt2,npt2)
      COMMON /paque/ rmin(NC), rmax(NC), rmass(NC), dx(NC)
      DIMENSION r(NC),j(NC)
c
      DO k=1, 10
         IND=npt2-(k-1)
         PRINT *, "E(",k,")=",EVALUES(IND)
      END DO
c
      DO l=1,4
      IND=npt2-(l-1)
      write(B, '(A,i4.4)') 'wave.', l
      OPEN(10,FILE=B)
      rsum=0.0
      DO i=1,npt2
         r(1)=(i-npt2/2)*dx(1)
         WRITE(10,22) r(1),Vd(r),EVALUES(IND)
    1        ,EVALUES(IND)+EVECT(i,IND)
         rsum=rsum + EVECT(i,IND)**2
      END DO
      PRINT *, "norm(",l,")=",rsum
      END DO
      CLOSE(10)
 22   FORMAT(6(e13.6,2x))
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutines to compute eigenvalues and eigenvectors from NR
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE TRED22(A,N,NP,D,E)
      IMPLICIT NONE
      INTEGER I,J,K,L,N,NP
      REAL A,D,E,H,SCALE,F,G,HH
      DIMENSION A(NP,NP),D(NP),E(NP)
```

459

```
      IF(N.GT.1)THEN
         DO 18 I=N,2,-1
            L=I-1
            H=0.
            SCALE=0.
            IF(L.GT.1)THEN
               DO 11 K=1,L
                  SCALE=SCALE+ABS(A(I,K))
11             CONTINUE
               IF(SCALE.EQ.0.)THEN
                  E(I)=A(I,L)
               ELSE
                  DO 12 K=1,L
                     A(I,K)=A(I,K)/SCALE
                     H=H+A(I,K)**2
12                CONTINUE
                  F=A(I,L)
                  G=-SIGN(SQRT(H),F)
                  E(I)=SCALE*G
                  H=H-F*G
                  A(I,L)=F-G
                  F=0.
                  DO 15 J=1,L
                     A(J,I)=A(I,J)/H
                     G=0.
                     DO 13 K=1,J
                        G=G+A(J,K)*A(I,K)
13                   CONTINUE
                     IF(L.GT.J)THEN
                        DO 14 K=J+1,L
                           G=G+A(K,J)*A(I,K)
14                      CONTINUE
                     ENDIF
                     E(J)=G/H
                     F=F+E(J)*A(I,J)
15                CONTINUE
                  HH=F/(H+H)
                  DO 17 J=1,L
                     F=A(I,J)
                     G=E(J)-HH*F
                     E(J)=G
                     DO 16 K=1,J
                        A(J,K)=A(J,K)-F*E(K)-G*A(I,K)
16                   CONTINUE
17                CONTINUE
               ENDIF
            ELSE
               E(I)=A(I,L)
            ENDIF
            D(I)=H
18       CONTINUE
```

460

```
      ENDIF
      D(1)=0.
      E(1)=0.
      DO 23 I=1,N
          L=I-1
          IF(D(I).NE.0.)THEN
              DO 21 J=1,L
                  G=0.
                  DO 19 K=1,L
                      G=G+A(I,K)*A(K,J)
 19               CONTINUE
                  DO 20 K=1,L
                      A(K,J)=A(K,J)-G*A(K,I)
 20               CONTINUE
 21           CONTINUE
          ENDIF
          D(I)=A(I,I)
          A(I,I)=1.
          IF(L.GE.1)THEN
              DO 22 J=1,L
                  A(I,J)=0.
                  A(J,I)=0.
 22           CONTINUE
          ENDIF
 23   CONTINUE
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE TQLI(D,E,N,NP,Z)
      IMPLICIT NONE
      INTEGER N,NP,I,K,L,M,ITER
      REAL D,E,Z,DD,G,R,S,C,P,F,B
      DIMENSION D(NP),E(NP),Z(NP,NP)
      IF (N.GT.1) THEN
          DO 11 I=2,N
              E(I-1)=E(I)
 11       CONTINUE
          E(N)=0.
          DO 15 L=1,N
              ITER=0
 1            DO 12 M=L,N-1
                  DD=ABS(D(M))+ABS(D(M+1))
                  IF (ABS(E(M))+DD.EQ.DD) GO TO 2
 12           CONTINUE
              M=N
 2            IF(M.NE.L)THEN
                  IF(ITER.EQ.30) THEN
                      PRINT *, 'too many iterations!'
                      STOP
                  END IF
                  ITER=ITER+1
```

461

```
                  G=(D(L+1)-D(L))/(2.*E(L))
                  R=SQRT(G**2+1.)
                  G=D(M)-D(L)+E(L)/(G+SIGN(R,G))
                  S=1.
                  C=1.
                  P=0.
                  DO 14 I=M-1,L,-1
                     F=S*E(I)
                     B=C*E(I)
                     IF(ABS(F).GE.ABS(G))THEN
                        C=G/F
                        R=SQRT(C**2+1.)
                        E(I+1)=F*R
                        S=1./R
                        C=C*S
                     ELSE
                        S=F/G
                        R=SQRT(S**2+1.)
                        E(I+1)=G*R
                        C=1./R
                        S=S*C
                     ENDIF
                     G=D(I+1)-P
                     R=(D(I)-G)*S+2.*C*B
                     P=S*R
                     D(I+1)=G+P
                     G=C*R-B
                     DO 13 K=1,N
                        F=Z(K,I+1)
                        Z(K,I+1)=S*Z(K,I)+C*F
                        Z(K,I)=C*Z(K,I)-S*F
 13                  CONTINUE
 14               CONTINUE
                  D(L)=D(L)-P
                  E(L)=G
                  E(M)=0.
                  GO TO 1
              ENDIF
 15        CONTINUE
       ENDIF
       RETURN
       END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       SUBROUTINE EIGSRT(D,V,N,NP)
       IMPLICIT NONE
       INTEGER N,NP,I,J,K
       REAL D,V,P
       DIMENSION D(NP),V(NP,NP)
       DO 13 I=1,N-1
         K=I
         P=D(I)
```

462

```fortran
        DO 11 J=I+1,N
          IF(D(J).GE.P)THEN
            K=J
            P=D(J)
          ENDIF
11      CONTINUE
        IF(K.NE.I)THEN
          D(K)=D(I)
          D(I)=P
          DO 12 J=1,N
            P=V(J,I)
            V(J,I)=V(J,K)
            V(J,K)=P
12        CONTINUE
        ENDIF
13    CONTINUE
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PIKSRT(N,ARR)
      IMPLICIT NONE
      INTEGER I,J,N
      REAL ARR,A
      DIMENSION ARR(N)
      DO 12 J=2,N
        A=ARR(J)
        DO 11 I=J-1,1,-1
          IF(ARR(I).LE.A)GO TO 10
          ARR(I+1)=ARR(I)
11      CONTINUE
        I=0
10      ARR(I+1)=A
12    CONTINUE
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccc
```

To visualize the output of the program listed above, save it in a file named SBH.f, compile it by typing

```
 gfortran SBH.f -o SBH
```

and run it by typing

```
./SBH
```

Then, cut the script attached below, save it with the name scr_SBH in the same directory where you run your code, and visualize the 4 eigenstates by typing

```
 gnuplot<scr_SBH
```

where the file named scr_SBH has the following lines:

Download from (http://ursula.chem.yale.edu/∼batista/classes/v572/scr_SBH)

```
set yrange[-2:4]
set xrange[-4:4]
plot "wave.0001" u 1:2 w l lw 3
pause .2
replot "wave.0001" u 1:3 w l  lw 3
pause .2
replot "wave.0001" u 1:4 w l  lw 3
pause .2
replot "wave.0002" u 1:3 w l lw 3
pause .2
replot "wave.0002" u 1:4 w l lw 3
pause .2
replot "wave.0003" u 1:3 w l lw 3
pause .2
replot "wave.0003" u 1:4 w l lw 3
pause .2
replot "wave.0004" u 1:3 w l lw 3
pause .2
replot "wave.0004" u 1:4 w l lw 3
pause 5.0
```

2. Propagate a state initialized as $\psi(0) = \langle x|0\rangle$, according to Eq. (978), by using the DVR Hamiltonian introduced by Eq. (992) and the Lanczos propagation scheme in the Krylov basis as implemented in *Expokit*. Compute the time-dependent probability of finding the system in the initial state $P_0(t) = \langle\psi(t)|0\rangle\langle0|\psi(t)\rangle$ and compare your results to benchmark calculations based on the standard SOFT approach described in Sec. 7.

Download the source code from here

Untar the tarball files by typing

tar -xvf SBH_expokit.tar

Compile by typing

make SBH

and run the program by typing

./SBH

Visualize the numerical time-dependent probability generated by the SBH method and compare it to benchmark calculations based on the non-adaiabatic SOFT propagation described in Sec. 7 by typing

gnuplot < scrp

Visualize the time-dependent wavepacket (real part and absolute value) generated by the SBH method by typing

gnuplot

61.23 Problem: WGB

Write a program to propagate a wavepacket, initialized according to Eq. (251) with coefficients defined according to (250), with $\alpha = 1$, $p_0 = 0$, $x_0 = -2\eta$, with $\eta = 1.3544$, and evolving in a quartic double-well potential $V(x) = x^4/(16\eta) - x^2/2$. Implement the WGB propagation method to evolve the expansion coefficients, according to Eqs. (255) and (261). Compare your results to the corresponding proagation based on the SOFT method described in Sec. 6.

Download the source code from here

Untar the tarball files by typing

tar -xvf WGB.tar

change directory by typing

cd WGB

Compile by typing

gfortran wgb.f -o wgb

and run the program by typing

./wgb

Visualize the evolution of the wavepacket and the comparison to benchmark calculations based on the SOFT propagation method described in Sec. 6 by running the mathematica notebook $wgb\_vis.nb$

61.24   Problem 17

17.1. Compute the photoabsorption spectrum of I_2. Assume that the transition dipole moments are independent of nuclear coordinates, and that the only allowed electronic transition induced by photoabsorption of I_2 is the $B \leftarrow X$ excitation. Assume the ground (g) and excited (e) states of I_2 can be described by the Morse Potential $V(R) = D_e \left(1 - e^{-\beta(R-R_{eq})}\right)^2 + V_0$, where R is the bond-length of I_2 and $V_0(g) = 0.00\ eV$; $V_0(e) = 0.94\ eV$; $D_e(g) = 18941\ cm^{-1}$; $D_e(e) = 4911\ cm^{-1}$; $\beta(g) = 1.517\ Å^{-1}$; $\beta(e) = 1.535\ Å^{-1}$; $R_{eq}(g) = 2.66\ Å$ and $R_{eq}(e) = 3.105\ Å$.

17.2. Compute the direct photoelectron detachment spectrum of I_2^- assuming that the electronic transitions induced by photoelectron detachment of $I_2^-(X)$ generate I_2 in the electronic states X and B.

Assume that the potential energy surfaces of the states $I_2^-(X)$, $I_2(X)$ and $I_2(B)$ can be described by simple Morse potentials, as reported by Batista and Coker [*J. Chem. Phys.* (1997) **106**:7102-7116].

Download the alpine source code from here
Untar the tarball files by typing
tar -xvf I2mPEDSpec.tar
change directory by typing
cd I2mPEDSpec
Compile by typing
./compile.sh
and run the program by typing
./alpine
./trans
Visualize the evolution of the wavepacket in the intermediate A' state by typing
gnuplot $anim.gpt$
Visualize the PED spectrum at 320 fs after photoelectrondetachment into the B I2 neutral state by typing
gnuplot plot "pw008.out" w l

61.25   Computational Problem: WTP

1. Write a program to write the DVR Liouvillian, introduced by Eq. (992), for the Morse potential.
2. Propagate the Wigner transform of a state initialized as $|0\rangle$, according to Eq. (978), using the DVR Liouvillian of item 1 and the Lanczos propagation scheme in the Krylov basis as implemented in *Expokit*. Compare the time-evolved Wigner transform to the corresponding propagation based on the classical Liouvillian $\mathbf{L}_c$.

Download the source code from here
Untar the tarball files by typing
tar -xvf FGL2_expokit.tar
change directory by typing
cd FGL2_expokit

Compile by typing

make FGL2

and run the program by typing

./FGL2

move the output files to the QM subdirectory by typing mv *wavp* QM/.

then modify the function D3V to make it zero, recompile and run

move the output files to the CM subdirectory by typing mv *wavp* CM/.

Visualize the evolution of the QM time-dependent Wigner transform as compared to the classically propagated Wigner transform by typing

gnuplot

61.26 Computational Problem 2-level WT

Write a program to propagate the Wigner transform of a 2-level system, described to the MM Hamiltonian introduced by Eq. (314), with $H_{11} = -H_{22} = H_{12} = H_{21} = 1.0$, initialized in one of the 2 states according to Eq. (324) and evolving by Velocity Verlet according to Eq. (320). Compare the Rabi oscillations of the time-dependent survival probability to the corresponding results obtained by SOFT quantum propagation.

Download the source code from here

Untar the tarball files by typing

```
tar -xvf VV_2level.tar
```

change directory by typing

```
cd VV_2level
```

Compile by typing

```
make VV_2level
```

Run the program by typing

```
./VV_2level
```

Visualize the comparison of Rabi oscillations by typing

```
gnuplot<scrp
```

Visualize the time-dependent Wigner transform by typing

```
gnuplot<scr
```

61.27 Computational Problem ICN

Write a computer program to implement the SOFT approach, as applied to the simulation of the ICN photodissociation as reported in JCP 112, 5566 (2000).

Download the source code from here

Untar the tarball files by typing

```
tar -xvf ICN.tar
```

change directory by typing

```
cd ICN
```

Visualize results by typing

```
 chmod +x scr
```

```
 ./scr
```

The visualization script generates the following files:

```
autocorr.pdf
norms.pdf
spectrum.pdf
wavefnbent.pdf
wavefnlinear.pdf
CNrot266.eps
CNrot248.eps
CNrot-lin1.eps
CNrot-lin2.eps
```

Compile by typing

```
 gfortran 3de.f -o 3de
```

and run the program by typing

```
 ./3de<pp_ICN
```

Input file:

```
39602.52 56014.306 0.0 0.0 0.0022735972 0.0010456488 5.1873 0.0
689  1 1  0.50
4.00 14.0 -6.28318530718  6.28318530718
0 6

#       read(5,*) amassx,amassy,xk,yk,sigx,sigy,xoff,yoff
#       read(5,*) nstep,ireport,ndump,dt
#       read(5,*) xmin,xmax,ymin,ymax
#       read(5,*) ntype,nlit
```

Source code:

```
      PROGRAM main
c
c     This code performs FFT time propagation of ICN photodissociation
c
      IMPLICIT NONE
      INTEGER NN,igammay,igammax,nptx,npty,ndump,JE,je2,i
      INTEGER istep,nstep,ireport,ndum,ntype,nstart,nlit,LL
      REAL pi,dx,dy,x,y,dt
      COMPLEX vprop,tprop1,tprop2,chi,rct,chi0
      PARAMETER(igammay=9,npty=2**igammay,NN=2)
      PARAMETER(igammax=9,nptx=2**igammax)
      DIMENSION vprop(nptx,npty,NN,NN),tprop1(nptx,npty)
      DIMENSION chi(nptx,npty,NN),tprop2(nptx,npty),rct(3000)
      common/ochi/chi0(nptx,nptx,NN)
c
c      CALL READTABLES()
      OPEN(1,FILE="rct")
      CALL ReadParam(nstep,ireport,ndump,dt,ntype,nlit)
      OPEN(unit=13,file='restartf',access='sequential'
     1   ,form='unformatted')
      REWIND 13
      IF(ntype.eq.1) THEN
 read(13) nstart,chi,rct,chi0
         rewind 13
      ELSE
         CALL INITWF(chi)
         nstart=1
      END IF

      CALL SetKinProp1(dt,tprop1)
      CALL SetKinProp2(dt,tprop2)
      CALL SetPotProp(dx,dy,dt,vprop)
c

c A file is opened to print norm information.
OPEN(18,FILE="norms.dat")

DO istep=nstart,nstep
        IF(mod(istep,ireport).eq.0)
     1     write(6,*) 'istep= ',istep,'  from=',nstep
        if(mod(istep-1,ndump).eq.0) then
```

471

```fortran
            CALL WriteRct(rct,chi,istep,dx,dy)
            CALL WriteProb(chi,istep,dx,dy,dt)
          END IF
          JE=istep
cc I changed the mod 10 to ireport in the next two lines
cc to print every instance.
 IF(mod(JE-1,ireport).eq.0) then
           je2=JE/ireport
           IF(mod(je2,1).EQ.0) THEN
        CALL DUMPWF(chi,je2)
           END IF
          END IF
c         IF(istep.GT.nstart) THEN
           DO LL=1,nlit
        CALL PROPAGATE(chi,vprop,tprop1,tprop2)
           END DO
       END DO
       DO i=1,nstep
          WRITE(1,22) dt*(i-1), real(rct(i)), aimag(rct(i))
       END DO
       nstep=nstep+1
       write(13) nstep,chi,rct,chi0
 22    FORMAT(6(e13.6,2x))
       END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       SUBROUTINE INITWF(chi)
c
       IMPLICIT NONE
       INTEGER NN,je2,nptx,npty,igammax,igammay,ii,jj
       COMPLEX chi,chi0,EYE,psi,CRV
       REAL c1,xmin,ymin,xmax,ymax,dx,dy,xoff,yoff
       REAL pi,cnormx,cnormy,EVALUES
       REAL amassx,amassy,xk,yk,sigx,sigy
       REAL*8 x,y
       PARAMETER(igammay=9,npty=2**igammay,NN=2)
       PARAMETER(igammax=9,nptx=2**igammax)
       DIMENSION chi(nptx,npty,NN),EVALUES(NN),psi(NN,NN),CRV(NN,NN)
       common/ochi/chi0(nptx,nptx,NN)
       common /xy/  xmin,xmax,ymin,ymax,xoff,yoff
       common /packet/  amassx,amassy,xk,yk,sigx,sigy
c
       EYE=(0.0,1.0)
```

```fortran
      pi=3.1415927
      dx=(xmax-xmin)/real(nptx)
      dy=(ymax-ymin)/real(npty)
c
      cnormx=1./sqrt(sqrt(0.5*pi)*sigx)
      cnormy=1./sqrt(sqrt(0.5*pi)*sigy)
      do ii=1,nptx
         x=xmin+ii*dx
         do jj=1,npty
            y=ymin+jj*dy
            chi(ii,jj,1)=cnormx*cnormy*exp(eye*xk*(x-xoff))
     1         *exp(-((x-xoff)/sigx)**2)
     1         *exp(eye*yk*(y-yoff))*exp(-((y-yoff)/sigy)**2)
            chi(ii,jj,2)=0.0
c
            chi0(ii,jj,1)=chi(ii,jj,1)
            chi0(ii,jj,2)=chi(ii,jj,2)
         end do
      end do
      CRV(1,1)=0.0
      CRV(1,2)=0.0
      CRV(2,1)=0.0
      CRV(2,2)=0.0
      y=0.0
      do ii=1,nptx
         x=(2.0+ii*6.0/nptx)/0.529177
         CALL HAMIL(CRV,x,y)
         WRITE(10,22) x,real(CRV(1,1))*27.2
     1       ,real(CRV(2,2))*27.2
     1       ,real(CRV(1,2))*27.2
         CALL SCHROC1(CRV,psi,EVALUES)
      END DO
 22   FORMAT(6(e13.6,2x))
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE DUMPWF(chi,je2)
c
      IMPLICIT NONE
      INTEGER je2,nptx,npty,igammax,igammay,ii,jj,NN
      COMPLEX chi,CRV
      REAL c1,c2,xmin,ymin,xmax,ymax,dx,dy,xoff,yoff
```

473

```
        REAL Potential,rat,hig
        REAL*8 x,y
        character*10 B1
        PARAMETER(igammay=9,npty=2**igammay,NN=2)
        PARAMETER(igammax=9,nptx=2**igammax)
        DIMENSION chi(nptx,npty,NN),CRV(NN,NN)
        common /xy/  xmin,xmax,ymin,ymax,xoff,yoff
c
        dx=(xmax-xmin)/real(nptx)
        dy=(ymax-ymin)/real(npty)
        rat=30.
        hig=-.0
c
        write(B1, '(A,i5.5)') 'wave.', 700+je2
        OPEN(51,FILE=B1)

        do ii=1,nptx
           x=xmin+ii*dx
           do jj=1,npty
              y=ymin+jj*dy
              c1=chi(ii,jj,1)*conjg(chi(ii,jj,1))
              c2=chi(ii,jj,2)*conjg(chi(ii,jj,2))
c              write(200+je2,44) x,y
c     1            ,real(chi(ii,jj,1)),aimag(chi(ii,jj,1))
c     1            ,real(chi(ii,jj,2)),aimag(chi(ii,jj,2))
              write(51,44) x,y
     1            ,real(chi(ii,jj,1)),aimag(chi(ii,jj,1))
     1            ,real(chi(ii,jj,2)),aimag(chi(ii,jj,2))
              IF(je2.EQ.0) THEN
                 CALL HAMIL(CRV,x,y)
                 write(199,44) x,y,real(CRV(1,1)),real(CRV(2,2))
              END IF
           end do
c           write(200+je2,44)
           write(51,44)
           IF(je2.EQ.0) write(199,44)
        end do
        CLOSE(51)
 44     format(10(e13.6,2x))
        RETURN
        END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

474

```fortran
      subroutine ReadParam(nstep,ireport,ndump,dt,ntype,nlit)
C
      IMPLICIT NONE
      INTEGER ntype,nstep,ireport,ndump,nlit
      REAL amassx,amassy,xk,yk,sigx,sigy,dt
      common /packet/  amassx,amassy,xk,yk,sigx,sigy
      common /xy/  xmin,xmax,ymin,ymax,xoff,yoff
C
      read(5,*) amassx,amassy,xk,yk,sigx,sigy,xoff,yoff
      read(5,*) nstep,ireport,ndump,dt
      read(5,*) xmin,xmax,ymin,ymax
      read(5,*) ntype,nlit

      sigx=sqrt(2./(amassx*sigx))
      sigy=sqrt(2./(amassy*sigy))
      write (6,*) 'mass:',amassx,amassy
      write (6,*) 'xy:', xmin,xmax,ymin,ymax
      write (6,*) 'x/y off:' ,xoff,yoff
      write (6,*) 'x/y init momentum:' ,xk,yk
      write (6,*) 'sigx/y :', sigx,sigy
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetKinProp(dt,tprop)

      IMPLICIT NONE
      INTEGER nptx,npty,kx,ky,nx,ny,igammay,igammax,NN
      REAL ysc,xsc,xmin,xmax,ymin,ymax,xoff,yoff,propfacx,propfacy
      REAL amassx,amassy,xk,yk,sigx,sigy,pi,alenx,aleny,dt
      COMPLEX tprop,eye
      parameter(igammay=9,npty=2**igammay,NN=2)
      parameter(igammax=9,nptx=2**igammax)
      DIMENSION tprop(nptx,npty)
      common /xy/  xmin,xmax,ymin,ymax,xoff,yoff
      common /packet/  amassx,amassy,xk,yk,sigx,sigy

      eye=(0.,1.)
      pi = 3.1415927

      alenx=xmax-xmin
      aleny=ymax-ymin
      propfacx=-dt/2./amassx*(2.*pi)**2
```

```fortran
      propfacy=-dt/2./amassy*(2.*pi)**2

      do kx=1,nptx
         if(kx.le.(nptx/2+1)) then
            nx=kx-1
         else
            nx=kx-1-nptx
         end if
         xsc=0.
         if(nx.ne.0) xsc=real(nx)/alenx
         do ky=1,npty
            if(ky.le.(npty/2+1)) then
               ny=ky-1
            else
               ny=ky-1-npty
            end if
            ysc=0.
            if(ny.ne.0) ysc=real(ny)/aleny
            tprop(kx,ky)=exp(eye*(propfacx*xsc**2+propfacy*ysc**2))
      end do
c
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetKinProp1(dt,tprop1)

      IMPLICIT NONE
      INTEGER nptx,npty,kx,ky,nx,ny,igammay,igammax,NN
      REAL ysc,xsc,xmin,xmax,ymin,ymax,xoff,yoff,propfacx,propfacy
      REAL amassx,amassy,xk,yk,sigx,sigy,pi,alenx,aleny,dt
      COMPLEX tprop1,eye
      parameter(igammax=9,nptx=2**igammax,NN=2)
      parameter(igammay=9,npty=2**igammay)
      DIMENSION tprop1(nptx,npty)
      common /xy/   xmin,xmax,ymin,ymax,xoff,yoff
      common /packet/  amassx,amassy,xk,yk,sigx,sigy

      eye=(0.,1.)
      pi = 3.1415927

      alenx=xmax-xmin
      propfacx=-dt/2./amassx*(2.*pi)**2
```

476

```fortran
      do kx=1,nptx
         if(kx.le.(nptx/2+1)) then
            nx=kx-1
         else
            nx=kx-1-nptx
         end if
         xsc=0.
         if(nx.ne.0) xsc=real(nx)/alenx
         do ky=1,npty
            tprop1(kx,ky)=exp(eye*(propfacx*xsc**2))
         end do
      end do
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetKinProp2(dt,tprop2)

      IMPLICIT NONE
      INTEGER nptx,npty,kx,ky,nx,ny,igammay,igammax,NN
      REAL ysc,xsc,xmin,xmax,ymin,ymax,xoff,yoff,propfacx,propfacy
      REAL x,dx,amassx,amassy,xk,yk,sigx,sigy,pi,alenx,aleny,dt
      COMPLEX tprop2,eye
      parameter(igammay=9,npty=2**igammay,NN=2)
      parameter(igammax=9,nptx=2**igammax)
      DIMENSION tprop2(nptx,npty)
      common /xy/   xmin,xmax,ymin,ymax,xoff,yoff
      common /packet/  amassx,amassy,xk,yk,sigx,sigy
c
      eye=(0.,1.)
      pi = 3.1415927
c
      dx=(xmax-xmin)/real(nptx)
c
      do kx=1,nptx
         x=xmin+kx*dx
         do ky=1,npty
            if(ky.le.(npty/2+1)) then
               ny=ky-1
            else
               ny=ky-1-npty
```

```fortran
            end if
            ysc=0.
            if(ny.ne.0) ysc=real(ny)*2.0*pi/aleny
            tprop2(kx,ky)=exp(-eye*dt*ysc**2*0.5*
     1            (1.0/(amassx*x*x)+1.0/(amassy)))
          end do
        end do
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine SetPotProp(dx,dy,dt,vprop)
      IMPLICIT NONE
      INTEGER NN,ii,jj,nptx,npty,i,j,k,igammax,igammay
      REAL xmin,xmax,ymin,ymax,xoff,yoff,dx,dy,dt
      REAL Potential,rsqnx,rsqny,EVALUES
      REAL*8 x,y
      COMPLEX vp,vprop,eye,dummy,psi,CRV
      parameter(igammay=9,npty=2**igammay,NN=2)
      parameter(igammax=9,nptx=2**igammax)
      DIMENSION vprop(nptx,npty,NN,NN),psi(NN,NN),CRV(NN,NN)
      DIMENSION vp(NN,NN),dummy(NN,NN),EVALUES(NN)
      common /xy/  xmin,xmax,ymin,ymax,xoff,yoff
      eye=(0.,1.)
      dx=(xmax-xmin)/real(nptx)
      dy=(ymax-ymin)/real(npty)
      do ii=1,nptx
         x=xmin+ii*dx
         do jj=1,npty
            y=ymin+jj*dy
            CALL HAMIL(CRV,x,y)
            CALL SCHROC1(CRV,psi,EVALUES)
            vp(1,1)=exp(-eye*0.5*dt*EVALUES(1))
            vp(1,2)=0.0
            vp(2,1)=0.0
            vp(2,2)=exp(-eye*0.5*dt*EVALUES(2))
            do i=1,2
               do j=1,2
                  dummy(i,j)=0.
                  do k=1,2
                     dummy(i,j)=dummy(i,j)+vp(i,k)*psi(j,k)
                  end do
```

```
               end do
            end do
            do i=1,2
               do j=1,2
                  vp(i,j)=0.
                  do k=1,2
                     vp(i,j)=vp(i,j)+psi(i,k)*dummy(k,j)
                  end do
               end do
               rsqnx=1.0/sqrt(1.0*nptx)
               rsqny=1.0/sqrt(1.0*npty)
               do i=1,2
                  do j=1,2
                     vprop(ii,jj,i,j)=vp(i,j)*rsqny*rsqnx
                  end do
               end do
            end do
         end do
      end do
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine WriteRct(rct,chi,istep,dx,dy)
c
      IMPLICIT NONE
      INTEGER NN,igammay,npty,igammax,nptx,ii,jj,istep
      REAL x,y,amassx,amassy,xk,yk,sigx,sigy,anorm,ref,tra
      REAL xmin,xmax,ymin,ymax,xoff,yoff,a1,anorm1,dx,dy,dt
      COMPLEX chi,chi0,rct
      parameter(igammay=9,npty=2**igammay,NN=2)
      parameter(igammax=9,nptx=2**igammax)
      DIMENSION chi(nptx,npty,NN),rct(3000)
      common /packet/  amassx,amassy,xk,yk,sigx,sigy
      common /xy/   xmin,xmax,ymin,ymax,xoff,yoff
      common/ochi/chi0(nptx,nptx,NN)
c
      rct(istep)=0.0
      do ii=1,nptx
         x=xmin+ii*dx
         do jj=1,nptx
            y=ymin+jj*dy
            rct(istep)=rct(istep)+conjg(chi0(ii,jj,1))*chi(ii,jj,1)
     1              +conjg(chi0(ii,jj,2))*chi(ii,jj,2)
```

```fortran
         end do
      end do
      rct(istep)=rct(istep)*dx*dy
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      FUNCTION Potential(x,y)
      IMPLICIT NONE
      REAL x,y,Potential,a,b,c,w
      a=0.5
      b=0.046146
      c=0.12
      w=1.0
      Potential=-a*x*x+x*x*x*x/(16.0*1.3544)+w**2/2.0
   1    *(-y-c*x**2/w**2)**2
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PROPAGATE(chi,vprop,tprop1,tprop2)
      IMPLICIT NONE
      INTEGER i,j,NN,in,ii,jj,nptx,npty,vec,igammay,igammax,vecx,vecy
      COMPLEX chi,vprop,chin1,chin2,chix1,chix2,chiy1,chiy2
      PARAMETER(igammay=9,npty=2**igammay,NN=2)
      PARAMETER(igammax=9,nptx=2**igammax)
      DIMENSION vec(2),chi(nptx,npty,NN)
      DIMENSION chin1(npty*nptx),chin2(npty*nptx)
      DIMENSION chix1(nptx),chix2(nptx)
      DIMENSION chiy1(npty),chiy2(npty),tprop1(nptx,npty)
      DIMENSION vprop(nptx,npty,NN,NN),tprop2(nptx,npty)
      vec(1)=nptx
      vec(2)=npty
      vecx=nptx
      vecy=npty
      DO ii=1,nptx
         DO jj=1,npty
            in=(ii-1)*npty+jj
            chin1(in)=0.0
            chin2(in)=0.0
            DO j=1,NN
               chin1(in)=chin1(in)+vprop(ii,jj,1,j)*chi(ii,jj,j)
               chin2(in)=chin2(in)+vprop(ii,jj,2,j)*chi(ii,jj,j)
            END DO
```

```
         END DO
      END DO
cccccccccccccccccccccccccccccccccccccccccccccccccccccc
      DO ii=1,nptx
         DO jj=1,npty
            in=(ii-1)*npty+jj
            chiy1(jj)=chin1(in)
            chiy2(jj)=chin2(in)
         END DO
         CALL fourn(chiy1,vecy,1,1)
         CALL fourn(chiy2,vecy,1,1)
         DO jj=1,npty
            in=(ii-1)*npty+jj
            chin1(in)=chiy1(jj)
            chin2(in)=chiy2(jj)
         END DO
      END DO
c
      DO ii=1,nptx
         DO jj=1,npty
            in=(ii-1)*npty+jj
            chin1(in)=tprop2(ii,jj)*chin1(in)
            chin2(in)=tprop2(ii,jj)*chin2(in)
         END DO
      END DO
c
      DO jj=1,npty
         DO ii=1,nptx
            in=(ii-1)*npty+jj
            chix1(ii)=chin1(in)
            chix2(ii)=chin2(in)
         END DO
         CALL fourn(chix1,vecx,1,1)
         CALL fourn(chix2,vecx,1,1)
         DO ii=1,nptx
            in=(ii-1)*npty+jj
            chin1(in)=chix1(ii)
         END DO
      END DO
c
c       CALL fourn(chin1,vec,2,1)
c       CALL fourn(chin2,vec,2,1)
```

481

```
ccccccccccccccccccccccccccccccccccccccccccccccccccc
        DO ii=1,nptx
           DO jj=1,npty
              in=(ii-1)*npty+jj
              chin1(in)=tprop1(ii,jj)*chin1(in)
              chin2(in)=tprop1(ii,jj)*chin2(in)
           END DO
        END DO
c
c        DO ii=1,nptx
c           DO jj=1,npty
c              in=(ii-1)*npty+jj
c              chin1(in)=tprop(ii,jj)*chin1(in)
c              chin2(in)=tprop(ii,jj)*chin2(in)
c           END DO
c        END DO
c
        CALL fourn(chin1,vec,2,-1)
        CALL fourn(chin2,vec,2,-1)
        DO ii=1,nptx
           DO jj=1,npty
              in=(ii-1)*npty+jj
              DO i=1,NN
                 chi(ii,jj,i)=vprop(ii,jj,i,1)*chin1(in)
     1                +vprop(ii,jj,i,2)*chin2(in)
              END DO
           END DO
        END DO
        RETURN
        END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c        SUBROUTINE HAMIL(CRV,x,y)
cc
c        IMPLICIT NONE
c        INTEGER NN,IDX,NNC,NPT
c        COMPLEX CRV
c        REAL*8 V1,V3,V13,xnc,pnc
c        REAL x,y,Potential,eV,ct
c        PARAMETER(NN=2,NNC=2,NPT=1)
c        DIMENSION CRV(NN,NN)
c        COMMON/NUCLEAR/xnc(NNC,NPT),pnc(NNC,NPT)
cc
```

```
c       IDX=1
c       xnc(1,1) = x
c       xnc(2,1) = y
c       CALL PES(V1,V3,V13,IDX)
c       CRV(1,1) = V3
c       CRV(1,2) = V13
c       CRV(2,1) = V13
c       CRV(2,2) = V1
cc
c       RETURN
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c       SUBROUTINE PES(V1,V3,V13,IDX)
cc
cc      Mokomura's PES JCP 100,7,4894-4909,(1994)
cc
c       IMPLICIT NONE
c       INTEGER NNC,IDX,NPT
c       REAL*8 V1,V3,V13,DEI,theta,r,RR
c       REAL*8 rinf,RRe,DR,x,y,z,a,b,c,kcn
c       REAL*8 AA,BB,CC,DD,xnc,pnc
cc
c       REAL*8 delrx,V1rev,V3rev,V4rev
c       REAL*8 a1,a2,a3,a4,a5,a6
c       REAL*8 b1,b2,b3,b4,b5,b6
c       REAL*8 c1,c2,c3,c4,c5,c6
c       REAL*8 CM,q1,q2,csen
cc
c       PARAMETER(NNC=2,NPT=1)
c       COMMON/TABLEIb/a(10),b(10),c(10),kcn
c       COMMON/const/AA(4),BB(3),CC(4),DD(3)
c       COMMON/NUCLEAR/xnc(NNC,NPT),pnc(NNC,NPT)
cc
c       rinf=2.2372
c       RRe=3.7909
cc       r=2.2141
c       r=rinf
c       CM=1.1922076
c       q1=xnc(1,IDX)**2.
c       q2=2.0*CM*xnc(1,IDX)*cos(xnc(2,IDX))
c       RR=(q1-q2+CM**2.)**0.5
c       theta= (xnc(1,IDX)*cos(xnc(2,IDX))-CM)/RR
c       DR=RR-RRe
```

```
cc
ccccc Comparison with Morokuma's ccccccccccc
cc      RR=xnc(1,IDX)
cc      DR=RR-RRe
cc      theta=xnc(2,IDX)
cc      theta=cos(theta)
ccccccccccccccccccccccccccccccccccccccccccc
c      CALL COEFF(DR,theta)
cc
c      x=r-(rinf+a(6)*EXP(-a(7)*DR))
c      y=r-(rinf+b(6)*EXP(-b(7)*DR))
c      z=r-(rinf+c(6)*EXP(-c(7)*DR))
cc
c      DEI=0.034646
cc
cc      V1=AA(1)*EXP(-a(1)*DR)+a(2)*EXP(-a(3)*DR)
cc     1 +0.5*kcn*(1.0+a(4)*EXP(-a(5)*DR))*x*x+AA(2)*EXP(-a(8)*DR)
cc     1      +AA(3)*EXP(-a(9)*DR)*COS(theta)-(AA(2)*EXP(-a(8)*DR)
cc     1      +AA(3)*EXP(-a(9)*DR))*COS(2.0*theta)
c      V1=AA(1)*EXP(-a(1)*DR)+a(2)*EXP(-a(3)*DR)
c     1 +0.5*kcn*(1.0+a(4)*EXP(-a(5)*DR))*x*x+AA(2)*EXP(-a(8)*DR)
c     1    +AA(3)*EXP(-a(9)*DR)*theta-(AA(2)*EXP(-a(8)*DR)
c     1     +AA(3)*EXP(-a(9)*DR))*(2.*theta**2.-1.)
cc
c
cc      1 +0.5*kcn*(1.0+b(4)*EXP(-b(5)*DR))*y*y+BB(2)*EXP(-b(8)*DR)
cc     1      +BB(3)*EXP(-b(9)*DR)*COS(theta)-(BB(2)*EXP(-b(8)*DR)
cc     1      +BB(3)*EXP(-b(9)*DR))*COS(2.0*theta)+DEI
c      V3=BB(1)*EXP(-b(1)*DR)+b(2)*EXP(-b(3)*DR)
c     1 +0.5*kcn*(1.0+b(4)*EXP(-b(5)*DR))*y*y+BB(2)*EXP(-b(8)*DR)
c     1      +BB(3)*EXP(-b(9)*DR)*theta-(BB(2)*EXP(-b(8)*DR)
c     1      +BB(3)*EXP(-b(9)*DR))*(2.*theta**2.-1.)+DEI
cc
cc      V13=DD(1)*(1.0-COS(2.0*theta))/((RR-DD(2))**2+DD(3)**2)
c      V13=DD(1)*(1.0-(2.*theta**2.-1))/((RR-DD(2))**2+DD(3)**2)
cc
cc#ifdef rev
c      a1=0.0257
c      a2=0.0
c      a3=0.885
c      a4=5.298
c      a5=-1.525
```

484

```
c       a6=-0.225
cc        delrx=RR-(a4+a5*COS(theta)+a6*COS(2.0*theta))
c         delrx=RR-(a4+a5*(theta)+a6*(2.0*theta**2.-1.0))
cc        V1rev=(1.0-COS(theta))*(a1+a2*delrx)*exp(-a3*delrx)
c        V1rev=(1.0-theta)*(a1+a2*delrx)*exp(-a3*delrx)
c        b1=0.0403
c        b2=-0.0258
c        b3=0.708
c        b4=4.758
c        b5=-1.181
c        b6=-0.194
cc        delrx=RR-(b4+b5*COS(theta)+b6*COS(2.0*theta))
c        delrx=RR-(b4+b5*(theta)+b6*(2.0*theta**2.-1.))
cc        V3rev=(1.0-COS(theta))*(b1+b2*delrx)*exp(-b3*delrx)
c        V3rev=(1.0-theta)*(b1+b2*delrx)*exp(-b3*delrx)
cc        c1=0.0245
cc        c2=-0.0054
cc        c3=0.906
cc        c4=4.932
cc        c5=-1.176
cc        c6=-0.250
cc        delrx=RR-(c4+c5*COS(theta)+c6*COS(2.0*theta))
cc        V4rev=(1.0-COS(theta))*(c1+c2*delrx)*exp(-c3*delrx)
cc
cc        V1=V1+V1rev
cc        V2=V2+V1rev
cc        V3=V3+V3rev
cc        V4=V4+V4rev
cc        V5=V5+V4rev
cc
cc#endif
cc
c        RETURN
c        END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c        SUBROUTINE COEFF(DR,theta)
c        IMPLICIT NONE
cc
c        INTEGER I
c        REAL*8 AA,BB,CC,DD,DR,theta,X0,X1,X2,Y0,Y1,Y2
c        COMMON/TABLEIa/X0(11),X1(11),X2(11),Y0(3),Y1(3),Y2(3)
c        COMMON/const/AA(4),BB(3),CC(4),DD(3)
```

```fortran
cc
c       DO I=1,4
c          AA(I)=X0(I)+X1(I)*DR+X2(I)*DR*DR
c          IF(I.LE.3) THEN
c             BB(I)=X0(I+4)+X1(I+4)*DR+X2(I+4)*DR*DR
c             DD(I)=Y0(I)+Y1(I)*(theta)+Y2(I)*((2.0*theta**2.-1.))
c          END IF
c          CC(I)=X0(I+7)+X1(I+7)*DR+X2(I+7)*DR*DR
c       END DO
cc
c       RETURN
c       END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c       SUBROUTINE HAMIL(CRV,R,theta)
cc
c       INTEGER NN,I,J
c       REAL*8 ct,GS,V1,V2,V12
c       REAL R,theta
c       COMPLEX CRV
c       PARAMETER(ct=4.5563353E-6,NN=2)
cc       PARAMETER(ct=1.0E-4,NN=2)
c       DIMENSION CRV(NN,NN)
cc
c       DO I=1,NN
c          DO J=1,NN
c             CRV(I,J)=0.0
c          END DO
c       END DO
cc
c       CALL PES(GS,V1,V2,V12,R,theta)
cc
c       CRV(1,1)=V1*ct
c       CRV(2,2)=V2*ct
c       CRV(1,2)=V12*ct
c       CRV(2,1)=V12*ct
cc
c       RETURN
c       END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c       SUBROUTINE SCHROC1(CRV,EVECT,EVALUES)
c       IMPLICIT NONE
c       CHARACTER*1 JOB
```

```
c        INTEGER NP,LDA,LWORK,IFAIL,NNC,I,J,IDX
c        REAL*8 DDV,EV,VI,WI,WORK,VR,HHPOT
c        REAL EVALUES
c        COMPLEX CRV,EVECT,EYE
c        PARAMETER(NP=2,LWORK=64*NP,NNC=NP)
c        DIMENSION EVALUES(NP),EVECT(NP,NP),DDV(NP,NP),VI(NP,NP),EV(NP)
c        DIMENSION WORK(LWORK),VR(NP,NP),CRV(NP,NP),WI(NP)
c        EXTERNAL f02ebf
c        EYE=(0.0,1.0)
c        JOB='V'
c        IFAIL = 0
c        DO I=1,NP
c           DO J=1,NP
c           END DO
c        END DO
c        CALL f02ebf(JOB,NP,DDV,NP,EV,
c       1     WI,VR,NP,VI,NP,WORK,LWORK,IFAIL)
c        DO I=1,NP
c           EVALUES(I)=EV(I)
c           DO J=1,NP
c              EVECT(I,J)=VR(I,J)+EYE*VI(I,J)
c           END DO
c        END DO
c        RETURN
c        END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       SUBROUTINE READTABLES()
c
       IMPLICIT NONE
       INTEGER I
       REAL*8 X0,X1,X2,Y0,Y1,Y2,a,b,c,kcn
       COMMON/TABLEIa/X0(11),X1(11),X2(11),Y0(3),Y1(3),Y2(3)
       COMMON/TABLEIb/a(10),b(10),c(10),kcn
c
       OPEN(1,FILE="table_I_II")
       DO I=1,11
          READ(1,*) X0(I),X1(I),X2(I)
       END DO
       DO I=1,10
          READ(1,*) a(I),b(I),c(I)
       END DO
       READ(1,*) kcn
```

```
      DO I=1,3
         READ(1,*) Y0(I),Y1(I),Y2(I)
      END DO
      CLOSE(1)
C
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine WriteProb(chi,istep,dx,dy,dt)
C
      IMPLICIT NONE
      INTEGER NN,igammay,npty,igammax,nptx,ii,jj,istep
      REAL x,y,amassx,amassy,xk,yk,sigx,sigy,anorm,ref,tra
      REAL xmin,xmax,ymin,ymax,xoff,yoff,a1,anorm1,dx,dy,dt
      REAL a2,anorm2,p1,part1,p2,part2
      COMPLEX chi, chi0, ac
      parameter(igammay=9,npty=2**igammay,NN=2)
      parameter(igammax=9,nptx=2**igammax)
      DIMENSION chi(nptx,npty,NN), chi0(nptx,npty,NN)
      common /packet/  amassx,amassy,xk,yk,sigx,sigy
      common /xy/  xmin,xmax,ymin,ymax,xoff,yoff
      common /init/ chi0
C
      anorm1=0.
      ref=0.
      tra=0.
C
      ac = (0.0, 0.0)
         x=xmin+ii*dx
         do jj=1,nptx
            y=ymin+jj*dy
            ac=ac + conjg(chi(ii,jj,1))*chi0(ii,jj,1)
            ac=ac + conjg(chi(ii,jj,2))*chi0(ii,jj,2)
            a1=chi(ii,jj,1)*conjg(chi(ii,jj,1))
            p1=a1**2
            a2=chi(ii,jj,2)*conjg(chi(ii,jj,2))
            p2=a2**2
            anorm1=anorm1+a1
            part1=part1+p1
            anorm2=anorm2+a2
            part2=part2+p2
            if (x.gt.(.0)) then
```

```
                tra=tra+a1
            else
                ref=ref+a1
            endif
          end do
        end do
        ac = ac *dx*dy
        anorm1=anorm1*dx*dy
        part1=anorm1**2/(part1*dx*dy)
          anorm2=anorm2*dx*dy
          part2=anorm2**2/(part2*dx*dy)
        anorm=anorm1
        ref=ref*dx*dy
        tra=tra*dx*dy
c        write(16,78) istep*dt,ref,tra
c        write(17,77) istep*dt,ac
          write(18,79) istep*dt,anorm1,anorm2,part1,part2
c
 77     format(3(e13.6,2x))
 78     format(4(e13.6,2x))
 79     format(6(e13.6,2x))
c
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c        SUBROUTINE PES(GS,V1,V2,V12,R0,theta)
cc
cc       Goldfield's PESs in Jacobi coords R and theta
cc       cm^{-1} and angstroms
cc
c        IMPLICIT NONE
c        REAL theta,R,R0
c        REAL*8 GS,V1,V2,V12
c        REAL*8 A1,alpha1,C6,B1,beta1,Ediss,Eso
c        REAL*8 A2,alpha2,gamm2,beta2
c        REAL*8 A12,alpha12,gamm12
c        REAL*8 kR,Re,ktheta,thetae
cc
c        R=R0*0.529177
c        A1=1.725E14
c        alpha1=8.5
c        C6=1.92E6
```

489

```fortran
c        B1=3.4E18
c        Ediss=0.0
cc26231.25
c        Eso=7603.0
cc
c        V1=A1*exp(-alpha1*R)-C6/R**6+B1*exp(-beta1*R)*theta**2
c     1   +Ediss+Eso
cc
c        A2=1.49E10
c        alpha2=5.0
c        gamm2=14.0
c        beta2=5.1
cc
c        V2=A2*(exp(-alpha2*R)*exp(-gamm2*theta**2)
c     1   +exp(-beta2*R)*theta**2)+Ediss
cc
c        A12=1.88E6
c        alpha12=2.25
c        gamm12=25.0
cc
c        V12=A12*exp(-alpha12*R)*(1.0-exp(-gamm12*(sin(theta))**2))
cc
c        RETURN
c        END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       SUBROUTINE SCHROC1(CRV,EVECT,EVALUES)
c
c      Hamiltonian Matrix Diagonalization
c
c      CRV: HERMITIAN MATRIX (INPUT)
c      EVECT: EIGENVECTORS (OUTPUT)
c      EVALUES: EIGENVALUES (OUTPUT)
c
       INTEGER N,I,J,NP
       REAL EVALUES,CRV2,EVECT2
       COMPLEX CRV,EVECT
       PARAMETER(N=2,NP=2)
       DIMENSION CRV(N,N),EVECT(N,N),EVALUES(N),E(NP)
       DIMENSION CRV2(N,N),EVECT2(N,N)
C
       DO I=1,N
           EVALUES(I)=0.0
```

```
          E(I)=0.0
          DO J=1,N
              CRV2(J,I)=CRV(J,I)
          END DO
      END DO
      CALL TRED2(CRV2,N,NP,EVALUES,E)
      CALL TQLI(EVALUES,E,N,NP,CRV2)
      CALL EIGSRT(EVALUES,CRV2,N,NP)
c
      DO I=1,N
          DO J=1,N
              EVECT(J,I)=CRV2(J,I)
          END DO
      END DO
C
      RETURN
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutines from Numerical Recipes to compute FFT
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE FOURN(DATA,NN,NDIM,ISIGN)
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION NN(NDIM),DATA(*)
      NTOT=1
      DO 11 IDIM=1,NDIM
          NTOT=NTOT*NN(IDIM)
 11   CONTINUE
      NPREV=1
      DO 18 IDIM=1,NDIM
          N=NN(IDIM)
          NREM=NTOT/(N*NPREV)
          IP1=2*NPREV
          IP2=IP1*N
          IP3=IP2*NREM
          I2REV=1
          DO 14 I2=1,IP2,IP1
              IF(I2.LT.I2REV)THEN
                  DO 13 I1=I2,I2+IP1-2,2
                      DO 12 I3=I1,IP3,IP2
                          I3REV=I2REV+I3-I2
                          TEMPR=DATA(I3)
                          TEMPI=DATA(I3+1)
                          DATA(I3)=DATA(I3REV)
```

```
                  DATA(I3+1)=DATA(I3REV+1)
                  DATA(I3REV)=TEMPR
                  DATA(I3REV+1)=TEMPI
12            CONTINUE
13          CONTINUE
          ENDIF
          IBIT=IP2/2
1         IF ((IBIT.GE.IP1).AND.(I2REV.GT.IBIT)) THEN
              I2REV=I2REV-IBIT
              IBIT=IBIT/2
              GO TO 1
          ENDIF
          I2REV=I2REV+IBIT
14      CONTINUE
        IFP1=IP1
2       IF(IFP1.LT.IP2)THEN
          IFP2=2*IFP1
          THETA=ISIGN*6.28318530717959D0/(IFP2/IP1)
          WPR=-2.D0*DSIN(0.5D0*THETA)**2
          WPI=DSIN(THETA)
          WR=1.D0
          WI=0.D0
          DO 17 I3=1,IFP1,IP1
              DO 16 I1=I3,I3+IP1-2,2
                  DO 15 I2=I1,IP3,IFP2
                      K1=I2
                      K2=K1+IFP1
                      TEMPR=SNGL(WR)*DATA(K2)-SNGL(WI)*DATA(K2+1)
                      TEMPI=SNGL(WR)*DATA(K2+1)+SNGL(WI)*DATA(K2)
                      DATA(K2)=DATA(K1)-TEMPR
                      DATA(K2+1)=DATA(K1+1)-TEMPI
                      DATA(K1+1)=DATA(K1+1)+TEMPI
15                CONTINUE
16            CONTINUE
              WTEMP=WR
              WR=WR*WPR-WI*WPI+WR
              WI=WI*WPR+WTEMP*WPI+WI
17        CONTINUE
          IFP1=IFP2
          GO TO 2
        ENDIF
        NPREV=N*NPREV
```

```
 18     CONTINUE
        RETURN
        END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c       Subroutines to compute eigenvalues and eigenvectors
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        SUBROUTINE TRED2(A,N,NP,D,E)
        IMPLICIT NONE
        INTEGER I,J,K,L,N,NP
        REAL A,D,E,H,SCALE,F,G,HH
        DIMENSION A(NP,NP),D(NP),E(NP)
        IF(N.GT.1)THEN
           DO 18 I=N,2,-1
              L=I-1
              H=0.
              SCALE=0.
              IF(L.GT.1)THEN
                 DO 11 K=1,L
                    SCALE=SCALE+ABS(A(I,K))
 11              CONTINUE
                 IF(SCALE.EQ.0.)THEN
                    E(I)=A(I,L)
                 ELSE
                    DO 12 K=1,L
                       A(I,K)=A(I,K)/SCALE
                       H=H+A(I,K)**2
 12                 CONTINUE
                    F=A(I,L)
                    G=-SIGN(SQRT(H),F)
                    E(I)=SCALE*G
                    H=H-F*G
                    A(I,L)=F-G
                    F=0.
                    DO 15 J=1,L
                       A(J,I)=A(I,J)/H
                       G=0.
                       DO 13 K=1,J
                          G=G+A(J,K)*A(I,K)
 13                    CONTINUE
                       IF(L.GT.J)THEN
                          DO 14 K=J+1,L
                             G=G+A(K,J)*A(I,K)
```

493

```
14                        CONTINUE
                       ENDIF
                       E(J)=G/H
                       F=F+E(J)*A(I,J)
                   HH=F/(H+H)
                   DO 17 J=1,L
                       F=A(I,J)
                       G=E(J)-HH*F
                       E(J)=G
                       DO 16 K=1,J
                           A(J,K)=A(J,K)-F*E(K)-G*A(I,K)
16                     CONTINUE
17                 CONTINUE
               ENDIF
           ELSE
               E(I)=A(I,L)
           ENDIF
           D(I)=H
18     CONTINUE
     ENDIF
     D(1)=0.
     E(1)=0.
     DO 23 I=1,N
         L=I-1
         IF(D(I).NE.0.)THEN
             DO 21 J=1,L
                 G=0.
                 DO 19 K=1,L
                     G=G+A(I,K)*A(K,J)
19               CONTINUE
                 DO 20 K=1,L
                     A(K,J)=A(K,J)-G*A(K,I)
20               CONTINUE
21           CONTINUE
         ENDIF
         D(I)=A(I,I)
         A(I,I)=1.
         IF(L.GE.1)THEN
             DO 22 J=1,L
                 A(I,J)=0.
                 A(J,I)=0.
22           CONTINUE
```

494

```fortran
          ENDIF
 23    CONTINUE
       RETURN
       END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       SUBROUTINE TQLI(D,E,N,NP,Z)
       IMPLICIT NONE
       INTEGER N,NP,I,K,L,M,ITER
       REAL D,E,Z,DD,G,R,S,C,P,F,B
       DIMENSION D(NP),E(NP),Z(NP,NP)
       IF (N.GT.1) THEN
           DO 11 I=2,N
               E(I-1)=E(I)
 11        CONTINUE
           E(N)=0.
           DO 15 L=1,N
               ITER=0
 1             DO 12 M=L,N-1
                   DD=ABS(D(M))+ABS(D(M+1))
 12            CONTINUE
               M=N
 2             IF(M.NE.L)THEN
                   IF(ITER.EQ.30) STOP
c PAUSE 'too many iterations!'
                   ITER=ITER+1
                   G=(D(L+1)-D(L))/(2.*E(L))
                   R=SQRT(G**2+1.)
                   G=D(M)-D(L)+E(L)/(G+SIGN(R,G))
                   S=1.
                   C=1.
                   P=0.
                   DO 14 I=M-1,L,-1
                       F=S*E(I)
                       B=C*E(I)
                       IF(ABS(F).GE.ABS(G))THEN
                           C=G/F
                           R=SQRT(C**2+1.)
                           E(I+1)=F*R
                           S=1./R
                           C=C*S
                       ELSE
                           S=F/G
```

495

```
                              R=SQRT(S**2+1.)
                              E(I+1)=G*R
                              C=1./R
                              S=S*C
                       ENDIF
                       G=D(I+1)-P
                       R=(D(I)-G)*S+2.*C*B
                       P=S*R
                       D(I+1)=G+P
                       G=C*R-B
                       DO 13 K=1,N
                           F=Z(K,I+1)
                           Z(K,I+1)=S*Z(K,I)+C*F
                           Z(K,I)=C*Z(K,I)-S*F
 13                    CONTINUE
 14               CONTINUE
                  D(L)=D(L)-P
                  E(L)=G
                  E(M)=0.
                  GO TO 1
              ENDIF
 15      CONTINUE
      ENDIF
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE EIGSRT(D,V,N,NP)
      IMPLICIT NONE
      INTEGER N,NP,I,J,K
      REAL D,V,P
      DIMENSION D(NP),V(NP,NP)
      DO 13 I=1,N-1
        K=I
        P=D(I)
          IF(D(J).GE.P)THEN
            K=J
            P=D(J)
          ENDIF
 11      CONTINUE
      IF(K.NE.I)THEN
          D(K)=D(I)
          D(I)=P
```

496

```
            DO 12 J=1,N
              P=V(J,I)
                V(J,I)=V(J,K)
                V(J,K)=P
12            CONTINUE
            ENDIF
13      CONTINUE
        RETURN
        END
ccccccccccccccccccccccccccccccccccccccccccccccccccccc
        SUBROUTINE PIKSRT(N,ARR)
        IMPLICIT NONE
        INTEGER I,J,N
        REAL ARR,A
        DIMENSION ARR(N)
        DO 12 J=2,N
          A=ARR(J)
          DO 11 I=J-1,1,-1
            IF(ARR(I).LE.A)GO TO 10
            ARR(I+1)=ARR(I)
11          CONTINUE
          I=0
10          ARR(I+1)=A
12      CONTINUE
        RETURN
        END
ccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

61.28   Problem Jortner-Marcus

1. Write a program to implement the Monte Carlo estimator of A_N, introduced by Eq. (493) for a system with 4 normal modes with $\beta = 3.0$, frequencies $\omega_{ci} = 0.75$, reduced masses $\mu_i = 1.0$, displacements $\Delta Q_i^0 = 1.0$ and $E_\lambda = 0.25$ in atomic units (i.e., $\hbar = 1.0$). Compare your results for $N_c = 10^6$ with the corresponding calculations obtained by summing the contributions of each normal mode up to $\nu = 3$.

   Download the source code from here
   Untar the tarball files by typing

```
tar -xvf mck.tar
```

   change directory by typing

```
cd mck
```

Visualize results by typing

```
gnuplot < scr
```

The visualization script generates the plot here showing the comparison of MC and straight sum calculation of log(k).

Compile by typing

```
gfortran mck.f -o mck
```

and run the program by typing

```
./mck
```

Parameter file:

Save the following parameter file as 'param.dat':

```
      IMPLICIT NONE
      INTEGER nm,nd,NMC,np,NFR
      REAL*8 gamm,hbar,Es,beta
      PARAMETER(nm=3,nd=4)
      PARAMETER (NMC=10000001,np=10,NFR=0,hbar=1.0)
      PARAMETER (beta=3.0,gamm=beta-0.1,Es=0.25)


!   nm: max quantum number nu
!   nd: # of modes
!   Es: reorganization energy
!   beta = 1/(kB T)   ! in atomic units
!   gamm: sampling function exponent
!   NFR: reading flag
!   NFR=1 when reading from input file freq_mass_displ.dat
```

Code:

Save the following code file as 'mck.f':

```
      PROGRAM mck
C
C     MC calculation of Jortner-Marcus ET rate constant:
C     nd-dimensional normal modes
C     Eq (III 4'') JCP 63: 4358-4369 (1978)
C
      INCLUDE 'param.dat'
      INTEGER i,j,l,iseed,k,nflag,nu2,nu1,nu
      INTEGER IND,ncount,kk
      REAL*8 dGmin,dGmax,dG,pi,A,Ar,re,mu,Dq,D,D2,Prod
```

498

```fortran
      REAL*8 f,f1,fn,rsum,rsum2,his,y,gn,om,w,Z,pf
      DIMENSION his(nm+1),y(nd),nu1(nd),nu2(nd),w(nd)
      DIMENSION D2(nd),mu(nd),Dq(nd)
      COMMON/counts/ncount
c
      pi=acos(-1.0)
      dGmin=-Es-1.5
      dGmax=-Es+4.5
      ncount=0
      om=.750
c
      OPEN(1,FILE="freq_mass_displ.dat")
      DO i=1,nd
         IF(NFR.EQ.1) THEN
            READ(1,*) w(i),mu(i),Dq(i)
         ELSE
            w(i)=om
            mu(i)=1.0
            Dq(i)=1.0
         END IF    ! Definition p. 4361 (above Eq. III. 1'')
         D=Dq(i)*sqrt(mu(i)*w(i)/hbar)
         D2(i)=D**2
      END DO
      CLOSE(1)
c
      OPEN(1,FILE="conv")
      DO k=1,np
         iseed=87212
         CALL srand(iseed)   ! Initialize random sequence
c
         nflag=-1            ! Sets that it's the first MC iteration
         CALL BIN(nu2,his,nflag)
         CALL MMC(nu1,nu2,nflag,w)
         nflag=1             ! No longer the first iteration
c
         dG=dGmin+(k-1)*(dGmax-dGmin)/(np-1)
         A=0.0               ! Initialize the partition function
         DO i=1,NMC          ! # of MC configs
            CALL MMC(nu1,nu2,nflag,w)
            CALL BIN(nu2,his,nflag)    ! Build histogram of nu2
            A = A + sqrt(pi/hbar**2*beta/Es)*Prod(nu2,D2)  ! Eq (III. 4'')
     1            *gn(nu2,w,dG)/fn(nu2,w)
```

499

```fortran
         END DO
         CALL A_analytic(Ar,w,dG,D2)
     1        ,log(sqrt(pi/hbar**2*beta/Es)*Ar) ! comment for nd > 4
         PRINT *, "DG,MC(A), Analy(A)", dG+Es,log(A*Z(w)/NMC)
     1        ,log(sqrt(pi/hbar**2*beta/Es)*Ar)
       END DO
       CLOSE(1)
c
       OPEN (1, FILE= "histo")
       rsum=0.0
       rsum2=0.0
       DO i=1,nm+1
          nu=i-1
          rsum=rsum+his(i)        ! check normalization
          rsum2=rsum2+f1(nu,w(1))
          WRITE(1,22) nu*1.0, his(i),f1(nu,w(1)) ! check histogram
       END DO
       CLOSE(1)
       PRINT *, "norm_his=", rsum
       PRINT *, "norm_f=", rsum2
c
 22    FORMAT(6(e13.6,2x))
       END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       SUBROUTINE BIN(nu2,his,nflag)
c
c      Histogram of configs for quantum number nu2(1)
c      nflag < 0 initializes the histogram to zero
c      nflag >= 0 increments the histogram beans with nu2(1)
c
       INCLUDE 'param.dat'
       INTEGER IND,nflag,i,nu2
       REAL*8 his
       DIMENSION  nu2(nd),his(nm+1)
c
       IF(nflag.LT.0) THEN       ! Initialize histogram
          DO i=1,nm+1
             his(i)=0.0
          END DO
       ELSE
          IND=1+nu2(1)                ! IND = 1,..., nm+1
          his(IND)=his(IND)+1.0/NMC
```

500

```fortran
        END IF
C
        RETURN
        END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        SUBROUTINE MMC(nu1,nu2,nflag,w)
C
C       Metropolis MC:
C       Given an initial config nu1, generate the next one nu2
C
        INCLUDE 'param.dat'
        INTEGER kk,k,nflag,nu1,nu2,ncount,dnu
        REAL*8 ratio,r,y,f,w
        DIMENSION nu1(nd), nu2(nd), w(nd)
C
        IF(nflag.LT.0) THEN       ! Initial config of quantum numbers nu
           DO k=1,nd
              nu1(k)=rand()*(nm+1)
           END DO
        ELSE
           DO k=1,nd
              nu2(k)=nu1(k)
           END DO
           kk=rand()*nd+1
           nu2(kk)=rand()*(nm+1)
C
           ratio=f(nu2,nu1,w)        ! Metropolis MC criteria
           r=rand()
           IF(r.GT.ratio) THEN
              DO k=1,nd
                 nu2(k)=nu1(k)
              END DO
           ELSE
              ncount=ncount+1
              DO k=1,nd
                 nu1(k)=nu2(k)
              END DO
           END IF
        END IF
        RETURN
        END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```fortran
      FUNCTION gn(nu2,w,dG)
      INCLUDE 'param.dat'
      INTEGER round,k,nu2
      REAL*8 gn,w,nut,dG
      DIMENSION nu2(nd),w(nd)
c
      nut=0.0
      DO k=1,nd ! Summing all nd-dimensional nus in the exponent
         nut=nut+hbar*w(k)*nu2(k)
      END DO
      gn=exp(-beta*(dG-Es-nut)**2/(4.0*Es))
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      FUNCTION f(nu2,nu1,w)
      INCLUDE 'param.dat'
      INTEGER k,nu1,nu2
      REAL *8 f,nut,w
      DIMENSION nu2(nd),nu1(nd),w(nd)
      nut=0.0
      DO k=1,nd
         nut=nut+gamm*hbar*w(k)*(nu2(k)-nu1(k))
      END DO
      f=exp(-nut)
      RETURN
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      FUNCTION fn(nu2,w)
      INCLUDE 'param.dat'
      INTEGER k,nu2
      REAL *8 fn,nut,w
      DIMENSION nu2(nd),w(nd)
      fn=1.0
      DO k=1,nd
         fn=fn*exp(-gamm*hbar*w(k)*nu2(k))
      END DO
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      FUNCTION f1(nu,w1)
      INCLUDE 'param.dat'
      INTEGER nu
      REAL*8 f1,w1
```

```fortran
      f1=exp(-gamm*hbar*w1*nu)
     1      /(1.0/(1.0-exp(-gamm*hbar*w1)))
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      FUNCTION Z(w)
      INCLUDE 'param.dat'
      INTEGER i,j
      REAL*8 Z,rn,w
      DIMENSION w(nd)
      Z=1.0
      DO j=1,nd
c          rn=0.0
c          DO i=1,nm+1
c             rn=rn+exp(-gamm*hbar*w(j)*(i-1))
c          END DO
         rn=1.0/(1.0-exp(-gamm*hbar*w(j)))
         Z=Z*rn
      END DO
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      FUNCTION Prod(nu2,D2)
c
c     Prod = Prod_{i=1}^nd (D^2)^nu(i)/nu(i)!
c
      INCLUDE 'param.dat'
      INTEGER nu2,i,j,fac
      REAL*8 Prod,D2
      DIMENSION nu2(nd),D2(nd)
c
      Prod=1.0
      DO i=1,nd
         fac=1                    ! Compute fac = n2(i) !
         IF(nu2(i).GT.1) THEN
            DO j=1,nu2(i)
            END DO
         END IF
         Prod=Prod*exp(-D2(i)/2.0)*D2(i)**(nu2(i))/fac
      END DO
c
      RETURN
```

503

```
        END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        SUBROUTINE A_analytic(Ar,w,dG,D2)
        INCLUDE 'param.dat'
        REAL*8 Ar,re,dG,w,pi,Prod,D2
        INTEGER nu2,i1,i2,i3,i4
        DIMENSION nu2(nd),w(nd),D2(nd)
        pi=acos(-1.0)
c
c       Assumes nd=4
c      (modified for smaller nd by commenting out do loops)
c
        Ar=0.0

        DO i1=0,3
           nu2(1)=i1
           DO i2=0,3
             nu2(2)=i2
c       i2=0.0   !  uncommented when DO loop i2 is commented out
             DO i3=0,3
                nu2(3)=i3
c       i3=0.0   !  uncommented when DO loop i3 is commented out
                DO i4=0,3
                   nu2(4)=i4
c       i4=0.0   !  uncommented when DO loop i4 is commented out
                   re=dG-Es
                   re=re-hbar*w(1)*i1
                   re=re-hbar*w(2)*i2
                   re=re-hbar*w(3)*i3
                   re=re-hbar*w(4)*i4
             Ar=Ar+Prod(nu2,D2)*exp(-beta*re**2/(4.0*Es)) ! Eq (III. 4'')
                END DO
             END DO
           END DO
        END DO
        RETURN
        END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

61.29 Computational Problem: IVR

Given a system with 84 vibrational modes with frequencies ω_j given by the second column of the file freq.txt and the anharmonic coupling constants χ_{jk} given in the file xmat.txt. Initialize the system with a quantum of vibronic excitation in state $j = 9$ and evolve it in time by using the Lanczos propagation scheme in the Krylov basis as implemented in *Expokit*, assuming that state 13 has an absorbing potential. Display the evolution of populations in states j=9,10 and 11.

Download the source code from here

Untar the tarball files by typing

```
tar -xvf ivr.tar
```

change directory by typing

```
cd ivr
```

Open ivr.m with Matlab and run it.
The visualization generates the plot showing the time dependent populations.
Code ivr.m:

```
clear all
X = importdata('./expokit_matlab/xmat.txt');
W = importdata('./expokit_matlab/freq.txt');
[n,n] = size(X);
H = zeros(n,n);
for ir=1:n
    for ic=1:n
     H(ir,ic)=0.0;
        for j=1:n
            H(ir,ic)=H(ir,ic)+W(j,2)*kronDel(ir, ic)*(kronDel(j, ic)+0.5);
            for k=1:j
                H(ir,ic)=H(ir,ic)+X(j,k)*(kronDel(j, ic)+0.5)*(kronDel(k, ic
            end
        end
        H(ir,ic)=H(ir,ic)*4.55633E-6;
    end
end
H(13,13)=H(13,13)-0.1*4.55633E-2*1i;
v = eye(n,1);
v(1)=0.0;
v(9)=0.0;
v(10)=1.0;
v(11)=0.0;
dt=2000.0;
```

```matlab
np=500;
t = linspace(0,np*dt,np);
figure
for inx = 1:np
    f1(inx)=v(9)*conj(v(9));
    f2(inx)=v(10)*conj(v(10));
    f3(inx)=v(11)*conj(v(11));
  [v,err] = expv(-1i*dt,H,v);
end
  t=t/41.3/1.0E3;
  plot(t,f1,'blue',t,f2,'red',t,f3,'black');
  xlabel('Time, ps');
  ylabel('P(t)');
  legend('mode # 9', 'mode # 10', 'mode # 11');
```

Code kronDel.m:

```matlab
function d=kronDel(j,k)

d=j==k;

end
```

Code expv.m:

```matlab
%  [w, err, hump] = expv( t, A, v, tol, m )
%  EXPV computes an approximation of w = exp(t*A)*v for a
%  general matrix A using Krylov subspace  projection techniques.
%  It does not compute the matrix exponential in isolation but instead,
%  it computes directly the action of the exponential operator on the
%  operand vector. This way of doing so allows for addressing large
%  sparse problems. The matrix under consideration interacts only
%  via matrix-vector products (matrix-free method).
%
%  w = expv( t, A, v )
%  computes w = exp(t*A)*v using a default tol = 1.0e-7 and m = 30.
%
%  [w, err] = expv( t, A, v )
%  renders an estimate of the error on the approximation.
%
%  [w, err] = expv( t, A, v, tol )
%  overrides default tolerance.
%
```

```
%  [w, err, hump] = expv( t, A, v, tol, m )
%  overrides default tolerance and dimension of the Krylov subspace,
%  and renders an approximation of the 'hump'.
%
%  The hump is defined as:
%          hump = max||exp(sA)||, s in [0,t]  (or s in [t,0] if t < 0).
%  It is used as a measure of the conditioning of the matrix exponential
%  problem. The matrix exponential is well-conditioned if hump = 1,
%  whereas it is poorly-conditioned if hump >> 1. However the solution
%  can still be relatively fairly accurate even when the hump is large
%  (the hump is an upper bound), especially when the hump and
%  ||w(t)||/||v|| are of the same order of magnitude (further details in
%  reference below).
%
%  Example 1:
%  ----------
%    n = 100;
%    A = rand(n);
%    v = eye(n,1);
%    w = expv(1,A,v);
%
%  Example 2:
%  ----------
%    % generate a random sparse matrix
%    n = 100;
%    A = rand(n);
%    for j = 1:n
%        for i = 1:n
%            if rand < 0.5, A(i,j) = 0; end;
%        end;
%    end;
%    v = eye(n,1);
%    A = sparse(A); % invaluable for a large and sparse matrix.
%
%    tic
%    [w,err] = expv(1,A,v);
%    toc
%
%    disp('w(1:10) ='); disp(w(1:10));
%    disp('err =');     disp(err);
%
%    tic
```

```
%     w_matlab = expm(full(A))*v;
%     toc
%
%     disp('w_matlab(1:10) ='); disp(w_matlab(1:10));
%     gap = norm(w-w_matlab)/norm(w_matlab);
%     disp('||w-w_matlab|| / ||w_matlab|| ='); disp(gap);
%
%  In the above example, n could have been set to a larger value,
%  but the computation of w_matlab will be too long (feel free to
%  discard this computation).
%
%  See also MEXPV, EXPOKIT.

%  Roger B. Sidje (rbs@maths.uq.edu.au)
%  EXPOKIT: Software Package for Computing Matrix Exponentials.
%  ACM - Transactions On Mathematical Software, 24(1):130-156, 1998

function [w, err, hump] = expv( t, A, v, tol, m )

[n,n] = size(A);
if nargin == 3,
  tol = 1.0e-7;
  m = min(n,30);
end;
if nargin == 4,
  m = min(n,30);
end;

anorm = norm(A,'inf');
mxrej = 10;  btol  = 1.0e-7;
gamma = 0.9; delta = 1.2;
mb    = m; t_out   = abs(t);
nstep = 0; t_new   = 0;
t_now = 0; s_error = 0;
rndoff= anorm*eps;

k1 = 2; xm = 1/m; normv = norm(v); beta = normv;
fact = (((m+1)/exp(1))^(m+1))*sqrt(2*pi*(m+1));
t_new = (1/anorm)*((fact*tol)/(4*beta*anorm))^xm;
s = 10^(floor(log10(t_new))-1); t_new = ceil(t_new/s)*s;
sgn = sign(t); nstep = 0;
```

```
w = v;
hump = normv;
while t_now < t_out
  nstep = nstep + 1;
  t_step = min( t_out-t_now,t_new );
  V = zeros(n,m+1);
  H = zeros(m+2,m+2);

  V(:,1) = (1/beta)*w;
  for j = 1:m
     p = A*V(:,j);
     for i = 1:j
        H(i,j) = V(:,i)'*p;
        p = p-H(i,j)*V(:,i);
     end;
     s = norm(p);
     if s < btol,
        k1 = 0;
        mb = j;
        t_step = t_out-t_now;
        break;
     end;
     H(j+1,j) = s;
     V(:,j+1) = (1/s)*p;
  end;
  if k1 ~= 0,
     H(m+2,m+1) = 1;
     avnorm = norm(A*V(:,m+1));
  end;
  ireject = 0;
  while ireject <= mxrej,
     mx = mb + k1;
     F = expm(sgn*t_step*H(1:mx,1:mx));
     if k1 == 0,
err_loc = btol;
        break;
     else
        phi1 = abs( beta*F(m+1,1) );
        phi2 = abs( beta*F(m+2,1) * avnorm );
        if phi1 > 10*phi2,
           err_loc = phi2;
           xm = 1/m;
```

```
        elseif phi1 > phi2,
            err_loc = (phi1*phi2)/(phi1-phi2);
            xm = 1/m;
        else
            err_loc = phi1;
            xm = 1/(m-1);
        end;
    end;
    if err_loc <= delta * t_step*tol,
        break;
    else
        t_step = gamma * t_step * (t_step*tol/err_loc)^xm;
        s = 10^(floor(log10(t_step))-1);
        t_step = ceil(t_step/s) * s;
        if ireject == mxrej,
            error('The requested tolerance is too high.');
        end;
        ireject = ireject + 1;
    end;
  end;
  mx = mb + max( 0,k1-1 );
  w = V(:,1:mx)*(beta*F(1:mx,1));
  beta = norm( w );
  hump = max(hump,beta);

  t_now = t_now + t_step;
  t_new = gamma * t_step * (t_step*tol/err_loc)^xm;
  s = 10^(floor(log10(t_new))-1);
  t_new = ceil(t_new/s) * s;

  err_loc = max(err_loc,rndoff);
  s_error = s_error + err_loc;
end;
err = s_error;
hump = hump / normv;
```