Advanced Quantum Mechanics

Chem 572a: Lecture Notes

Prof. Victor S. Batista

Room: Sterling Chemistry Laboratories (SCL) 21

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. Midterms will be on February 19 and April 23.
The final grading evaluation is the same for both undergraduate and graduate students:
homework (50%), and three mid-terms (50%).
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 162 upon request by the students via email to victor.batista@yale.edu.

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 $\hat{p} = -i\hbar\partial/\partial x$.

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 following equation*:

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

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, $E = \frac{p^2}{2m} + V(x)$.

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

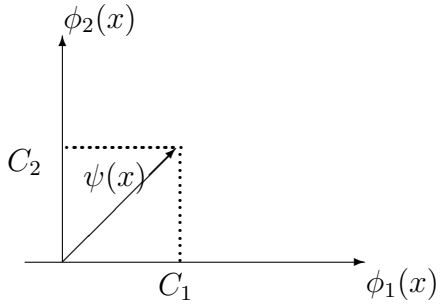
$$\hat{A}\phi_j(x) = a_j \phi_j(x), \quad \text{and} \qquad \int dx \phi_j(x)^* \phi_k(x) = \delta_{jk}.$$

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,



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.

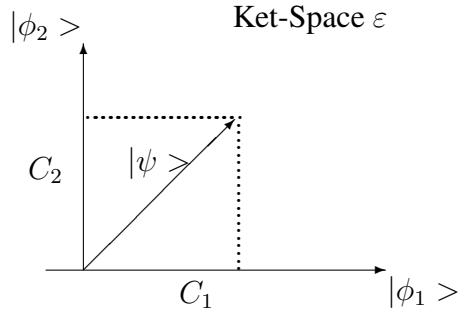
Note 2: According to the Expansion Postulate, a function $\psi(x)$ is uniquely and completely defined by the coefficients C_j, associated with its expansion in a complete set of eigenfunctions $\phi_j(x)$. However, the coefficients of such expansion would be different if the same basis functions ϕ_j 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 $\mid \psi_j >$ 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:

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

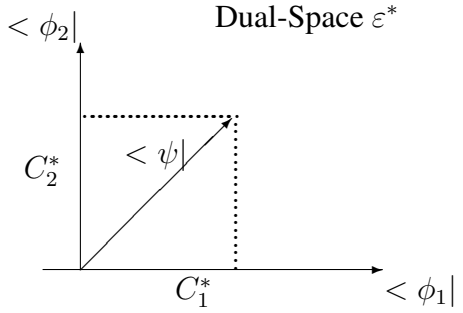
Note that functional $<\psi|$ is linear because the scalar product is a linear functional. Therefore,

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

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 $< \psi \mid$ along $< \phi_j \mid$.

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

$$|\psi >= \sum_j C_j|\phi_j >, \tag{1}$$

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

$$C_j =< \phi_j|\psi > . \tag{2}$$

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

$$|\psi >= \sum_j |\phi_j >< \phi_j|\psi > . \tag{3}$$

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 **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 **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 **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 **j** as follows:

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

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,

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

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

$$\Psi_0(x) = \int dx' c(x') \delta(x - x'), \tag{5}$$

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. (5) as follows,

$$\Psi_0(x) = \Delta \sum_{j=1}^{n} c_j \delta(x - x_j), \tag{6}$$

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. (6), 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 $c_j \equiv \langle x_j, p_j, \gamma_j|\Psi_0\rangle$.

5.1 Computational Problem 1

Write a computer program to represent the wave-packet, introduced by Eq. (4) 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:

$$\Psi_0(p) = \langle p|\Psi_0\rangle. \tag{7}$$

Inserting the closure relation $\hat{\mathbf{1}} = \int dx|x\rangle\langle x|$ in Eq. (7), we obtain that

$$\langle p|\Psi_0\rangle = \int dx \langle p|x\rangle\langle x|\Psi_0\rangle = (2\pi)^{-1/2} \int dx e^{-ipx}\langle x|\Psi_0\rangle. \tag{8}$$

is the Fourier transform of the initial state. The second equality in Eq. (8) was obtained by using:

$$\langle x|p\rangle = (2\pi)^{-1/2} e^{ipx}, \tag{9}$$

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 [FFT] 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. (4), 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:

$$V(\hat{x}) = \frac{1}{2} m\omega^2 (\hat{x} - \bar{x})^2. \tag{10}$$

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

$$V(\hat{x})\Psi_0(x) = V(x)\Psi_0(x) \equiv \tilde{\Psi}_0(x). \tag{11}$$

Since $\tilde{\Psi}_0(x)$ is just another function, Eq. (11) 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. (10) for the

initial wave-packet, introduced by Eq. (4), 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,

$$G(x) = \langle x|\hat{p}|\Psi_0\rangle = -i\nabla\Psi_0(x). \tag{12}$$

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:

$$G(x_j) = -i\frac{\Psi_0(x_{j+1}) - \Psi_0(x_{j-1})}{2\Delta}. \tag{13}$$

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. (12),

$$G(x) = \langle x|\hat{p}|\Psi_0\rangle = \int dp\langle x|\hat{p}|p\rangle\langle p|\Psi_0\rangle = (2\pi)^{-1/2}\int dp e^{ipx}p\langle p|\Psi_0\rangle, \tag{14}$$

since $\langle p|\Psi_0\rangle$ is defined, according to Eq. (8), as the Fourier transform of the initial state. Note that the second equality of Eq. (14) is obtained by introducing the substitution

$$\langle x|p\rangle = (2\pi)^{-1/2}e^{ix\hat{p}}. \tag{15}$$

While Eq. (14) 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. (4), 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. (10) and compare your result with the zero-point energy $E_0 = \omega/2$.

6 Heisenberg Representation

Most of the problems of interest in Chemistry have equations that are too complicated to be solved analytically. This observation has been stated by Paul Dirac as follows: *The underlying physical*

laws necessary for the mathematical theory of a large part of Physics and the whole of Chemistry are thus completed and the difficulty is only that exact application of these laws leads to the equations much too complicated to be soluble. It is, therefore, essential, to introduce numerical and approximate methods (e.g., *perturbation methods* and *variational methods*).

In this section, we describe the matrix representation, introduced by Heisenberg, which is most useful for numerical methods to solve the eigenvalue problem,**R4(124) R3(240)**

$$\hat{H}|\psi_l\rangle = E_l|\psi_l\rangle, \tag{16}$$

for an arbitrary state $|\psi_l\rangle$ of a system (*e.g.*, an atom, or molecule) expanded in a basis set $\{\phi_j\}$, as follows:

$$|\psi_l\rangle = \sum_j C_l^{(j)}|\phi_j\rangle, \tag{17}$$

where $C_l^{(j)} = \langle\phi_j|\psi_l\rangle$, and $\langle\phi_j|\phi_k\rangle = \delta_{jk}$. Substituting Eq. (17) into Eq. (16) we obtain:

$$\sum_j \hat{H}|\phi_j\rangle C_l^{(j)} = \sum_j E_l C_l^{(j)}|\phi_j\rangle.$$

Applying the functional $\langle\phi_k|$ to both sides of this equation, we obtain:

$$\sum_j \langle\phi_k|\hat{H}|\phi_j\rangle C_l^{(j)} = \sum_j E_l\langle\phi_k|\phi_j\rangle C_l^{(j)}, \tag{18}$$

where $\langle\phi_k|\phi_j\rangle = \delta_{kj}$ and k = 1, 2, ..., n.
Introducing the notation $H_{kj} = \langle\phi_k|\hat{H}|\phi_j\rangle$ we obtain,

$$\begin{array}{ll}(k=1) & \rightarrow \\ (k=2) & \rightarrow \\ & \cdots \\ (k=n) & \rightarrow \end{array} \begin{cases} H_{11}C_l^{(1)} + H_{12}C_l^{(2)} + H_{13}C_l^{(3)} + ... + H_{1n}C_l^{(n)} = E_lC_l^{(1)} + 0C_l^{(2)} + ... + 0C_l^{(n)}, \\ H_{21}C_l^{(1)} + H_{22}C_l^{(2)} + H_{23}C_l^{(3)} + ... + H_{2n}C_l^{(n)} = 0C_l^{(1)} + E_lC_l^{(2)} + ... + 0C_l^{(n)}, \\ ... \\ H_{n1}C_l^{(1)} + H_{n2}C_l^{(2)} + H_{n3}C_l^{(3)} + ... + H_{nn}C_l^{(n)} = 0C_l^{(1)} + 0C_l^{(2)} + ... + E_lC_l^{(n)}, \end{cases} \tag{19}$$

that can be conveniently written in terms of *matrices* and *vectors* as follows,

$$\begin{bmatrix} H_{11} & H_{12} & ... & H_{1n} \\ H_{21} & H_{22} & ... & H_{2n} \\ ... & & & \\ H_{n1} & H_{n2} & ... & H_{nn} \end{bmatrix} \begin{bmatrix} C_l^{(1)} \\ C_l^{(2)} \\ ... \\ C_l^{(n)} \end{bmatrix} = \begin{bmatrix} E_l & 0 & ... & 0 \\ 0 & E_l & ... & 0 \\ ... & & & \\ 0 & 0 & ... & E_l \end{bmatrix} \begin{bmatrix} C_l^{(1)} \\ C_l^{(2)} \\ ... \\ C_l^{(n)} \end{bmatrix}. \tag{20}$$

This is the *Heisenberg representation* of the eigenvalue problem introduced by Eq. (16). According to the *Heisenberg representation*, also called *matrix representation*, the *ket* $|\psi_l\rangle$ is represented by the *vector* C_l, with components $C_l^{(j)} = \langle\phi_j|\psi_l\rangle$, with $j = 1, ..., n$, and the *operator* $\hat{H}$ is represented by the *matrix* $\mathbf{H}$ with elements $H_{jk} = \langle\phi_j|\hat{H}|\phi_k\rangle$.

The expectation value of the Hamiltonian,

$$\langle\psi_l|H|\psi_l\rangle = \sum_j \sum_k C_l^{(k)*} \langle\phi_k|\hat{H}|\phi_j\rangle C_l^{(j)},$$

can be written in the matrix representation as follows,

$$\langle\psi_l|H|\psi_l\rangle = C_l^{\dagger}\mathbf{H}C_l = \begin{bmatrix} C_l^{(1)*} & C_l^{(2)*} & ... & C_l^{(n)*} \end{bmatrix} \begin{bmatrix} H_{11} & H_{12} & ... & H_{1n} \\ H_{21} & H_{22} & ... & H_{2n} \\ ... & & & \\ H_{n1} & H_{n2} & ... & H_{nn} \end{bmatrix} \begin{bmatrix} C_l^{(1)} \\ C_l^{(2)} \\ ... \\ C_l^{(n)} \end{bmatrix}.$$

Note:
(1) It is important to note that according to the matrix representation the *ket-vector* $|\psi_l\rangle$ is represented by a column vector with components $C_l^{(j)} = \langle\phi_j|\psi_l\rangle$, and the *bra-vector* $\langle\psi_l|$ is represented by a *row vector* with components $C_l^{(j)*}$.
(2) If an *operator* is hermitian (e.g., $\hat{H}$), it is represented by a *hermitian matrix* (i.e., a matrix where any two elements which are symmetric with respect to the principal diagonal are complex conjugates of each other). The diagonal elements of a hermitian matrix are real numbers, therefore, its eigenvalues are real.
(3) The eigenvalue problem has a non-trivial solution *only* when the determinant $\det[\mathbf{H} - \hat{\mathbf{1}}E]$ vanishes:

$$\det[\mathbf{H} - \hat{\mathbf{1}}E] = 0, \quad \text{where } \hat{\mathbf{1}} \text{ is the unity matrix.}$$

This equation has n roots, which are the eigenvalues of $\mathbf{H}$.
(3) Finally, we note that the matrix of column eigenvectors $\mathbf{C}$ satisfy the equation, $\mathbf{HC} = \mathbf{CE}$, where $\mathbf{E}$ is the diagonal matrix of eigenvalues:

$$\begin{bmatrix} H_{11} & H_{12} & ... & H_{1n} \\ H_{21} & H_{22} & ... & H_{2n} \\ ... & & & \\ H_{n1} & H_{n2} & ... & H_{nn} \end{bmatrix} \begin{bmatrix} C_1^{(1)} & C_2^{(1)} & \cdots & C_n^{(1)} \\ C_1^{(2)} & C_2^{(2)} & \cdots & C_n^{(2)} \\ ... & ... & ... & ... \\ C_1^{(n)} & C_2^{(n)} & \cdots & C_n^{(n)} \end{bmatrix} = \begin{bmatrix} C_1^{(1)} & C_2^{(1)} & \cdots & C_n^{(1)} \\ C_1^{(2)} & C_2^{(2)} & \cdots & C_n^{(2)} \\ ... & ... & ... & ... \\ C_1^{(n)} & C_2^{(n)} & \cdots & C_n^{(n)} \end{bmatrix} \begin{bmatrix} E_1 & 0 & ... & 0 \\ 0 & E_2 & ... & 0 \\ ... & & & \\ 0 & 0 & ... & E_n \end{bmatrix}. \tag{21}$$

7 Fourier Grid Hamiltonian

The goal of this section is to introduce the Fourier grid Hamiltonian (FGH),

$$\begin{aligned} H(j,j') &= V(x_j)\langle x_j|x_j'\rangle + \frac{\Delta x \Delta p}{2\pi\hbar} \sum_{k=1}^{n_p} e^{\frac{i}{\hbar}(x_{j'}-x_j)p_k} \frac{p_k^2}{2m}, \\ &= V(x_j)\delta_{jj'} + \frac{\Delta x \Delta p}{2\pi\hbar} \sum_{k=1}^{n_p} e^{\frac{i}{\hbar}(x_{j'}-x_j)p_k} \frac{p_k^2}{2m}, \end{aligned} \tag{22}$$

as described by Marston and Balint-Kurti [*J. Chem. Phys.* (1989) **91**:3571-3576] . We write the Hamiltonian as a matrix in the representation of equally spaced delta functions $\delta(x - x_j)$, with coordinates

$$x_j = (j - n_x/2)\Delta_x, \tag{23}$$

where $\Delta_x = (x_{max} - x_{min})/n_x$ and $j = 1$–n_x, and momenta $p_k = \Delta p(k - n_p/2)$ with $\Delta p = 2\pi/(x_{max} - x_{min})$. Equation (22) is derived by writing the kinetic energy in the basis of plane waves, as follows:

$$\begin{aligned}
\langle x_l|\hat{T}|x_j\rangle &= \langle x_l|\frac{\hat{p}^2}{2m}|x_j\rangle, \\
&= \int dp \int dp' \langle x_l|p'\rangle\langle p'|\frac{\hat{p}^2}{2m}|p\rangle\langle p|x_j\rangle, \\
&= \int dp \int dp' \langle x_l|p'\rangle\frac{p^2}{2m}\langle p'|p\rangle\langle p|x_j\rangle, \\
&= \int dp\langle x_l|p\rangle\frac{p^2}{2m}\langle p|x_j\rangle = \frac{\Delta x}{2\pi\hbar}\int dp e^{\frac{i}{\hbar}(x_l - x_j)p}\frac{p^2}{2m}, \\
&= \frac{\Delta x \Delta p}{2\pi\hbar}\sum_{k=1}^{n_p} e^{\frac{i}{\hbar}(x_l - x_j)p_k}\frac{p_k^2}{2m},
\end{aligned} \tag{24}$$

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 $\Delta x\langle x_j|x_k\rangle = \delta_{jk}$.

7.1 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 5 eigenvalues and eigenfunctions of the particle in the box with $m = a = 1$. Compare your numerical and analytical solutions. Modify the potential to obtain the analogous eigenstates for the Harmonic oscillator introduced by Eq. (10) with $m = 1$ and $\omega = 1$. Verify that the eigenvalues are $E(\nu) = (1/2 + \nu)\hbar\omega$, $\nu = 0$–4.

The link (http://ursula.chem.yale.edu/∼batista/classes/vvv/pbox.m) provides a Matlab solution to the FGH computational assignment.

The link http://ursula.chem.yale.edu/∼batista/classes/vvv/M1.pdf provides a Matlab tutorial with a detailed explanation of the solution to the computational assignment, prepared by Dr. Videla.

The link (http://ursula.chem.yale.edu/∼batista/classes/vvv/hbox.m) provides the corresponding Matlab solution to the harmonic well potential.

The link (http://ursula.chem.yale.edu/∼batista/classes/vvv/2DFGH.tar) provides the corresponding Matlab solution to the 2-dimensional harmonic well potential.

8 Variational Theorem

The expectation value of the Hamiltonian, computed with any trial wave function, is always higher or equal than the energy of the ground state. Mathematically,

$$< \psi|\hat{H}|\psi > \geq E_0,$$

where $\hat{H}\phi_j = E_j\phi_j$.
Proof: $\psi = \sum_j C_j\phi_j$, where $\{\phi_j\}$ is a basis set of orthonormal eigenfunctions of the Hamiltonian $\hat{H}$.

$$\begin{aligned} < \psi|\hat{H}|\psi > &= \sum_j \sum_k C_k^* C_j < \phi_k|\hat{H}|\phi_j >, \\ &= \sum_j \sum_k C_k^* C_j E_j \delta_{kj}, \\ &= \sum_j C_j^* C_j E_j \qquad \geq \qquad E_0 \quad \sum_j C_j^* C_j, \end{aligned}$$

where, $\sum_j C_j^* C_j = 1$.
Variational Approach: Starting with an initial trial wave function ψ defined by the expansion coefficients $\{C_j^{(0)}\}$, the optimum solution of an arbitrary problem described by the Hamiltonian $\hat{H}$ can be obtained by minimizing the expectation value $< \psi|\hat{H}|\psi >$ with respect to the expansion coefficients.

Having found the ground state $|\psi_0\rangle$, we can obtain the first excited state analogously by adding to the expectation value of the energy a penalty term proportional to the norm of the overlap between the ground and variational states, $\langle\psi|\hat{H}|\psi\rangle + \gamma|\langle\psi_0|\psi\rangle|^2$. Higher energy states $|\psi_n\rangle$ are found analogously by minimization of the cost function $\langle\psi|\hat{H}|\psi\rangle + \gamma\sum_{j=0}^{n-1}|\langle\psi_0|\psi\rangle|^2$.
Exercise: Write a program to solve the time independent Schrödinger equation by using the FGH method in conjunction with the variational theorem and apply it to find the first 5 eigenvalues and eigenfunctions of the particle in the box with $m = a = 1$. Compare your numerical and analytical solutions. Modify the potential to obtain the analogous eigenstates for the Harmonic oscillator introduced by Eq. (10) with $m = 1$ and $\omega = 1$. Verify that the eigenvalues are $E(\nu) = (1/2+\nu)\hbar\omega$, $\nu = 0$–4.

The link (http://ursula.chem.yale.edu/∼batista/classes/vvv/VT570.tar) , provides a Matlab implementation of the variational method as applied to the calculation of the ground and excited states of a harmonic well.

The link http://ursula.chem.yale.edu/∼batista/classes/vvv/M2.pdf , provides a detailed description of the solution to the computational assignment, prepared by Dr. Pablo Videla.

9 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:

$$\Psi_t(x) = \int dx' \langle x|e^{-i\hat{H}t}|x'\rangle\langle x'|\Psi_0\rangle, \tag{25}$$

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

$$\langle x|e^{-i\hat{H}t}|x'\rangle = \sqrt{\frac{m\omega}{2\pi \text{sinh}(it\omega)}} \text{exp}\left(-\frac{m\omega}{2\text{sinh}(\omega it)}[(x^2+x'^2)\text{cosh}(\omega it) - 2xx']\right). \tag{26}$$

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. (25) as follows,

$$\Psi_{t_{k+1}}(x) = \int dx' \langle x|e^{-i\hat{H}\tau}|x'\rangle\langle x'|\Psi_{t_k}\rangle. \tag{27}$$

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,

$$e^{-i\hat{H}\tau} = e^{-iV(\hat{x})\tau/2} e^{-i\hat{p}^2\tau/(2m)} e^{-iV(\hat{x})\tau/2} + O(\tau^3), \tag{28}$$

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

9.1 Computational Problem 5

Expand the exponential operators in both sides of Eq. (28) and show that the Trotter expansion is accurate to second order in powers of τ.

Substituting Eq. (28) into Eq. (27) and inserting the closure relation $\hat{\mathbf{1}} = \int dp|p\rangle\langle p|$ gives,

$$\Psi_{t_{k+1}}(x) = \int dp \int dx' e^{-iV(\hat{x})\tau/2} \langle x|p\rangle e^{-ip^2\tau/(2m)} \langle p|x'\rangle e^{-iV(x')\tau/2} \Psi_{t_k}(x'). \tag{29}$$

By substituting $\langle p|x'\rangle$ and $\langle x|p\rangle$ according to Eqs. (9) and (15), respectively, we obtain:

$$\Psi_{t_{k+1}}(x) = e^{-iV(\hat{x})\tau/2} \frac{1}{\sqrt{2\pi}} \int dp e^{ixp} e^{-ip^2\tau/(2m)} \frac{1}{\sqrt{2\pi}} \int dx' e^{-ipx'} e^{-iV(x')\tau/2} \Psi_{t_k}(x'). \tag{30}$$

According to Eq. (30), 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,

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

9.2 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. (10). Compare the resulting propagated state with the analytic solution obtained by substituting Eq. (26) into Eq. (25).

9.3 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. (26) into Eq. (25). 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:

$$\begin{aligned} p_{j+1} &= p_j + (F(x_j) + F(x_{j+1}))\tau/2 \\ x_{j+1} &= x_j + p_j\tau/m + F(x_j)\tau^2/(2m). \end{aligned} \tag{31}$$

9.4 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.

9.5 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.

9.6 Imaginary time propagation

Note that with the variable substitution $\tau \rightarrow -it$, with real t, the time evolution operator becomes a decaying exponential $e^{-\hat{H}t/\hbar}$ that reduces the amplitude of the initial wavepacket $\Psi_0(x) = \sum_j c_j \phi_j(x)$, as follows:

$$\Psi_t(x) = e^{-i\hat{H}\tau/\hbar}\Psi_0(x) = \sum_j c_j e^{-E_j t}\phi_j(x), \tag{32}$$

where $\hat{H}\phi_j(x) = E_j\phi_j(x)$. Terms with higher E_j are reduced more than those with smaller E_j. After renormalizing the resulting wavefunction $\Psi_t(x)$ (by dividing it by the square root of its norm), we get a state enriched with low energy components. The imaginary time propagation and renormalization procedure can be repeated several times until the function stops changing since it composed solely by the ground state $\phi_0(x)$, after removal of all other components (of higher energies) at a faster rate.

Having found ϕ_0, we can proceed to find ϕ_1 as done for ϕ_0 but including orthogonalization relative to ϕ_0, $\Psi_t(x) \rightarrow \Psi_t(x) - \langle\phi_0|\Psi_t\rangle\phi_0(x)$, after each propagation step, right before renormalization. Higher energy states are found analogously, by orthogonalization of the propagated state relative to all previously found eigenstates.

9.7 Ehrenfest Dynamics

The goal of this section is to show that the expectation values $\langle\hat{x}\rangle = \langle\psi|\hat{x}|\psi\rangle$ and $\langle\hat{p}\rangle = \langle\psi|\hat{p}|\psi\rangle$ are conjugate variables in the sense that they evolve according to the classical equations of motion

(*i.e.*, Hamilton's equaitons):

$$\begin{aligned}\frac{d}{dt}\langle\hat{x}\rangle &= \langle\frac{\partial\hat{H}}{\partial\hat{p}}\rangle \\ &= \frac{\langle\hat{p}\rangle}{m}, \\ \frac{d}{dt}\langle p\rangle &= -\langle\frac{\partial\hat{H}}{\partial\hat{x}}\rangle \\ &= -\langle V'\rangle,\end{aligned} \tag{33}$$

where $\hat{H} = \hat{p}^2/2m + V(\hat{x})$. This remarkable result, introduced by Eq. (33), is known as *Ehrenfest's theorem* and can be demonstrated, as follows.

First, we show that since ψ evolves according to the Schrödinger equation:

$$i\hbar\frac{\partial\psi}{\partial t} = \hat{H}\psi, \tag{34}$$

then

$$\langle\hat{p}\rangle = m\frac{d}{dt}\langle\hat{x}\rangle. \tag{35}$$

Using integration by parts, we obtain:

$$\begin{aligned}\langle\hat{p}\rangle &= -i\hbar\langle\psi|\frac{\partial}{\partial x}|\psi\rangle, \\ &= -\frac{i\hbar}{2}\langle\psi|\frac{\partial}{\partial x} + \frac{\partial}{\partial x}|\psi\rangle, \\ &= -\frac{i\hbar}{2}\int dx\left[\psi^*\frac{\partial\psi}{\partial x} + \psi^*\frac{\partial\psi}{\partial x}\right], \\ &= -\frac{i\hbar}{2}\int dx\left[\psi^*\frac{\partial\psi}{\partial x} - \psi\frac{\partial\psi^*}{\partial x}\right], \\ &= m\int j\,dx, \\ &= -m\int x\frac{\partial j}{\partial x}\,dx,\end{aligned} \tag{36}$$

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,

$$\frac{d}{dt}\psi^*\psi + \frac{\partial j}{\partial x} = 0. \tag{37}$$

Therefore,

$$\begin{aligned}\langle\hat{p}\rangle &= m\int x\frac{d}{dt}\psi^*\psi\,dx, \\ &= m\frac{d}{dt}\int\psi^* x\psi\,dx, \\ &= m\frac{d}{dt}\langle\hat{x}\rangle.\end{aligned} \tag{38}$$

Next, we show that

$$\frac{d}{dt}\langle\hat{p}\rangle = -\langle V'\rangle, \tag{39}$$

by substituting Eq. (314) into Eq. (36) and integrating by parts, as follows:

$$\begin{aligned}
\frac{d}{dt}\langle\hat{p}\rangle &= m\frac{d}{dt}\int_{-\infty}^{\infty} dx\, j, \\
&= -\frac{i\hbar}{2}\int_{-\infty}^{\infty} dx \left[\frac{d\psi^*}{dt}\frac{\partial\psi}{\partial x} + \psi^*\frac{\partial}{\partial x}\frac{d\psi}{dt} - \frac{d\psi}{dt}\frac{\partial\psi^*}{\partial x} - \psi\frac{\partial}{\partial x}\frac{d\psi^*}{dt}\right] \\
&= i\hbar\int_{-\infty}^{\infty} dx \left[\frac{d\psi}{dt}\frac{\partial\psi^*}{\partial x} - \frac{d\psi^*}{dt}\frac{\partial\psi}{\partial x}\right] \\
&= \int_{-\infty}^{\infty} dx \left[-\frac{\hbar^2}{2m}\frac{\partial^2\psi}{\partial x^2}\frac{\partial\psi^*}{\partial x} + V\psi\frac{\partial\psi^*}{\partial x} + c.c.\right] \\
&= \int_{-\infty}^{\infty} dx \left[-\frac{\hbar^2}{2m}\frac{\partial}{\partial x}\left(\frac{\partial\psi}{\partial x}\frac{\partial\psi^*}{\partial x}\right) + V\left(\psi\frac{\partial\psi^*}{\partial x} + \psi^*\frac{\partial\psi}{\partial x}\right)\right], \\
&= \int_{-\infty}^{\infty} dx V\left(\psi\frac{\partial\psi^*}{\partial x} + \psi^*\frac{\partial\psi}{\partial x}\right),
\end{aligned} \tag{40}$$

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

$$\begin{aligned}
\frac{d}{dt}\langle\hat{p}\rangle &= \int_{-\infty}^{\infty} dx \frac{\partial\psi^*\psi}{\partial x} V, \\
&= -\int_{-\infty}^{\infty} dx \psi^*\frac{\partial V}{\partial x}\psi.
\end{aligned} \tag{41}$$

9.8 Exercise: Analytical versus SOFT Propagation

1. Write a code to simulate the SOFT propagation of a wavepacket bouncing back and forth on a harmonic well, as described by the Hamiltonian $H = p^2/(2 * m) + V(x)$, with $m = 1$ and $V(x) = 0.5 * x^2$ after initializing the state according to the ground state displaced from its equilibrium position, as follows: $\psi(x, 0) = exp(-(x-1)^2/2)/\sqrt[4]{\pi}$.
2. Compute the first 5 eigenvalues E_n and eigenstates Φ_n with n =1–5 of the harmonic oscillator by using imaginary time propagation.
3. Compare the quantum dynamics simulation based on the SOFT method to the corresponding simulation based on the superposition of the first 5 eigenstates). Note that both methods agree with each other, although the SOFT method by-passes the need of computing the eigenvalues and eigenfunctions of the Hamiltonian.

Solution: The link (http://ursula.chem.yale.edu/∼batista/classes/vvv/HO570.tar) provides a Matlab implementation of the SOFT method to the simulation of evolution of a wavepacket in a harmonic well in real time, as solved for Exercise 5. In addition, the Matlab code implements the SOFT

propagation method to find the lowest 5 eigenstates of the harmonic oscillator by 'evolution' in imaginary time.

9.9 Exercise: Real and Imaginary Time Evolution

1. Write a Matlab code to simulate the evolution of a wavepacket bouncing back and forth on a harmonic well, described by the Hamiltonian $H = p^2/(2 * m) + V(x)$, with $V(x) = 0.5 * x^2$ after initializing the state according to the ground state displaced from its equilibrium position, as follows: $\psi(x, 0) = exp(-(x-1)^2/2)/\sqrt[4]{\pi}$.

2. Compute the expectation values of position and momentum as a function of time $x(t)$ and $p(t)$ and compare them to the corresponding classical values obtained by integrating Hamilton's equation with the Velocity-Verlet algorithm:

$$\begin{aligned} p_{j+1} &= p_j + (F(x_j) + F(x_{j+1}))\tau/2, \\ x_{j+1} &= x_j + p_j\tau/m + F(x_j)\tau^2/(2m), \end{aligned} \tag{42}$$

with $x_0 = 1$ and $p_0 = 0$ the initial position and momentum of the harmonic oscillator and x_j and p_j the position and momentum at time $t = j * \tau$, while $F(x_j) = -V'(x_j) = -x_j$.

3. Compute the expectation values of position and momentum as a function of time $x(t)$ and $p(t)$ and compare them to the Ehrenfest trajectory obtained by integrating Hamilton's equation, using mean force:

$$\begin{aligned} \langle p\rangle_{j+1} &= \langle p\rangle_j + (\langle F(x)\rangle_j + \langle F(x)\rangle_{j+1})\tau/2, \\ \langle x\rangle_{j+1} &= \langle x\rangle_j + \langle p\rangle_j\tau/m + \langle F(x)\rangle_j\tau^2/(2m), \end{aligned} \tag{43}$$

with $\langle x\rangle_0 = 1$ and $\langle p\rangle_0 = 0$ the initial position and momentum of the harmonic oscillator and $\langle x\rangle_j$ and $\langle p\rangle_j$ the mean position and momentum at time $t = j*\tau$, while $\langle F(x)\rangle_j = -\langle V'(x)\rangle_j = -\langle x\rangle_j$.

4. Find the ground state of the harmonic well by propagating the wavepacket in imaginary time (*i.e.*, using the propagation time increment $\tau = -it$, with real t) and renormalizing the wave function after each propagation step.

5. Find the first excited state of the harmonic well by propagating the wavepacket in imaginary time (*i.e.*, using the propagation time increment $\tau = -it$, with real t), projecting out the ground state component and renormalizing the wave function after each propagation step.

6. Find the first 9 excited states, iteratively, by imaginary time propagation as in item 4, projecting out lower energy states and renormalizing after each propagation step.

7. Change the potential to that of a Morse oscillator $V(x) = De(1 - exp(-a(x - x_e)))^2$, with $x_e = 0$, $D_e = 8$, 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. Compare the expectation values $x(t)$ and $p(t)$ to the corresponding classical and Ehrenfest trajectories obtained according to the Velocity-Verlet algorithm.

Solution: The link (http://ursula.chem.yale.edu/∼batista/classes/vvv/HO570.tar) provides a Matlab implementation of the SOFT method as applied to the simulation of evolution of a wavepacket in a harmonic well in real time. In addition, the Matlab code implements the SOFT propagation method to find the lowest 10 eigenstates of the harmonic oscillator by 'evolution' in imaginary time.

10 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,

$$\hat{H} = \hat{p}^2/(2m) + \hat{V}, \tag{44}$$

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

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

$$|\Psi(\mathbf{x};t)\rangle = \varphi_1(\mathbf{x};t)|1\rangle + \varphi_2(\mathbf{x};t)|2\rangle, \tag{45}$$

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,

$$e^{-i\hat{H}2\tau} \approx e^{-i\frac{\hat{p}^2}{2m}\tau}e^{-iV(\hat{\mathbf{x}})2\tau}e^{-i\frac{\hat{p}^2}{2m}\tau} \approx e^{-i\frac{\hat{p}^2}{2m}\tau}e^{-iV_0(\hat{\mathbf{x}})\tau}e^{-iV_c(\hat{\mathbf{x}})2\tau}e^{-iV_0(\hat{\mathbf{x}})\tau}e^{-i\frac{\hat{p}^2}{2m}\tau}, \tag{46}$$

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,

$$\begin{pmatrix} \varphi_1'(\mathbf{x};t+\tau) \\ \varphi_2'(\mathbf{x};t+\tau) \end{pmatrix} = \begin{pmatrix} e^{-i\frac{\hat{\mathbf{p}}^2}{2m}\tau} & 0 \\ 0 & e^{-i\frac{\hat{\mathbf{p}}^2}{2m}\tau} \end{pmatrix} \begin{pmatrix} \varphi_1(\mathbf{x};t) \\ \varphi_2(\mathbf{x};t) \end{pmatrix}. \tag{47}$$

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

$$\begin{pmatrix} \varphi_1''(\mathbf{x};t+\tau) \\ \varphi_2''(\mathbf{x};t+\tau) \end{pmatrix} = \mathbf{M} \begin{pmatrix} \varphi_1'(\mathbf{x};t+\tau) \\ \varphi_2'(\mathbf{x};t+\tau) \end{pmatrix}, \tag{48}$$

  with

$$\mathbf{M} \equiv \mathbf{L}^{-1} \begin{pmatrix} e^{-iE_1(x)\tau} & 0 \\ 0 & e^{-iE_2(x)\tau} \end{pmatrix} \mathbf{L}, \tag{49}$$

  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,

$$\mathbf{M} \equiv \begin{pmatrix} e^{-i\hat{V}_1(\hat{\mathbf{x}})2\tau}\cos(2V_c(\hat{\mathbf{x}})\tau) & -i\,\sin(2V_c(\hat{\mathbf{x}})\tau)\, e^{-i(\hat{V}_1(\hat{\mathbf{x}})+\hat{V}_2(\hat{\mathbf{x}}))\tau} \\ -i\,\sin(2V_c(\hat{\mathbf{x}})\tau)\, e^{-i(\hat{V}_1(\hat{\mathbf{x}})+\hat{V}_2(\hat{\mathbf{x}}))\tau} & \cos(2V_c(\hat{\mathbf{x}})\tau)\, e^{-i\hat{V}_2(\hat{\mathbf{x}})2\tau} \end{pmatrix}. \quad (50)$$

- 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:

$$\begin{pmatrix} \varphi_1(\mathbf{x};t+2\tau) \\ \varphi_2(\mathbf{x};t+2\tau) \end{pmatrix} = \begin{pmatrix} e^{-i\frac{\hat{\mathbf{p}}^2}{2m}\tau} & 0 \\ 0 & e^{-i\frac{\hat{\mathbf{p}}^2}{2m}\tau} \end{pmatrix} \begin{pmatrix} \varphi_1''(\mathbf{x};t+\tau) \\ \varphi_2''(\mathbf{x};t+\tau) \end{pmatrix}. \quad (51)$$

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

10.1 Problem 10

(a) Derive Eq. (50) by considering that,

$$e^{-i\mathbf{V}_c 2\tau} = \mathbf{D}^\dagger \begin{pmatrix} e^{iV_c(\mathbf{x})2\tau} & 0 \\ 0 & e^{-iV_c(\mathbf{x})2\tau} \end{pmatrix} \mathbf{D}, \quad (52)$$

with

$$\mathbf{D} = \mathbf{D}^\dagger \equiv \begin{pmatrix} -1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}, \quad (53)$$

since

$$e^{-i\mathbf{V}_c 2\tau} = \mathbf{1} + (-i\mathbf{V}_c 2\tau) + \frac{1}{2!}(-i\mathbf{V}_c 2\tau)^2 + ..., \quad (54)$$

and

$$\mathbf{V}_c \equiv \begin{pmatrix} 0 & V_c(\mathbf{x}) \\ V_c(\mathbf{x}) & 0 \end{pmatrix} = \mathbf{D}^\dagger \begin{pmatrix} -V_c(\mathbf{x}) & 0 \\ 0 & V_c(\mathbf{x}) \end{pmatrix} \mathbf{D}, \quad (55)$$

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

10.2 Problem 11

Derive Eq. (49) by writing the matrix V in the basis of adiabatic eigenstates

$$\begin{aligned} \phi_1(x) &= L_{11}(x)|1\rangle + L_{21}(x)|2\rangle, \\ \phi_2(x) &= L_{12}(x)|1\rangle + L_{22}(x)|2\rangle, \end{aligned} \quad (56)$$

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

$$e^{-i\mathbf{V}2\tau} = \mathbf{1} + (-i\mathbf{V}2\tau) + \frac{1}{2!}(-i\mathbf{V}2\tau)^2 + ..., \quad (57)$$

show that in the adiabatic representation

$$e^{-i\mathbf{V}2\tau} = \begin{pmatrix} e^{-iE_1(x)2\tau} & 0 \\ 0 & e^{-iE_2(x)2\tau} \end{pmatrix}. \tag{58}$$

Finally, show that the diagonal matrix introduced by Eq. (58) can be rotated to the representation of diabatic states $|1\rangle$, $|2\rangle$ according to the similarity transformation

$$\mathbf{L}^{-1} \begin{pmatrix} e^{-iE_1(x)\tau} & 0 \\ 0 & e^{-iE_2(x)\tau} \end{pmatrix} \mathbf{L}. \tag{59}$$

10.3 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. (49). 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. (4). Use $x_0 = -2.2$, $p_0 = 0$, $m = 1$, $\omega = 1$ and two coupled potential energy surfaces described by the potential energy matrix

$$V = \begin{pmatrix} V_1(x) & \delta \\ \delta & V_2(x) \end{pmatrix}, \tag{60}$$

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. (60), with $\delta = 0$ and compare your results with those obtained in item (a).
(c)**Testing Marcus theory:** Rerun your code for a system of two coupled displaced harmonic oscillators described by the 2×2 potential energy matrix given in item (a), with $m = 100$, $\omega = 0.01$ and coupling $\delta = H_{if} = 0.1\omega$, with $V_1(x) = m\omega^2x^2/2$, and $V_2(x) = m\omega^2(x-\bar{x}_2)^2/2 - \Delta E$ with $\bar{x}_2 = \sqrt{2\lambda/(m\omega^2)}$, where $\lambda = 36\omega$. Initiate the wavepacket in the ground vibrational state $|\nu_0\rangle$ of V_1 and compute the population of the product state $P_2(\nu_0) = \langle\varphi_2|\varphi_2\rangle$ at time corresponding to half a period, $t_{1/2} = \pi/\omega$, to estimate the rate of electron transfer as $\kappa_0(\Delta E) = P_2(\nu_0)/t_{1/2}$. Repeat the calculation for initial states $|\nu_j\rangle$, with j =0–30, and estimate the thermal rate at $\beta = 20$ as the Boltzmann average $\kappa(\Delta E) = Z^{-1}\sum_{j=0}^{\infty}\kappa_j(\Delta E)exp(-\beta\hbar\omega(\frac{1}{2}+\nu_j))$, with the partition function $Z = \sum_{j=0}^{\infty} exp(-\beta\hbar\omega(\frac{1}{2}+\nu_j))$. Compare $\kappa(\Delta E)$ as a function of ΔE =0–5λ to the corresponding rates given by Marcus theory $\kappa(\Delta E) = |H_{if}|^2\sqrt{\pi\beta/\lambda}exp(-\beta E_a(\Delta E))$ where $E_a(\Delta E) = (\lambda - \Delta E)^2/(4.0\lambda)$.
Solution: The link (here) provides a Matlab solution of item (c).

11 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:

$$C(t) = \langle A(0)B(t)\rangle = \frac{1}{\tau}\int_0^\tau dt' A(q(t'),p(t'))B(q(t'+t),p(t'+t)), \tag{61}$$

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,

$$C(t) = Tr[\hat{\rho}\hat{A}\hat{B}(t)], \tag{62}$$

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

$$B(t) = \langle\Psi_0|\hat{B}(t)|\Psi_0\rangle = \langle\Psi_0|e^{(i/\hbar)\hat{H}t}\hat{B}e^{-(i/\hbar)\hat{H}t}|\Psi_0\rangle, \tag{63}$$

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,

$$\hat{H} = \hat{p}^2/(2m) + \hat{V}, \tag{64}$$

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,

$$C(t) = Z^{-1}Tr[e^{-\beta\hat{H}}\hat{A}e^{(i/\hbar)\hat{H}t}\hat{B}e^{-(i/\hbar)\hat{H}t}], \tag{65}$$

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

$$\begin{aligned} C(t) = Z^{-1}\int dx\int dx'\int dx''\int dx'''\int dx''''\langle x|e^{-\beta\hat{H}}|x'\rangle\langle x'|\hat{A}|x''\rangle\langle x''|e^{(i/\hbar)\hat{H}t}|x'''\rangle \\ \langle x'''|\hat{B}|x''''\rangle\langle x''''|e^{-(i/\hbar)\hat{H}t}|x\rangle. \end{aligned} \tag{66}$$

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,

$$\langle x_0|e^{-\beta\hat{H}}|x_n\rangle = \langle x_0|e^{-\epsilon\hat{H}}e^{-\epsilon\hat{H}}...e^{-\epsilon\hat{H}}|x_n\rangle, \tag{67}$$

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

$$\langle x_0|e^{-\beta\hat{H}}|x_n\rangle = \int dx_1...\int dx_{n-1}\langle x_0|e^{-\epsilon\hat{H}}|x_1\rangle...\langle x_{n-1}|e^{-\epsilon\hat{H}}|x_n\rangle = \prod_{j=1}^{n}\int dx_j\langle x_{j-1}|e^{-\epsilon\hat{H}}|x_j\rangle. \tag{68}$$

[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

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

$$e^{-\epsilon\hat{H}} \approx e^{-\epsilon\hat{V}/2}e^{-\epsilon\hat{p}^2/(2m)}e^{-\epsilon\hat{V}/2}, \tag{69}$$

to second order accuracy.

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

$$\langle x_0|e^{-\epsilon\hat{H}}|x_1\rangle = \int dx \int dp \int dx' \int dp' \langle x_0|e^{-\epsilon\hat{V}/2}|x'\rangle\langle x'|p'\rangle\langle p'|e^{-\epsilon\hat{p}^2/(2m)}|p\rangle\langle p|x\rangle\langle x|e^{-\epsilon\hat{V}/2}|x_1\rangle, \tag{70}$$

where

$$\langle x|p\rangle = \frac{1}{\sqrt{2\pi\hbar}}e^{\frac{i}{\hbar}xp}, \tag{71}$$

since

$$-i\hbar\frac{\partial}{\partial x}\langle x|p\rangle = p\langle x|p\rangle. \tag{72}$$

Furthermore,

$$\langle x|e^{-\epsilon\hat{V}/2}|x'\rangle = e^{-\epsilon V(x)/2}\delta(x-x'). \tag{73}$$

Therefore,

$$\begin{array}{l}\langle x_0|e^{-\epsilon\hat{H}}|x_1\rangle = \frac{1}{2\pi\hbar}\int dx \int dp \int dx' \int dp' e^{-\epsilon V(x')/2}\delta(x'-x_0)e^{\frac{i}{\hbar}x'p'}e^{-\epsilon p^2/(2m)}\delta(p-p') \\ e^{-\frac{i}{\hbar}xp}\delta(x-x_1)e^{-\epsilon V(x_1)/2},\end{array} \tag{74}$$

which gives,

$$\langle x_0|e^{-\epsilon\hat{H}}|x_1\rangle = \frac{1}{2\pi\hbar}e^{-\frac{\epsilon}{2}[V(x_0)+V(x_1)]}\int dp e^{-\epsilon p^2/(2m)+\frac{i}{\hbar}(x_0-x_1)p}, \tag{75}$$

or,

$$\langle x_0|e^{-\epsilon\hat{H}}|x_1\rangle = \frac{1}{2\pi\hbar}e^{-\frac{\epsilon}{2}(V(x_0)+V(x_1))}\sqrt{\frac{m}{2\pi\epsilon\hbar^2}}e^{-\frac{1}{2}m\left[\frac{(x_1-x_0)}{\hbar\epsilon}\right]^2\epsilon}. \tag{76}$$

Matrix elements of the Boltzmann operator at finite-temperature can be obtained by substituting Eq. (76) into Eq. (68):

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

where $\omega = 1/(\hbar\sqrt{\epsilon})$. Note that the r.h.s of Eq. (77) 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 multidimentional integral, introduced by Eq. (77), 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*.

11.1 Exercise 13

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

$$\hat{H} = \frac{\hat{p}^2}{2m} + \frac{1}{2}m\omega_0^2\hat{x}^2, \tag{78}$$

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:

$$\langle x|e^{-\beta\hat{H}}|x'\rangle = \sqrt{\frac{m\omega}{2\pi\hbar sinh(\beta\hbar\omega)}} exp\left(-\frac{m\omega}{2\hbar sinh(\omega\beta\hbar)}\left[(x^2+x'^2)cosh(\omega\beta) - 2xx'\right]\right), \tag{79}$$

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

$$\langle x|e^{-\beta\hat{H}}|x'\rangle = \sqrt{\frac{m}{2\pi\beta\hbar^2}} exp\left(-\frac{m}{2\beta\hbar^2}\left[(x-x')^2\right]\right), \tag{80}$$

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. (76) and we obtain the short-time propagator as follows:

$$\langle x|e^{-\frac{i}{\hbar}\hat{H}\tau}|x'\rangle = \sqrt{\frac{m}{2\pi\hbar i\tau}} e^{\frac{i}{\hbar}\left(\frac{1}{2}m\left[\frac{(x-x')}{\tau}\right]^2 - \frac{1}{2}[V(x)+V(x')]\right)\tau}. \tag{81}$$

Then, we concatenate the short-time propagators introduced by Eq. (81) and we obtain the finite-time propagator,

$$\langle x_0|e^{-\frac{i}{\hbar}\hat{H}t}|x_n\rangle = \int dx_1 ... \int dx_{n-1}\left(\frac{m}{2\pi\hbar i\tau}\right)^{n/2} e^{\frac{i}{\hbar}\left(\sum_{j=1}^{n}\frac{1}{2}m\left[\frac{(x_j-x_{j-1})}{\tau}\right]^2 - \frac{1}{2}[V(x_j)+V(x_{j-1})]\right)\tau}, \tag{82}$$

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

$$\langle x_0|e^{-\frac{i}{\hbar}\hat{H}t}|x_n\rangle = \int \mathfrak{D}[x(t)] e^{\frac{i}{\hbar}S_c(t)}, \tag{83}$$

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

$$S_c(t') \equiv \int_0^{t'} dt \quad \left[\frac{1}{2}m\left(\frac{\partial}{\partial t}x(t)\right)^2 - V(x(t))\right], \tag{84}$$

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

$$\int \mathfrak{D}[x(t)] f(x(t)) \equiv \int dx_1 ... \int dx_{n-1}\left(\frac{m}{2\pi\hbar i\tau}\right)^{n/2} f(x(t)), \tag{85}$$

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.

11.2 Ring Polymer Implementation

An alternative expression for the matrix elements of the Boltzmann operator can be obtained by introducing into Eq. (77) the following substitution:

$$\left(\frac{m}{2\pi\epsilon\hbar^2}\right)^{1/2} = (2\pi\hbar)^{-1}\int dp_j e^{-\epsilon\frac{p_j^2}{2m}}, \tag{86}$$

as follows:

$$\begin{aligned}\langle x_0|e^{-\beta\hat{H}}|x_n\rangle = (2\pi\hbar)^{-n}&\int dp_1 ... \int dp_n e^{-\epsilon\sum_{j=1}^{n}\frac{p_j^2}{2m}} \\ &\times \int dx_1 ... \int dx_{n-1} e^{-\epsilon\sum_{j=1}^{n} V(x_j)+\frac{1}{2}m\omega^2(x_j-x_{j-1})^2},\end{aligned} \tag{87}$$

to obtain

$$\langle x_0|e^{-\beta\hat{H}}|x_n\rangle = (2\pi\hbar)^{-n}\int dx_1 ... \int dx_{n-1}\int dp_1 ... \int dp_n\, e^{-\epsilon H_n(x_0;\mathbf{x}^n,\mathbf{p}^n)}, \tag{88}$$

where H_n is the n-bead polymer Hamiltonian

$$H_n(x_0;\mathbf{x}^n,\mathbf{p}^n) = \sum_{j=1}^{n}\frac{p_j^2}{2m} + V(x_j) + \frac{1}{2}m\omega^2(x_j - x_{j-1})^2, \tag{89}$$

and

$$\begin{aligned}Z &= \int dx_0 \int dx_n \delta(x_0 - x_n)\langle x_0|e^{-\beta\hat{H}}|x_n\rangle \\ &= (2\pi\hbar)^{-n}\int d\mathbf{x}^n \int d\mathbf{p}^n\, \delta(x_0 - x_n) e^{-\epsilon H_n(x_0;\mathbf{x}^n,\mathbf{p}^n)}.\end{aligned} \tag{90}$$

This section was prepared by Kenneth Jung.

The discrete path integral representation of the partition function is given (in units of $\hbar = 1$) by

$$\begin{aligned}Z &= \left(\frac{mN}{2\pi\beta}\right)^{N/2}\int dx_1 \int dx_2 \cdots \int dx_N \exp\left\{-\beta_N\sum_{j=1}^{N}\frac{1}{2}m\omega_N^2(x_{j+1}-x_j)^2 + V(x_j)\right\} \\ &= \left(\frac{mN}{2\pi\beta}\right)^{N/2}\int dx_1 \int dx_2 \cdots \int dx_N e^{-\beta U(x_1,x_2,\cdots x_N)}\end{aligned} \tag{91}$$

where N is the number of imaginary time slices (also referred to as beads) used to construct the path, $\beta_N = \beta/N$, $\omega_N = 1/\beta_N$ and $V(x)$ is the external potential. The second line was introduced to make the expression resemble a classical configuration integral. It is important to note that the path integral representation in Eq. (91) is cyclic and exact in the limit as $N \to \infty$.

Thermal expectation values can be similarly written in a path integral form by path integral discretizing the expression

$$\langle A\rangle = Tr\left[e^{-\beta\hat{H}}\hat{A}\right]/Tr\left[e^{-\beta\hat{H}}\right] = \frac{1}{Z}Tr\left[e^{-\beta\hat{H}}\hat{A}\right], \tag{92}$$

to get

$$\langle A\rangle = \frac{1}{Z}\left(\frac{mN}{2\pi\beta}\right)^{N/2}\int dx_1\int dx_2\cdots\int dx_N e^{-\beta U(x_1,x_2,\cdots x_N)}A_N(x_1,x_2,\cdots x_N), \tag{93}$$

where

$$A_N(x_1,x_2,\cdots x_N) = \sum_{j=1}^{N}A(x_j), \tag{94}$$

is the bead averaged observable. Eq. (93) is also exact in the limit as $N\to\infty$ and relies on the assumption that the operator $\hat{A}$ is a function of the position operator. A similar expression can also be found for cases where $\hat{A}$ is solely a function of the momentum operator.

Path integral Monte Carlo (PIMC) gives a way to evaluate Eq. (93) using standard sampling techniques to generate paths (which are commonly referred to as polymer configurations). A matlab implementation of a basic PIMC algorithm for computing the average energy of a 1-D harmonic oscillator can be found here . Before running the code:

(1) Derive the expression for the average energy of a harmonic oscillator using the standard definition of the partition function.

(2) Take the classical limit ($\beta\to 0$) and see that it agrees with the equipartition theorem.

Now run the code with the default parameters and compare the PIMC result of the average energy to the analytic expressions obtained. Then change the temperature β from 2.0 to 8.0 and rerun the code. Why doesn't the result agree at this lower temperature? Try to interpret this in terms of paths and configurations. How can this be fixed?

12 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. (65):

$$C(t) = Z^{-1}\int d\mathbf{x}\int d\mathbf{x}'\int d\mathbf{x}''\langle\mathbf{x}|e^{-\frac{\beta}{2}\hat{H}_0}|\mathbf{x}'\rangle A(\mathbf{x}')\langle\mathbf{x}'|e^{i\hat{H}_1 t}\hat{B}e^{-i\hat{H}_1 t}|\mathbf{x}''\rangle\langle\mathbf{x}''|e^{-\frac{\beta}{2}\hat{H}_0}|\mathbf{x}\rangle. \tag{95}$$

The computational task necessary to obtain $C(t)$, according to Eq. (95), 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,

$$\{\frac{\partial}{\partial\beta} - \frac{1}{2m}\nabla^2_{\mathbf{x}} + V_0(\mathbf{x})\}\rho(\mathbf{x},\mathbf{x}';\beta) = 0, \tag{96}$$

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,

$$\rho(\mathbf{x},\mathbf{x}';\epsilon) = \left(\frac{m}{2\pi\epsilon}\right)^{1/2} e^{-\frac{\epsilon}{2}[V_0(\mathbf{x})+V_0(\mathbf{x}')]} e^{-\frac{m}{2\epsilon}(\mathbf{x}-\mathbf{x}')^2}, \tag{97}$$

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

Equation (96) is formally integrated as follows,

$$\rho(\mathbf{x},\mathbf{x}';\beta) = \int d\mathbf{x}''\rho(\mathbf{x},\mathbf{x}'';\beta-\epsilon)\rho(\mathbf{x}'',\mathbf{x}';\epsilon), \tag{98}$$

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,

$$\hat{\mathbf{1}} = \int d\mathbf{x}_j|\mathbf{x}_j\rangle\langle\mathbf{x}_j|, \tag{99}$$

yielding,

$$\langle\mathbf{x}|e^{-(\beta-\epsilon)\hat{H}_0}|\mathbf{x}''\rangle = \int d\mathbf{x}_{s-1}... \int d\mathbf{x}_1\langle\mathbf{x}|e^{-i\hat{H}_0\tau}|\mathbf{x}_{s-1}\rangle...\langle\mathbf{x}_1|e^{-i\hat{H}_0\tau}|\mathbf{x}''\rangle, \tag{100}$$

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

Each finite-time propagator, introduced by Eq. (100), is approximated for sufficiently small imaginary-time slices τ by the Trotter expansion to second-order accuracy,

$$e^{-i\,\hat{H}_0\tau} \approx e^{-i\,V_0(\hat{\mathbf{x}})\tau/2} e^{-i\,\frac{\hat{\mathbf{p}}^2}{2m}\tau}\, e^{-i\,V_0(\hat{\mathbf{x}})\tau/2}. \tag{101}$$

12.1 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. (79) 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 in full-rank representation 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 [and here] or low-rank decomposition methods as implemented in tensor train representations.

12.2 Relation of Kubo Transforms to standard TCF's

This section was contributed by Kenneth Jung

There exists a general relationship of Kubo correlation functions to standard correlation functions through Fourier space which can be found by working in the energy representation of the Kubo transform.

$$\begin{aligned} C_{AB}(t) &= \frac{1}{Z} tr\left[e^{-\beta\hat{H}}\hat{A}(0)\hat{B}(t)\right] \\ &= \frac{1}{Z} tr\left[e^{-\beta\hat{H}}\hat{A}(0)e^{i\hat{H}t/\hbar}\hat{B}(0)e^{-i\hat{H}t/\hbar}\right] \\ &= \frac{1}{Z}\sum_{n,m}\langle n|\, e^{-\beta\hat{H}}\hat{A}(0)\,|m\rangle\,\langle m|\, e^{i\hat{H}t/\hbar}\hat{B}(0)e^{-i\hat{H}t/\hbar}\,|n\rangle \\ &= \frac{1}{Z}\sum_{n,m} e^{-\beta E_n}A_{nm}e^{iE_m t/\hbar}B_{mn}e^{-iE_n t/\hbar} \\ &= \frac{1}{Z}\sum_{n,m} e^{-\beta E_n}e^{i(E_m-E_n)t/\hbar}A_{nm}B_{mn} \end{aligned} \tag{102}$$

where $A_{nm} = \langle n|\,\hat{A}\,|m\rangle$ and $B_{mn} = \langle m|\,\hat{B}\,|n\rangle$. This provides a straightforward way to evaluate the correlation function if the Hamiltonian can be efficiently diagonalized. Next we will Fourier transform the correlation function to obtain

$$
\begin{aligned}
\tilde{C}_{AB}(\omega) &= \int dt e^{-i\omega t} C_{AB}(t) \\
&= \frac{1}{Z}\int dt e^{-i\omega t}\sum_{n,m} e^{-\beta E_n} e^{i(E_m-E_n)t/\hbar} A_{nm}B_{mn} \\
&= \frac{1}{Z}\sum_{n,m} e^{-\beta E_n} A_{nm}B_{mn}\int dt e^{-i\omega t} e^{i(E_m-E_n)t/\hbar} \\
&= \frac{1}{Z}\sum_{n,m} e^{-\beta E_n} A_{nm}B_{mn}\delta\left[\omega-(E_m-E_n)\right]
\end{aligned}
\tag{103}
$$

Next we will evaluate the Kubo form in the eigen basis

$$
\begin{aligned}
K_{AB}(t) &= \frac{1}{Z\beta}\int_0^\beta d\lambda tr\left[e^{-(\beta-\lambda)\hat{H}}\hat{A}(0)e^{-\lambda\hat{H}}\hat{B}(t)\right] \\
&= \frac{1}{Z\beta}\int_0^\beta d\lambda tr\left[e^{-(\beta-\lambda)\hat{H}}\hat{A}(0)e^{-\lambda\hat{H}}e^{i\hat{H}t/\hbar}\hat{B}(0)e^{-i\hat{H}t/\hbar}\right] \\
&= \frac{1}{Z\beta}\int_0^\beta d\lambda \sum_{n.m}\langle n|\, e^{-(\beta-\lambda)\hat{H}}\hat{A}(0)\,|m\rangle\,\langle m|\, e^{-\lambda\hat{H}}e^{i\hat{H}t/\hbar}\hat{B}(0)e^{-i\hat{H}t/\hbar}\,|n\rangle \\
&= \frac{1}{Z\beta}\int_0^\beta d\lambda \sum_{n.m} e^{-(\beta-\lambda)E_n}A_{nm}e^{-\lambda E_m}e^{iE_mt/\hbar}B_{mn}e^{-iE_nt/\hbar} \\
&= \frac{1}{Z\beta}\sum_{n.m}\int_0^\beta d\lambda e^{\lambda(E_n-E_m)}e^{-\beta E_n}e^{i(E_m-E_n)t/\hbar}A_{nm}B_{mn} \\
&= \frac{1}{Z\beta}\sum_{n.m}\frac{e^{\beta(E_n-E_m)}-1}{(E_n-E_m)}e^{-\beta E_n}e^{i(E_m-E_n)t/\hbar}A_{nm}B_{mn} \\
&= \frac{1}{Z\beta}\sum_{n.m}\frac{1-e^{-\beta(E_m-E_n)}}{(E_m-E_n)}e^{-\beta E_n}e^{i(E_m-E_n)t/\hbar}A_{nm}B_{mn}
\end{aligned}
\tag{104}
$$

and again we Fourier transform to get

$$
\begin{aligned}
\tilde{K}_{AB}(\omega) &= \int dt e^{-i\omega t} K_{AB}(t) \\
&= \frac{1}{Z\beta}\sum_{n.m}\frac{1-e^{-\beta(E_m-E_n)}}{(E_m-E_n)}e^{-\beta E_n}A_{nm}B_{mn}\int dt e^{-i\omega t}e^{i(E_m-E_n)t/\hbar} \\
&= \frac{1}{Z\beta}\sum_{n.m}\frac{1-e^{-\beta(E_m-E_n)}}{(E_m-E_n)}e^{-\beta E_n}A_{nm}B_{mn}\delta\left[\omega-(E_m-E_n)\right]
\end{aligned}
\tag{105}
$$

If we compare this result with the Fourier transform of the normal TCF and rename $(E_m - E_n)$ as $\hbar\omega$ we see that the Kubo TCF and the normal TCF are related through their Fourier transforms as

$$\tilde{C}_{AB}(\omega) = \frac{\beta\hbar\omega}{1 - e^{-\beta\hbar\omega}} \tilde{K}_{AB}(\omega) \tag{106}$$

12.3 Rates from two-point correlation functions

The goal of this section is to introduce the flux-flux, flux-side and side-side correlation functions C_f, $C_{f,s}$ and C_s, respectively, and to show how to use them to compute the rate of a chemical reaction $k(T)$ correctly and directly (*i.e.*, without approximations and without computing intermediate quantities), as shown by Bill Miller, as follows:

$$\begin{aligned} k(T) &= Z^{-1} \int_0^\infty dt C_f(t), \\ &= Z^{-1} \lim_{t\to\infty} C_{f,s}(t), \end{aligned} \tag{107}$$

where $\beta = (k_B T)^{-1}$ is the inverse temperature T and $Z = Tr[exp(-\beta\hat{H})]$ is the canonical partition function when the system is described by the Hamiltonian,

$$\hat{H} = \hat{p}^2/(2m) + V(x). \tag{108}$$

The flux-flux correlation function C_f is defined as the time-derivative of the flux-side correlation function $C_{f,s}$, as follows:

$$\begin{aligned} C_f(t) &= Tr[\hat{F} e^{\frac{i}{\hbar}\hat{H}t_c^*} \hat{F} e^{-\frac{i}{\hbar}\hat{H}t_c}], \\ &= Tr[e^{-\frac{\beta}{2}\hat{H}} \hat{F} e^{-\frac{\beta}{2}\hat{H}} e^{\frac{i}{\hbar}\hat{H}t} \hat{F} e^{-\frac{i}{\hbar}\hat{H}t}], \\ &= \frac{d}{dt} Tr[e^{-\frac{\beta}{2}\hat{H}} \hat{F} e^{-\frac{\beta}{2}\hat{H}} e^{\frac{i}{\hbar}\hat{H}t} h(x-s) e^{-\frac{i}{\hbar}\hat{H}t}] = \frac{d}{dt} C_{f,s}(t), \end{aligned} \tag{109}$$

with $t_c = t - i\hbar\beta/2$, while $C_{f,s}$ is the time derivative of the side-side correlation function C_s, as shown later in this section. Note that the second line of Eq. (107) simply states that $C_{f,s}(0) = 0$ when the system is initially a reactant since $\lim_{t\to 0} Tr[e^{-\frac{\beta}{2}\hat{H}} \hat{F} e^{-\frac{\beta}{2}\hat{H}} e^{\frac{i}{\hbar}\hat{H}t} h(x-s) e^{-\frac{i}{\hbar}\hat{H}t}] = 0$.

Here, $\hat{F}$ is the flux operator for reactants transforming into products as they cross a dividing surface placed at $x = s$ in configurational space (*i.e.*, $\hat{F}$ gives the number of molecules crossing per unit time the surface separating reactants and products).

As shown in the last line of Eq. (109), the Heisenberg flux operator $\hat{F}^H = e^{\frac{i}{\hbar}\hat{H}t} \hat{F} e^{-\frac{i}{\hbar}\hat{H}t}$ is the time derivative of the Heisenberg projection operator $\hat{h}^H(x-s) = e^{\frac{i}{\hbar}\hat{H}t} h(x-s) e^{-\frac{i}{\hbar}\hat{H}t}$, with $h(x-s) = 1$ when $x > s$ and $h(x-s) = 0$, otherwise. Note that $Tr[\hat{\rho}\hat{h}^H]$ gives the product

population at time t, therefore its time derivative defines the flux, as follows:

$$\begin{aligned} e^{\frac{i}{\hbar}\hat{H}t}\hat{F}e^{-\frac{i}{\hbar}\hat{H}t} &= \frac{d}{dt}e^{\frac{i}{\hbar}\hat{H}t}h(x-s)e^{-\frac{i}{\hbar}\hat{H}t}, \\ &= \frac{i}{\hbar}\left(\hat{H}e^{\frac{i}{\hbar}\hat{H}t}h(x-s)e^{-\frac{i}{\hbar}\hat{H}t} - e^{\frac{i}{\hbar}\hat{H}t}h(x-s)e^{-\frac{i}{\hbar}\hat{H}t}\hat{H}\right), \\ &= \frac{i}{\hbar}[\hat{H},\hat{h}^H(x-s)], \\ &= e^{\frac{i}{\hbar}\hat{H}t}\frac{i}{\hbar}[\hat{H},h(x-s)]e^{-\frac{i}{\hbar}\hat{H}t}. \\ \hat{F} &= \frac{i}{\hbar}[\hat{H},h(x-s)]. \end{aligned} \tag{110}$$

Substituting the Hamiltonian, introduced by Eq. (108), into Eq. (111), we obtain:

$$\begin{aligned} \hat{F} &= \frac{i}{\hbar 2m}[\hat{p}^2,h(x-s)], \\ &= \frac{i}{\hbar 2m}\left(\hat{p}[\hat{p},h(x-s)] + [\hat{p},h(x-s)]\hat{p}\right), \\ &= \frac{i}{\hbar 2}\left(\frac{\hat{p}}{m}[\hat{p},h(x-s)] + [\hat{p},h(x-s)]\frac{\hat{p}}{m}\right), \\ &= \frac{1}{2}\left(\frac{\hat{p}}{m}\delta(x-s) + \delta(x-s)\frac{\hat{p}}{m}\right), \end{aligned} \tag{111}$$

where we have substituted $[\hat{p},h(x-s)] = -i\hbar\left(\frac{dh(x-s)}{dx} + h(x-s)\frac{d}{dx} - h(x-s)\frac{d}{dx}\right)$ with $\frac{dh(x-s)}{dx} = \delta(x-s)$.

Note that the expectation value of the flux, is the current $j(x)$ at $x=s$:

$$\begin{aligned} \langle\psi|\hat{F}|\psi\rangle &= \frac{1}{2m}\left(\langle\psi|\hat{p}|\delta(x-s)\psi\rangle + \langle\psi\delta(x-s)|\hat{p}|\psi\rangle\right), \\ &= -\frac{i\hbar}{2m}\left(-\langle\frac{\partial\psi}{\partial x}|\delta(x-s)\psi\rangle + \langle\psi\delta(x-s)|\frac{\partial\psi}{\partial x}\rangle\right), \\ &= -\frac{i\hbar}{2m}\left(-\psi'(s)^*\psi(s) + \psi(s)^*\psi'(s)\right), \\ &= j(s). \end{aligned} \tag{112}$$

To show that Eq. (107) makes sense, we first note that according to Eq. (109),

$$C_{f,s}(t) = Tr[e^{-\frac{\beta}{2}\hat{H}}\hat{F}e^{-\frac{\beta}{2}\hat{H}}e^{\frac{i}{\hbar}\hat{H}t}h(x-s)e^{-\frac{i}{\hbar}\hat{H}t}], \tag{113}$$

and according to Eq. (107),

$$k(T) = Z^{-1}\lim_{t\to\infty} Tr[e^{-\frac{\beta}{2}\hat{H}}\hat{F}e^{-\frac{\beta}{2}\hat{H}}e^{\frac{i}{\hbar}\hat{H}t}h(x-s)e^{-\frac{i}{\hbar}\hat{H}t}]. \tag{114}$$

Furthermore, we show below that

$$\lim_{t\to\infty} e^{\frac{i}{\hbar}\hat{H}t} h(x-s) e^{-\frac{i}{\hbar}\hat{H}t} = \lim_{t\to\infty} e^{\frac{i}{\hbar}\hat{H}t} h(\hat{p}) e^{-\frac{i}{\hbar}\hat{H}t}, \tag{115}$$

so

$$k(T) = Z^{-1} \lim_{t\to\infty} Tr[e^{-\frac{\beta}{2}\hat{H}} \hat{F} e^{-\frac{\beta}{2}\hat{H}} e^{\frac{i}{\hbar}\hat{H}t} h(\hat{p}) e^{-\frac{i}{\hbar}\hat{H}t}]. \tag{116}$$

Now, using the cyclic permutation we get

$$k(T) = Z^{-1} \lim_{t\to\infty} Tr[\hat{F} e^{-\frac{\beta}{2}\hat{H}} e^{\frac{i}{\hbar}\hat{H}t} h(\hat{p}) e^{-\frac{i}{\hbar}\hat{H}t} e^{-\frac{\beta}{2}\hat{H}}], \tag{117}$$

and using that $e^{-\beta\hat{H}/2} e^{i\hat{H}t/\hbar} = e^{i\hat{H}t/\hbar} e^{-\beta\hat{H}/2}$, we get

$$k(T) = Z^{-1} \lim_{t\to\infty} Tr[\hat{F} e^{\frac{i}{\hbar}\hat{H}t} e^{-\frac{\beta}{2}\hat{H}} h(p) e^{-\frac{\beta}{2}\hat{H}} e^{-\frac{i}{\hbar}\hat{H}t}]. \tag{118}$$

Introducing $\hat{G}_\beta = e^{-\frac{\beta}{2}\hat{H}} h(\hat{p}) e^{-\frac{\beta}{2}\hat{H}}$, we obtain

$$k(T) = Z^{-1} \lim_{t\to\infty} Tr[\hat{F} e^{\frac{i}{\hbar}\hat{H}t} \hat{G}_\beta e^{-\frac{i}{\hbar}\hat{H}t}], \tag{119}$$

which according to Eq. (111), gives

$$k(T) = \frac{Z^{-1}}{2m} \lim_{t\to\infty} Tr[\hat{p}\delta(x-s) e^{\frac{i}{\hbar}\hat{H}t} \hat{G}_\beta e^{-\frac{i}{\hbar}\hat{H}t}] + Tr[\delta(x-s)\hat{p} e^{\frac{i}{\hbar}\hat{H}t} \hat{G}_\beta e^{-\frac{i}{\hbar}\hat{H}t}]. \tag{120}$$

Using the fact that $\hat{G}_\beta = e^{-\frac{\beta}{2}\hat{H}} h(\hat{p}) e^{-\frac{\beta}{2}\hat{H}}$ is Hermitian (*i.e.*, $\hat{G}_\beta = \hat{G}_\beta^\dagger$ with $\dagger$ representing the adjoint, or complex conjugate), we obtain:

$$\begin{aligned}
k(T) &= \frac{Z^{-1}}{2m} \lim_{t\to\infty} Tr[e^{\frac{i}{\hbar}\hat{H}t} \hat{G}_\beta e^{-\frac{i}{\hbar}\hat{H}t} \delta(x-s)\hat{p}]^* + Tr[\delta(x-s)\hat{p} e^{\frac{i}{\hbar}\hat{H}t} \hat{G}_\beta e^{-\frac{i}{\hbar}\hat{H}t}], \\
&= \frac{Z^{-1}}{2m} \lim_{t\to\infty} Tr[\delta(x-s)\hat{p} e^{\frac{i}{\hbar}\hat{H}t} \hat{G}_\beta e^{-\frac{i}{\hbar}\hat{H}t}]^* + Tr[\delta(x-s)\hat{p} e^{\frac{i}{\hbar}\hat{H}t} \hat{G}_\beta e^{-\frac{i}{\hbar}\hat{H}t}], \\
&= Z^{-1} \lim_{t\to\infty} \text{Re}\{Tr[\delta(x-s)\frac{\hat{p}}{m} e^{\frac{i}{\hbar}\hat{H}t} \hat{G}_\beta e^{-\frac{i}{\hbar}\hat{H}t}]\}, \\
&= Z^{-1} \lim_{t\to\infty} \text{Re}\{Tr[\delta(x-s)\frac{\hat{p}}{m} e^{-\frac{\beta}{2}\hat{H}} \mathcal{P} e^{-\frac{\beta}{2}\hat{H}}]\}, \\
&= Z^{-1} \text{Re}\{Tr[e^{-\beta\hat{H}} \delta(x-s)\frac{\hat{p}}{m}\hat{\mathcal{P}}]\}, \\
&= \text{Re}\{Tr[\hat{\rho}\delta(x-s)\frac{\hat{p}}{m}\hat{\mathcal{P}}]\},
\end{aligned} \tag{121}$$

where Re denotes 'the real part of', and $\hat{\mathcal{P}} = \lim_{t\to\infty} e^{\frac{i}{\hbar}\hat{H}t} h(\hat{p}) e^{-\frac{i}{\hbar}\hat{H}t}$ which commutes with $e^{-\frac{\beta}{2}\hat{H}}$ (*i.e.*, $e^{\frac{i}{\hbar}\hat{H}t}\hat{p}e^{-\frac{i}{\hbar}\hat{H}t_c} - e^{\frac{i}{\hbar}\hat{H}t_c^*}\hat{p}e^{-\frac{i}{\hbar}\hat{H}t} = 0$, because $h(\hat{p})$ is Hermitian). Note that the projection operator $\hat{\mathcal{P}}$ keeps only the molecules moving forward (*i.e.*, towards the product side) at the asymptotic

long time, while $\hat{\rho}\delta(x-s)\hat{p}/m$ gives the number of molecules crossing the dividing surface per unit time.

To prove Eq. (115), we introduce the Møller operator,

$$\hat{\Omega} = e^{\frac{i}{\hbar}\hat{H}_0 t} e^{-\frac{i}{\hbar}\hat{H} t}, \tag{122}$$

where $\hat{H}_0 = \hat{p}^2/(2m)$. Substituting into Eq. (115), we obtain:

$$\lim_{t\to\infty} \hat{\Omega}^\dagger e^{\frac{i}{\hbar}\hat{H}_0 t} h(x-s) e^{-\frac{i}{\hbar}\hat{H}_0 t}\hat{\Omega} = \lim_{t\to\infty} \hat{\Omega}^\dagger e^{\frac{i}{\hbar}\hat{H}_0 t} h(\hat{p}) e^{-\frac{i}{\hbar}\hat{H}_0 t}\hat{\Omega}, \tag{123}$$

and finally we show that the matrix elements of both operators are the same,

$$\langle x| e^{\frac{i}{\hbar}\hat{H}_0 t} h(x-s) e^{-\frac{i}{\hbar}\hat{H}_0 t}|x'\rangle = \langle x| e^{\frac{i}{\hbar}\hat{H}_0 t} h(\hat{p}) e^{-\frac{i}{\hbar}\hat{H}_0 t}|x'\rangle, \tag{124}$$

in the limit when $t \to \infty$.

Substituting $\hat{H}_0$ into the r.h.s. of Eq. (124), we obtain:

$$\begin{aligned}
\langle x| e^{\frac{i}{\hbar}\hat{H}_0 t} h(\hat{p}) e^{-\frac{i}{\hbar}\hat{H}_0 t}|x'\rangle &= \langle x| e^{\frac{i}{\hbar}\frac{\hat{p}^2}{2m} t} h(\hat{p}) e^{-\frac{i}{\hbar}\frac{\hat{p}^2}{2m} t}|x'\rangle, \\
&= \langle x| h(\hat{p})|x'\rangle, \\
&= \int_0^\infty dp \langle x|p\rangle\langle p|x'\rangle, \\
&= \frac{1}{2\pi\hbar}\int_0^\infty dp e^{\frac{i}{\hbar}p(x-x')}.
\end{aligned} \tag{125}$$

Inserting closure and $\hat{H}_0$ into the l.h.s. of Eq. (124), we obtain:

$$\begin{aligned}
\langle x| e^{\frac{i}{\hbar}\hat{H}_0 t} h(x-s) e^{-\frac{i}{\hbar}\hat{H}_0 t}|x'\rangle &= \int_s^\infty dx'' \int_{-\infty}^\infty dp e^{\frac{i}{\hbar}\frac{p^2}{2m}t} \langle x|p\rangle\langle p|x''\rangle \int_{-\infty}^\infty dp' \langle x''|p'\rangle\langle p'|x'\rangle e^{-\frac{i}{\hbar}\frac{p'^2}{2m}t}, \\
&= \int_s^\infty dx'' \frac{1}{2\pi\hbar}\int_{-\infty}^\infty dp e^{\frac{i}{\hbar}\frac{p^2}{2m}t} e^{\frac{i}{\hbar}p(x-x'')} \frac{1}{2\pi\hbar}\int_{-\infty}^\infty dp' e^{\frac{i}{\hbar}p'(x''-x')} e^{-\frac{i}{\hbar}\frac{p'^2}{2m}t}, \\
&= \int_s^\infty dx'' \frac{1}{2\pi\hbar}\sqrt{\frac{\pi 2m\hbar}{-it}} e^{\frac{2m\hbar}{\hbar^2 4it}(x-x'')^2} \frac{1}{2\pi\hbar}\sqrt{\frac{\pi 2m\hbar}{it}} e^{-\frac{2m\hbar}{\hbar^2 4it}(x''-x')^2}, \\
&= \int_s^\infty dx'' \frac{1}{2\pi\hbar}\frac{m}{t} e^{\frac{i}{\hbar}\frac{m}{2t}((x''-x')^2-(x-x'')^2)}, \\
&= \int_s^\infty dx'' \frac{1}{2\pi\hbar}\frac{m}{t} e^{\frac{i}{\hbar}\frac{m}{2t}(x'^2-x^2)} e^{\frac{i}{\hbar}\frac{m}{t}(x-x')x''}.
\end{aligned} \tag{126}$$

Introducing the change of variables $p = mx''/t$, with $dp = dx'' m/t$, we obtain:

$$\langle x| e^{\frac{i}{\hbar}\hat{H}_0 t} h(x-s) e^{-\frac{i}{\hbar}\hat{H}_0 t}|x'\rangle = e^{\frac{i}{\hbar}\frac{m}{2t}(x'^2-x^2)} \frac{1}{2\pi\hbar}\int_{\frac{ms}{t}}^\infty dp e^{\frac{i}{\hbar}p(x-x')}, \tag{127}$$

which in the limit when $t \to \infty$ is identical to Eq. (125).

The rate constant can also be obtained as the time-derivative of the side-side correlation function C_s, considering that $e^{-\frac{\beta}{2}\hat{H}}$ commutes with $e^{-\frac{i}{\hbar}\hat{H}t}$, and that according to Eq. (111), $e^{-\frac{i}{\hbar}\hat{H}t}\hat{F}e^{\frac{i}{\hbar}\hat{H}t} = \frac{i}{\hbar}e^{-\frac{i}{\hbar}\hat{H}t}[\hat{H}, h(x-s)]e^{\frac{i}{\hbar}\hat{H}t} = -\frac{d}{dt}e^{-\frac{i}{\hbar}\hat{H}t}h(x-s)e^{\frac{i}{\hbar}\hat{H}t} = \frac{d}{dt}e^{-\frac{i}{\hbar}\hat{H}t}(1-h(x-s))e^{\frac{i}{\hbar}\hat{H}t}$, so

$$\begin{aligned} k(T) &= Z^{-1}\lim_{t\to\infty}\frac{d}{dt}Tr[e^{-\frac{\beta}{2}\hat{H}}(1-h(x-s))e^{-\frac{\beta}{2}\hat{H}}e^{\frac{i}{\hbar}\hat{H}t}h(x-s)e^{-\frac{i}{\hbar}\hat{H}t}] = Z^{-1}\lim_{t\to\infty}\frac{d}{dt}C_s(t), \\ &= Z^{-1}\lim_{t\to\infty}\frac{d}{dt}\int_{-\infty}^{\infty}dx\int_{\infty}^{\infty}dx'\langle x|(1-h(x-s))e^{\frac{i}{\hbar}\hat{H}t_c^*}|x'\rangle\langle x'|h(x-s)e^{-\frac{i}{\hbar}\hat{H}t_c}|x\rangle, \\ &= Z^{-1}\lim_{t\to\infty}\frac{d}{dt}\int_{-\infty}^{\infty}dx\int_{s}^{\infty}dx'\langle x|(1-h(x-s))e^{\frac{i}{\hbar}\hat{H}t_c^*}|x'\rangle\langle x'|e^{-\frac{i}{\hbar}\hat{H}t_c}|x\rangle, \end{aligned} \tag{128}$$

with $\ddot{C}_s = \dot{C}_{f,s} = C_f$.

When the dividing surface is at the origin ($s = 0$), $1 - h(x-s) = h(s-x)$, so

$$\begin{aligned} k(T) &= Z^{-1}\lim_{t\to\infty}\frac{d}{dt}\int_{-\infty}^{0}dx\int_{0}^{\infty}dx'\langle x|e^{\frac{i}{\hbar}\hat{H}t_c^*}|x'\rangle\langle x'|e^{-\frac{i}{\hbar}\hat{H}t_c}|x\rangle, \\ &= Z^{-1}\lim_{t\to\infty}\frac{d}{dt}\int_{-\infty}^{0}dx\int_{0}^{\infty}dx'|\langle x'|e^{-\frac{i}{\hbar}\hat{H}t_c}|x\rangle|^2. \end{aligned} \tag{129}$$

12.4 Standard and Kubo three-point correlation function relationship

Analogously to the derivation in the previous section, we could derive the corresponding relationship for the three-point correlation function, as follows:

$$\begin{aligned} C_{ABC}(t,t') &= \frac{1}{Z}tr\left[e^{-\beta\hat{H}}\hat{A}(0)\hat{B}(t)\hat{C}(t')\right] \\ &= \frac{1}{Z}tr\left[e^{-\beta\hat{H}}\hat{A}(0)e^{i\hat{H}t/\hbar}\hat{B}(0)e^{-i\hat{H}(t-t')/\hbar}\hat{C}(0)e^{-i\hat{H}t'/\hbar}\right] \\ &= \frac{1}{Z}\sum_{n,m,l}\langle n|\,e^{-\beta\hat{H}}\hat{A}(0)\,|m\rangle\,\langle m|\,e^{i\hat{H}t/\hbar}\hat{B}(0)e^{-i\hat{H}t/\hbar}\,|l\rangle\,\langle l|e^{i\hat{H}t'/\hbar}\hat{C}(0)e^{-i\hat{H}t'/\hbar}|n\rangle \\ &= \frac{1}{Z}\sum_{n,m,l}e^{-\beta E_n}A_{nm}e^{iE_mt/\hbar}B_{ml}e^{-iE_l(t-t')/\hbar}C_{ln}e^{-iE_nt'/\hbar} \\ &= \frac{1}{Z}\sum_{n,m,l}e^{-\beta E_n}e^{i(E_m-E_l)t/\hbar}e^{i(E_l-E_n)t'/\hbar}A_{nm}B_{ml}C_{ln} \end{aligned} \tag{130}$$

which generates the the standard three point correlation function in frequency domain by Fourier transform, as follows:

$$\begin{aligned}
\tilde{C}_{ABC}(\omega,\omega') &= \frac{1}{(2\pi\hbar)^2}\int dt' e^{-i\omega' t'}\int dt e^{-i\omega t} C_{ABC}(t,t') \\
&= \frac{1}{Z}\frac{1}{(2\pi\hbar)^2}\int dt' e^{-i\omega' t'}\int dt e^{-i\omega t}\sum_{n,m,l} e^{-\beta E_n} e^{i(E_m-E_l)t/\hbar} e^{i(E_l-E_n)t'/\hbar} A_{nm}B_{ml}C_{ln} \\
&= \frac{1}{Z}\frac{1}{(2\pi\hbar)^2}\sum_{n,m,l} e^{-\beta E_n} A_{nm}B_{ml}C_{ln}\int dt' e^{i(E_l-E_n-\hbar\omega')t'/\hbar}\int dt e^{i(E_m-E_l-\hbar\omega)t/\hbar} \\
&= \frac{1}{Z}\sum_{n,m,l} e^{-\beta E_n} A_{nm}B_{ml}C_{ln}\delta\left(E_{ml}-\hbar\omega\right)\delta\left(E_{ln}-\hbar\omega'\right) \qquad (131)
\end{aligned}$$

Next we will evaluate the Kubo transformed in the basis of eigenstates, as follows:

$$\begin{aligned}
K_{AB}(t,t') &= \frac{1}{Z\beta^2}\int_0^\beta d\lambda\int_0^\lambda d\lambda' tr\left[e^{-(\beta-\lambda)\hat{H}}\hat{A}(0)e^{-(\lambda-\lambda')\hat{H}}\hat{B}(t)e^{-\lambda'\hat{H}}C(t')\right] \\
&= \frac{1}{Z\beta^2}\int_0^\beta d\lambda\int_0^\lambda d\lambda'\sum_{n.m,l}\langle n|\, e^{-(\beta-\lambda)\hat{H}}\hat{A}(0)\,|m\rangle \\
&\quad\times\langle m|\, e^{-(\lambda-\lambda')\hat{H}}e^{i\hat{H}t/\hbar}\hat{B}(0)e^{-i\hat{H}t/\hbar}e^{-\lambda'\hat{H}}e^{i\hat{H}t'/\hbar}|l\rangle\langle l|\hat{C}(0)e^{-i\hat{H}t'/\hbar}\,|n\rangle \\
&= \frac{1}{Z\beta^2}\int_0^\beta d\lambda\int_0^\lambda d\lambda'\sum_{n.m,l} e^{-(\beta-\lambda)E_n}A_{nm}e^{-(\lambda-\lambda')E_m}e^{iE_mt/\hbar}B_{ml} \\
&\quad\times e^{-iE_lt/\hbar}e^{-\lambda' E_l}e^{iE_lt'/\hbar}C_{ln}e^{-iE_nt'/\hbar} \\
&= \frac{1}{Z\beta^2}\sum_{n.m,l} e^{-\beta E_n}\int_0^\beta d\lambda\int_0^\lambda d\lambda' e^{\lambda E_n}A_{nm}e^{-(\lambda-\lambda')E_m}e^{iE_mt/\hbar}B_{ml} \\
&\quad\times e^{-iE_lt/\hbar}e^{-\lambda' E_l}e^{iE_lt'/\hbar}C_{ln}e^{-iE_nt'/\hbar} \\
&= \frac{1}{Z\beta^2}\sum_{n.m,l} A_{nm}B_{ml}C_{ln}e^{-\beta E_n}e^{-iE_{lm}t/\hbar}e^{-iE_{nl}t'/\hbar}\int_0^\beta d\lambda e^{\lambda E_{nm}}\int_0^\lambda d\lambda' e^{-\lambda' E_{lm}} \\
&= \frac{1}{Z\beta^2}\sum_{n.m,l} A_{nm}B_{ml}C_{ln}e^{-\beta E_n}e^{-iE_{lm}t/\hbar}e^{-iE_{nl}t'/\hbar}\int_0^\beta d\lambda e^{\lambda(E_{nm})}\frac{[e^{-\lambda E_{lm}}-1]}{E_{ml}} \\
&= \frac{1}{Z\beta^2}\sum_{n.m,l} A_{nm}B_{ml}C_{ln}e^{-\beta E_n}e^{-iE_{lm}t/\hbar}e^{-iE_{nl}t'/\hbar}\int_0^\beta d\lambda\frac{[e^{-\lambda E_{ln}}-e^{\lambda E_{nm}}]}{E_{ml}} \\
&= \frac{1}{Z\beta^2}\sum_{n.m,l}\frac{A_{nm}B_{ml}C_{ln}}{E_{ml}}e^{-\beta E_n}e^{-iE_{lm}t/\hbar}e^{-iE_{nl}t'/\hbar}\int_0^\beta d\lambda[e^{-\lambda E_{ln}}-e^{\lambda E_{nm}}] \\
&= \frac{1}{Z\beta^2}\sum_{n.m,l}\frac{A_{nm}B_{ml}C_{ln}}{E_{ml}}e^{-\beta E_n}e^{-iE_{lm}t/\hbar}e^{-iE_{nl}t'/\hbar}\left[\frac{e^{-\beta E_{ln}}-1}{E_{nl}}-\frac{e^{-\beta E_{nm}}-1}{E_{mn}}\right]
\end{aligned}$$

and computing the Fourier transformed with respect to t and t', we obtain:

$$\begin{aligned}
\tilde{K}_{AB}(\omega,\omega') &= \frac{1}{Z\beta^2}\sum_{n.m,l}\frac{A_{nm}B_{ml}C_{ln}}{E_{ml}}e^{-\beta E_n}\left[\frac{e^{-\beta E_{ln}}-1}{E_{nl}}-\frac{e^{-\beta E_{nm}}-1}{E_{mn}}\right] \\
&\quad\times\delta(E_{lm}-\hbar\omega)\delta(E_{nl}-\hbar\omega') \\
&= \frac{1}{Z\beta^2}\sum_{n.m,l}A_{nm}B_{ml}C_{ln}e^{-\beta E_n}\left[\frac{e^{-\beta\hbar\omega'}-1}{\hbar\omega\hbar\omega'}-\frac{e^{-\beta E_{nm}}-1}{(E_m-E_l+E_l-E_n)\hbar\omega}\right] \\
&\quad\times\delta(E_{lm}-\hbar\omega)\delta(E_{nl}-\hbar\omega') \\
&= \frac{1}{Z\beta^2}\sum_{n.m,l}A_{nm}B_{ml}C_{ln}e^{-\beta E_n}\left[\frac{e^{-\beta\hbar\omega'}-1}{\hbar\omega\hbar\omega'}+\frac{e^{-\beta E_{nm}}-1}{(E_{lm}+E_{nl})\hbar\omega}\right] \\
&\quad\times\delta(E_{lm}-\hbar\omega)\delta(E_{nl}-\hbar\omega') \\
&= \frac{1}{Z}\sum_{n.m,l}A_{nm}B_{ml}C_{ln}e^{-\beta E_n}\delta(E_{lm}-\hbar\omega)\delta(E_{nl}-\hbar\omega') \\
&\quad\times\frac{1}{\hbar^2\beta^2}\left[\frac{e^{-\beta\hbar\omega'}-1}{\omega\omega'}+\frac{e^{-\beta\hbar(\omega'+\omega)}-1}{(\omega+\omega')\omega}\right] \qquad (132)
\end{aligned}$$

Substituting Eq. (131) into Eq. (132), we obtain:

$$\tilde{K}_{ABC}(\omega,\omega') = \frac{1}{\hbar^2\beta^2}\left[\frac{e^{-\beta\hbar\omega'}-1}{\omega\omega'}+\frac{e^{-\beta\hbar(\omega'+\omega)}-1}{(\omega+\omega')\omega}\right]\tilde{C}_{ABC}(\omega,\omega'). \qquad (133)$$

12.5 Detailed Balance Relation of the Second Order Kubo Transform

This section was contributed by Kenneth Jung

The Kubo transform and RPMD both share the property of detailed balance. The goal of this section is to establish a detailed balance relationship of higher order Kubo transforms. We begin with

$$\begin{aligned}
K_{ABC}(t,t') &= \frac{1}{Z\beta^2}\int_0^\beta d\lambda\int_0^\lambda d\lambda' Tr[e^{-(\beta-\lambda)\hat{H}}\hat{A}e^{-(\lambda-\lambda')\hat{H}}\hat{B}(t)e^{-\lambda'\hat{H}}\widehat{C}(t')] \\
&= \langle A(0)B(t)C(t')\rangle_K \qquad (134)
\end{aligned}$$

with t' in general being just an additive constant of t

$$t' = t+\tau \qquad (135)$$

. giving $\langle A(0)B(t)C(t')\rangle = \langle A(0)B(t)C(t+\tau)\rangle$. Next we define

$$\langle A(-\tau)B(t-\tau)C(t)\rangle_K = \frac{1}{Z\beta^2}\int_0^\beta d\lambda\int_0^\lambda d\lambda' Tr[e^{-(\beta-\lambda)\hat{H}}\hat{A}(-\tau)e^{-(\lambda-\lambda')\hat{H}}\hat{B}(t-\tau)e^{-\lambda'\hat{H}}\widehat{C}(t)] \qquad (136)$$

, looking at the just the trace

$$I = Tr[e^{-(\beta-\lambda)\hat{H}}\hat{A}(-\tau)e^{-(\lambda-\lambda')\hat{H}}\hat{B}(t-\tau)e^{-\lambda'\hat{H}}\widehat{C}(t)] \quad (137)$$

$$= Tr[e^{-(\beta-\lambda)\hat{H}}e^{-i\hat{H}\tau/\hbar}\hat{A}e^{i\hat{H}\tau/\hbar}e^{-(\lambda-\lambda')\hat{H}}e^{i\hat{H}(t-\tau)/\hbar}\hat{B}e^{-i\hat{H}(t-\tau)/\hbar}e^{-\lambda'\hat{H}}e^{i\hat{H}t/\hbar}\widehat{C}e^{i\hat{H}t/\hbar}] \quad (138)$$

using the fact the Hamiltonian commutes with itself and using cyclic permutations of the trace it follows that

$$I = Tr[e^{-(\beta-\lambda)\hat{H}}\hat{A}e^{-(\lambda-\lambda')\hat{H}}\hat{B}(t)e^{-\lambda'\hat{H}}\widehat{C}(t+\tau)] \quad (139)$$

which when substituted back into the integral gives

$$\langle A(-\tau)B(t-\tau)C(t)\rangle = \langle A(0)B(t)C(t+\tau)\rangle \quad (140)$$

. This can be seen as a multi-time version of the standard detailed balance relationship $\langle A(0)B(t)\rangle = \langle A(-t)B(0)\rangle$. Using this result it is easy to show that $K^{sym}_{ABC}(t,t')$ also obeys this relationship.

13 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

$$\begin{aligned} i\hbar\frac{\partial\Psi_t(q)}{\partial t} &= \hat{H}\Psi_t(q), \\ &= -\frac{\hbar^2}{2m}\frac{\partial^2\Psi_t(q)}{\partial q^2} + V(q)\Psi_t(q), \end{aligned} \quad (141)$$

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. (141).

13.1 Hamilton Jacobi Equation

We consider the Hamiltonian

$$H(q,p) = \frac{p^2}{2m} + V(q), \quad (142)$$

and we define a canonical transformation in terms of the classical action

$$S = S(q, P, t) = \int_{t_i}^{t_f} dt'[p(t')\dot{q}(t') - H(p(t'), q(t'))], \tag{143}$$

as follows:

$$\begin{aligned} p &= \frac{S(q, P, t)}{\partial q}, \\ Q &= \frac{S(q, P, t)}{\partial P}, \end{aligned} \tag{144}$$

with

$$\boxed{\frac{\partial S(q, P, t)}{\partial t} + H(q, \frac{\partial S(q, P, t)}{\partial q}) = 0.} \tag{145}$$

and

$$\widetilde{H}(Q, P, t) \equiv \frac{\partial S(q, P, t)}{\partial t} + H(q, \frac{\partial S(q, P, t)}{\partial q}, \tag{146}$$

the Hamiltonian of the system for the transform variables Q and P. We note that the new conjugate variables Q and P are constant in time since according to Eqs. (145) and (146), $\widetilde{H}(Q, P, t) = 0$. Therefore,

$$\begin{aligned} \dot{Q} &= \frac{\partial \widetilde{H}(Q, P, t)}{\partial P} = 0, \\ \dot{P} &= -\frac{\partial \widetilde{H}(Q, P, t)}{\partial Q} = 0. \end{aligned} \tag{147}$$

For the free particle, with $x(t) = x_i + (x_f - x_i)/(t_f - t_i) * (t - t_i)$, the action is a function of the final coordinates, as follows:

$$S_{x_i,t_i}(x_f, t_f) = \frac{1}{2} m \frac{(x_f - x_i)^2}{t_f - t_i}, \tag{148}$$

so

$$\frac{\partial S}{\partial x_f} = m \frac{(x_f - x_i)^2}{t_f - t_i} = p, \tag{149}$$

and

$$\frac{\partial S}{\partial t_f} = -\frac{1}{2} m \frac{(x_f - x_i)^2}{(t_f - t_i)^2} = -\frac{p^2}{2m} = -E. \tag{150}$$

Equation (145) is the *Hamilton-Jacobi* equation and can be used to find the classical trajectory of coordinates and momenta $q(t)$ and $p(t)$, as follows: First, solve Eq. (145) for $S(q, P, t)$. Then, compute $p(t) = \partial S/\partial q$ by partial differentiation as defined in Eq. (351). Finally, obtain $q(t)$ by first computing $Q = \partial S/\partial P$ and then solving for $q(t)$ as a function of Q and P that are constant in time (Eq. (147)). We illustrate this method, we consider a Harmonic oscillator with Hamiltonian

$H(q,p) = p^2/(2m) + m\omega^2 x^2/2$, and initial conditions $x(0) = 0$ and $p(0) = \sqrt{2Em}$ for which we check the solution $x(t) = \sqrt{2E/(m\omega^2)}sin(\omega t)$. The Hamilton-Jacobi equation is

$$\frac{\partial S}{\partial t} + \frac{1}{2m}\left(\frac{\partial S}{\partial x}\right)^2 + \frac{1}{2}m\omega^2 x^2 = 0, \tag{151}$$

which has a solution of the form $S(q,t) = \tilde{S}(q,E) - Et$, since the equation does not involve t explicitly. Therefore,

$$\frac{1}{2m}\left(\frac{\partial \tilde{S}}{\partial x}\right)^2 + \frac{1}{2}m\omega^2 x^2 = E. \tag{152}$$

Solving for $\tilde{S}$, we obtain:

$$\tilde{S} = \int dx\sqrt{2mE - m^2\omega^2 x^2}. \tag{153}$$

Introducing the change of variables $sin(\phi) = \sqrt{m/(2E)}\omega x$, with $cos(\phi)d\phi = \sqrt{m/(2E)}\omega dx$, we obtain:

$$\begin{aligned}
\tilde{S} &= \frac{2E}{\omega}\int d\phi cos(\phi)\sqrt{1 - sin^2(\phi)} \\
&= \frac{2E}{\omega}\int d\phi cos^2(\phi), \\
&= \frac{E}{\omega}\int d\phi(1 + cos(2\phi)), \\
&= \frac{E}{\omega}(\phi + \frac{sin(2\phi)}{2}), \\
&= \frac{E}{\omega}\left[arcsin\left(\omega x\sqrt{\frac{m}{2E}}\right) + \sqrt{m/(2E)}\omega x\sqrt{1 - m\omega^2 x^2/(2E)}\right].
\end{aligned} \tag{154}$$

Having found $\tilde{S}$, we can obtain $x(t)$, as follows:

$$\begin{aligned}
t &= \frac{\partial \tilde{S}}{\partial E}, \\
&= \frac{1}{\omega}arcsin\left(\omega x\sqrt{\frac{m}{2E}}\right), \\
sin(\omega t) &= x\sqrt{\frac{m\omega^2}{2E}},
\end{aligned} \tag{155}$$

and p, as follows:

$$\begin{aligned}
p &= \frac{\partial \tilde{S}}{\partial x}, \\
&= \sqrt{2mE - m^2\omega^2 x^2},
\end{aligned} \tag{156}$$

as expected.

13.2 Quantum Dynamics: Motion of Hidden Variables

To solve the time-dependent Schrödinger equation, introduced by Eq. (141), we write $\Psi_t(x)$ in terms of the amplitude $A_t(x)$ and phase $S_t(x)$ functions, as follows:

$$\Psi_t(q) = A_t(q)e^{\frac{i}{\hbar}S_t(q)}, \tag{157}$$

where $A_t(q)$ and $S_t(q)$ are defined as real functions. Substituting Eq. (157) into Eq. (141), we obtain:

$$\begin{aligned} i\hbar\frac{\partial\Psi_t(q)}{\partial t} &= \left[i\hbar\frac{\partial A_t(q)}{\partial t} - A_t(q)\frac{\partial S_t(q)}{\partial t}\right]e^{\frac{i}{\hbar}S_t(q)}, \\ \hat{H}\Psi_t(q) &= \left[-\frac{\hbar^2}{2m}\frac{\partial^2 A}{\partial q^2} - \frac{i\hbar}{m}\frac{\partial A}{\partial q}\frac{\partial S}{\partial q} - \frac{i\hbar}{2m}A\frac{\partial^2 S}{\partial q^2} + \frac{1}{2m}A_t(q)\left(\frac{\partial S}{\partial q}\right)^2 + A(q)V(q)\right]e^{\frac{i}{\hbar}S_t(q)}. \end{aligned} \tag{158}$$

Since the first line of Eq. (158) must be equal to the second line, the real parts of the left hand sides of Eq. (158) must be equal:

$$\frac{\partial S_t(q)}{\partial t} + \frac{1}{2m}\left(\frac{\partial S}{\partial q}\right)^2 + V(q) + V_Q(q,t) = 0, \tag{159}$$

where

$$V_Q(q,t) = -\frac{\hbar^2}{2m}\frac{\partial^2 A_t(q)}{\partial q^2}\frac{1}{A_t(q)}, \tag{160}$$

is a time-dependent "quantum" potential determined by A_t. Note that Eq. (159) is the Hamilton-Jacobi equation for a system described by the Hamiltonian:

$$H(q,p) = \frac{p^2}{2m} + V(q) + V_Q(q,t), \tag{161}$$

with $p = \partial S/\partial q$, and $V_Q(q,t)$ the time-dependent "external" field potential defined by Eq. (160).

Furthermore, since the imaginary parts of the left hand sides of Eq. (158) must be equal, we obtain:

$$\frac{\partial A_t(q)}{\partial t} + \frac{\partial A_t(q)}{\partial q}\frac{\partial S_t(q)}{\partial q}\frac{1}{m} + A_t(q)\frac{\partial^2 S_t(q)}{\partial q^2}\frac{1}{2m} = 0. \tag{162}$$

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. (162), gives:

$$\begin{aligned} \frac{1}{2}\rho_t^{-1/2}\frac{\partial\rho_t}{\partial t} + \frac{1}{2}\rho_t^{-1/2}\frac{\partial\rho_t}{\partial q}\frac{\partial S_t(q)}{\partial q}\frac{1}{m} + \rho_t^{1/2}\frac{\partial^2 S_t(q)}{\partial q^2}\frac{1}{2m} &= 0, \\ \frac{\partial\rho_t}{\partial t} + \frac{\partial\rho_t}{\partial q}\frac{\partial S_t(q)}{\partial q}\frac{1}{m} + \rho_t\frac{\partial^2 S_t(q)}{\partial q^2}\frac{1}{m} &= 0, \\ \frac{\partial\rho_t}{\partial t} + \frac{\partial j_t}{\partial q} &= 0 \end{aligned} \tag{163}$$

that is the continuity equation for the classical current $j_t = \rho_t v$, with $v = \frac{p}{m}$.

13.3 Discussion of Bohmian Trajectories

The equations of motion of the auxiliary variables $q(t)$ and $p(t)$ are Hamilton's equations

$$\begin{aligned} \dot{q}(t) &= \frac{\partial H(q,p)}{\partial p}, \\ \dot{p}(t) &= -\frac{\partial H(q,p)}{\partial q}, \end{aligned} \tag{164}$$

with $H(q,p)$ defined according to Eq. (161). 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.

13.4 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].

14 Phase-Space Representation

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]).

Definition and Properties: Given a wavefunction $\Psi_t(x)$, the Wigner transform $\rho_t^W(p,q)$ is defined as follows:

$$\rho_t^W(p,q) \equiv \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\Psi_t^*(q+s/2)\Psi_t(q-s/2). \tag{165}$$

This quantity is similar to the phase-space probability density, since it gives the probability density $|\Psi_t(q)|^2$ when integrated with respect to p:

$$\begin{aligned}\int_{-\infty}^{\infty} dp\, \rho_t^W(p,q) &= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} dp \int_{-\infty}^{\infty} ds\, e^{ips/\hbar}\Psi_t^*(q+s/2)\Psi_t(q-s/2),\\ &= \int_{-\infty}^{\infty} ds\, \delta(s)\Psi_t^*(q+s/2)\Psi_t(q-s/2),\\ &= \Psi_t^*(q)\Psi_t(q),\\ &= |\Psi_t(q)|^2.\end{aligned} \tag{166}$$

In addition, $\rho_t^W(p,q)$ gives the Fourier transform probability density $|\widetilde{\Psi}_t(p)|^2$ when integrated with respect to q (where $\widetilde{\Psi}$ is the Fourier transform of Ψ):

$$\begin{aligned}\int_{-\infty}^{\infty} dq\rho_t^W(p,q) &= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} dq \int_{-\infty}^{\infty} ds\, e^{ips/\hbar}\Psi_t^*(q+s/2)\Psi_t(q-s/2),\\ &= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} dx' \int_{-\infty}^{\infty} dx e^{ip(x-x')/\hbar}\Psi_t^*(x)\Psi_t(x'),\\ &= \left|\frac{1}{\sqrt{2\pi\hbar}}\int_{-\infty}^{\infty} dx e^{-ipx/\hbar}\Psi_t(x)\right|^2,\\ &= |\widetilde{\Psi}_t(p)|^2.\end{aligned} \tag{167}$$

where in the second line of Eq. (167) 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:

$$E_t = \frac{\int dq \int dp \rho_t^W(p,q)H(q,p)}{\int dq \int dp \rho_t^W(p,q)}. \tag{168}$$

Therefore, $\rho_t^W(q,p)$ 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.

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. (165), and substituting the time-derivative to the wavefunctions by using the time-dependent Schrödinger equation

$$\begin{aligned}\frac{\partial \Psi_t(q \pm s/2)}{\partial t} &= -\frac{1}{i\hbar}\frac{\hbar^2}{2m}\frac{\partial^2 \Psi_t(q \pm s/2)}{\partial q^2} + \frac{1}{i\hbar}V(q \pm s/2)\Psi_t(q \pm s/2), \\ &= -\frac{1}{i\hbar}\frac{\hbar^2}{2m}4\frac{\partial^2 \Psi_t(q \pm s/2)}{\partial s^2} + \frac{1}{i\hbar}V(q \pm s/2)\Psi_t(q \pm s/2),\end{aligned} \tag{169}$$

where in the second line of Eq. (170) we have made the substitution $\frac{\partial \Psi_t^*(q\pm s/2)}{\partial q} = \pm 2\frac{\partial \Psi_t^*(q\pm s/2)}{\partial s}$. Thus, the time-derivative of the Wigner transform is

$$\begin{aligned}\frac{\partial \rho_t^W(p,q)}{\partial t} &= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\left[\frac{\partial \Psi_t^*(q+s/2)}{\partial t}\Psi_t(q-s/2) + \Psi_t^*(q+s/2)\frac{\partial \Psi_t(q-s/2)}{\partial t}\right], \\ &= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\{\frac{i}{\hbar}\left[V(q+s/2) - V(q-s/2)\right]\Psi_t^*(q+s/2)\Psi_t(q-s/2) \\ &\quad - \frac{i}{\hbar}\frac{\hbar^2}{2m}4\left[\Psi_t(q-s/2)\frac{\partial^2}{\partial s^2}\Psi_t^*(q+s/2) - \Psi_t^*(q+s/2)\frac{\partial^2}{\partial s^2}\Psi_t(q-s/2)\right]\}, \\ &= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\{\frac{i}{\hbar}\left[V(q+s/2) - V(q-s/2)\right]\Psi_t^*(q+s/2)\Psi_t(q-s/2) \\ &\quad - \frac{i}{\hbar}\frac{\hbar^2}{2m}4\frac{\partial}{\partial s}\left[\Psi_t(q-s/2)\frac{\partial}{\partial s}\Psi_t^*(q+s/2) - \Psi_t^*(q+s/2)\frac{\partial}{\partial s}\Psi_t^*(q-s/2)\right]\}, \\ &= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\{\frac{i}{\hbar}\left[V(q+s/2) - V(q-s/2)\right]\Psi_t^*(q+s/2)\Psi_t(q-s/2) \\ &\quad - \frac{i}{\hbar}\frac{\hbar^2}{2m}2\frac{\partial}{\partial s}\left[\Psi_t(q-s/2)\frac{\partial}{\partial q}\Psi_t^*(q+s/2) + \Psi_t^*(q+s/2)\frac{\partial}{\partial q}\Psi_t^*(q-s/2)\right]\}.\end{aligned} \tag{170}$$

Therefore,

$$\begin{aligned}\frac{\partial \rho_t^W(p,q)}{\partial t} &= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\{\frac{i}{\hbar}\left[V(q+s/2) - V(q-s/2)\right]\Psi_t^*(q+s/2)\Psi_t(q-s/2) \\ &\quad + \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\frac{\hbar}{mi}\frac{\partial}{\partial s}\frac{\partial}{\partial q}\left[\Psi_t(q-s/2)\Psi_t^*(q+s/2)\right]\}.\end{aligned} \tag{171}$$

Integrating by parts the second line of Eq. (171), using $\int_a^b udv = uv|_a^b - \int_a^b vdu$ with $u \equiv e^{\frac{i}{\hbar}ps}$ and

$v \equiv \frac{\hbar}{mi}\frac{\partial}{\partial q}\left[\Psi_t(q-s/2)\Psi_t^*(q+s/2)\right]$, we obtain:

$$\begin{aligned}\frac{\partial \rho_t^W(p,q)}{\partial t} &= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\frac{i}{\hbar}\left[V(q+s/2)-V(q-s/2)\right]\Psi_t^*(q+s/2)\Psi_t(q-s/2) \\ &\quad - \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\frac{p}{m}\frac{\partial}{\partial q}\left[\Psi_t(q-s/2)\Psi_t^*(q+s/2)\right], \\ &= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\frac{i}{\hbar}\left[V(q+s/2)-V(q-s/2)\right]\Psi_t^*(q+s/2)\Psi_t(q-s/2) \\ &\quad - \frac{p}{m}\frac{\partial \rho_t^W(p,q)}{\partial q}.\end{aligned} \tag{172}$$

14.1 Motion of Auxiliary Variables:

To find a solution of Eq. (172) 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:

$$\Psi_t^*(q+s/2)\Psi_t(q-s/2) \equiv \int dR_0 \int dP_0 e^{-\frac{i}{\hbar}P_t s}\delta(R_t-q)\rho_0^W(P_0,R_0). \tag{173}$$

Substituting Eq. (173) into Eq. (172) we obtain:

$$\begin{aligned}\frac{\partial \rho_t^W(p,q)}{\partial t} &= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds \int dR_0 \int dP_0\, e^{\frac{i}{\hbar}(p-P_t)s}\{\frac{i}{\hbar}\left[V(q+s/2)-V(q-s/2)\right]\delta(R_t-q) \\ &\quad - \frac{p}{m}\frac{\partial \delta(R_t-q)}{\partial q}\}\rho_0^W(P_0,R_0).\end{aligned} \tag{174}$$

Furthermore, substituting Eq. (173) into Eq. (165) gives:

$$\rho_t^W(p,q) = \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds \int dR_0 \int dP_0\, e^{\frac{i}{\hbar}(p-P_t)s}\delta(R_t-q)\rho_0^W(P_0,R_0), \tag{175}$$

and computing the time-derivative of both sides of Eq. (175) we obtain:

$$\begin{aligned}\frac{\partial \rho_t^W(p,q)}{\partial t} &= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds \int dR_0 \int dP_0\, e^{\frac{i}{\hbar}(p-P_t)s}\left[-\frac{i}{\hbar}\frac{\partial P_t}{\partial t}s\delta(R_t-q)+\frac{\partial \delta(R_t-q)}{\partial t}\right]\rho_0^W(P_0,R_0), \\ &= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds \int dR_0 \int dP_0\, e^{\frac{i}{\hbar}(p-P_t)s}\left[-\frac{i}{\hbar}\frac{\partial P_t}{\partial t}s\delta(R_t-q)+\frac{\partial \delta(R_t-q)}{\partial R_t}\frac{\partial R_t}{\partial t}\right]\rho_0^W(P_0,R_0), \\ &= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds \int dR_0 \int dP_0\, e^{\frac{i}{\hbar}(p-P_t)s}\left[-\frac{i}{\hbar}\frac{\partial P_t}{\partial t}s\delta(R_t-q)-\frac{\partial \delta(R_t-q)}{\partial q}\frac{\partial R_t}{\partial t}\right]\rho_0^W(P_0,R_0).\end{aligned} \tag{176}$$

Comparing the third line of Eq. (176) with Eq. (174) we obtain the equations of motion of {R(t),P(t)}:

$$\begin{aligned}\frac{\partial P_t}{\partial t} &= -\frac{(V(q+s/2)-V(q-s/2))}{s},\\ \frac{\partial R_t}{\partial t} &= \frac{p}{m}.\end{aligned} \tag{177}$$

In the classical limit, $\hbar \to 0$, Eqs. (177) become Hamilton's equations,

$$\begin{aligned}\frac{\partial P_t}{\partial t} &= -\frac{\partial V(q)}{\partial q},\\ \frac{\partial R_t}{\partial t} &= \frac{P_t}{m},\end{aligned} \tag{178}$$

since the most significant contributions to $\rho_t^W(p,q)$, as defined by Eq. (173), result from $P_t \approx p$ and $s \approx 0$. Therefore, in the classical limit, R_t and P_t become independent of s and

$$\rho_t^W(p,q) = \int dR_0 \int dP_0\, \delta(p-P_t)\delta(R_t-q)\rho_0^W(P_0,R_0), \tag{179}$$

that is identical to the classical evolution of the Wigner-transform.

14.2 Wigner-Weyl Symbols

The *Wigner-Weyl quantization* is another formulation of quantum mechanics in phase-space representation of quantum operators, as defined by the Wigner-Weyl transform of any Hermitian operator $\hat{\Omega}(\hat{x},\hat{p})$, introducing the corresponding *Weyl symbol*, as follows:

$$\Omega_W(x,p) = \int ds e^{\frac{i}{\hbar}ps}\langle x-\frac{s}{2}|\hat{\Omega}(\hat{x},\hat{p})|x+\frac{s}{2}\rangle. \tag{180}$$

Such a representation is possible so long as the off-diagonal matrix elements of the operator in coordinate space decay to zero. As reviewed in the (literature), we find that the Weyl symbols of operators of coordinates $\hat{\Omega}(\hat{x})$ or momenta $\hat{\Omega}(\hat{p})$ are the classical expressions of those operators:

$$\Omega_W(x) = \Omega(x), \tag{181}$$

$$\Omega_W(p) = \Omega(p). \tag{182}$$

since

$$\begin{aligned}\int ds e^{\frac{i}{\hbar}ps}\langle x-\frac{s}{2}|f(x)|x+\frac{s}{2}\rangle &= \int ds dx' e^{\frac{i}{\hbar}ps}\langle x-\frac{s}{2}|x'\rangle\langle x'|x+\frac{s}{2}\rangle f(x'),\\ &= \int ds e^{\frac{i}{\hbar}ps}\delta(s) f(x-\frac{s}{2})\end{aligned} \tag{183}$$

and

$$\begin{aligned}\int ds e^{\frac{i}{\hbar}ps}\langle x-\frac{s}{2}|f(\hat{p})|x+\frac{s}{2}\rangle &= \int dsdp' e^{\frac{i}{\hbar}ps}\langle x-\frac{s}{2}|p'\rangle\langle p'|x+\frac{s}{2}\rangle f(p') \\ &= \frac{1}{2\pi\hbar}\int dsdp' e^{\frac{i}{\hbar}(p-p')s} f(p') = f(p).\end{aligned} \tag{184}$$

More generally, the Weyl symbols of fully symmetrized polynomials of coordinates and momenta are obtained simply by substituting the operators by their corresponding classical variables $\hat{p} \to p$ and $\hat{x} \to q$.

Traces: Here, we show that traces of ensemble averages can be computed just like classical averages over the Wigner distribution, as follows:

$$\begin{aligned}\int dxdp\rho_W(x,p)A_W(x,p) &= \int dxdpdsds' e^{\frac{i}{\hbar}p(s+s')}\langle x-\frac{s}{2}|\hat{\rho}|x+\frac{s}{2}\rangle\langle x-\frac{s'}{2}|\hat{A}|x+\frac{s'}{2}\rangle, \\ &= \int dxdsds'\delta(s+s')\langle x-\frac{s}{2}|\hat{\rho}|x+\frac{s}{2}\rangle\langle x-\frac{s'}{2}|\hat{A}|x+\frac{s'}{2}\rangle, \\ &= \int dxds\langle x-\frac{s}{2}|\hat{\rho}|x+\frac{s}{2}\rangle\langle x+\frac{s}{2}|\hat{A}|x-\frac{s}{2}\rangle.\end{aligned} \tag{185}$$

Introducing the change of variables $\tilde{x} = x - \frac{s}{2}$ with $d\tilde{x} = dx$ and $\frac{\tilde{s}}{2} = x + \frac{s}{2}$ with $d\tilde{s} = ds$, we obtain:

$$\begin{aligned}\int dxdp\rho_W(x,p)A_W(x,p) &= \int d\tilde{x}d\tilde{s}\langle\tilde{x}|\hat{\rho}|\frac{\tilde{s}}{2}\rangle\langle\frac{\tilde{s}}{2}|\hat{A}|\tilde{x}\rangle, \\ &= \int d\tilde{x}\langle\tilde{x}|\hat{\rho}\hat{A}|\tilde{x}\rangle, \\ &= Tr[\hat{\rho}\hat{A}].\end{aligned} \tag{186}$$

Sums and Products: The Weyl symbol of a sum of operators is the sum of the Weyl symbols,

$$(\Omega_1+\Omega_2)_W(q,p) = (\Omega_1)_W(q,p) + (\Omega_2)_W(q,p), \tag{187}$$

since the integral of a sum is the sum of the integrals. The Weyl symbol of the product of operators, however, is less trivial. As shown below, it is defined as follows:

$$(\Omega_1\Omega_2)_W(q,p) = (\Omega_1)_W(q,p)e^{-\frac{i\hbar}{2}\Lambda}(\Omega_2)_W(q,p), \tag{188}$$

with Λ defined, as follows:

$$\Lambda = \sum_j \frac{\overleftarrow{\partial}}{\partial p_j}\frac{\partial}{\partial q_j} - \frac{\overleftarrow{\partial}}{\partial q_j}\frac{\partial}{\partial p_j}. \tag{189}$$

To derive Eq. (188), we evaluate the Wigner transform of the product of operators, as follows:

$$\begin{aligned}(\Omega_1\Omega_2)_W(x,p) &= \int ds e^{\frac{i}{\hbar}sp}\langle x-\frac{s}{2}|\hat{\Omega}_1\hat{\Omega}_2|x+\frac{s}{2}\rangle, \\ &= \int dsdx' e^{\frac{i}{\hbar}sp}\langle x-\frac{s}{2}|\hat{\Omega}_1|x'\rangle\langle x'|\hat{\Omega}_2|x+\frac{s}{2}\rangle,\end{aligned} \tag{190}$$

Writing the operators in terms of their Fourier expansions,

$$\Omega_1(\hat{x},\hat{p}) = \frac{1}{2\pi\hbar}\int d\sigma \int d\tau \alpha(\sigma,\tau) e^{i(\sigma\hat{x}+\tau\hat{p})/\hbar}, \tag{191}$$

and

$$\Omega_2(\hat{x},\hat{p}) = \frac{1}{2\pi\hbar}\int d\sigma \int d\tau \beta(\sigma,\tau) e^{i(\sigma\hat{x}+\tau\hat{p})/\hbar}, \tag{192}$$

and evaluating them in coordinate representation, as follows:

$$\langle x|\Omega_1(\hat{x},\hat{p})|x'\rangle = \frac{1}{2\pi\hbar}\int d\sigma \int d\tau \alpha(\sigma,\tau)\langle x|e^{i(\sigma\hat{x}+\tau\hat{p})/\hbar}|x'\rangle. \tag{193}$$

using the Baker-Hausdorff theorem $e^{\hat{A}+\hat{B}} = e^{\hat{A}}e^{\hat{B}}e^{-[\hat{A},\hat{B}]/2}$, with $[\hat{x},\hat{p}] = i\hbar$, we obtain:

$$\begin{aligned}
\langle x|e^{\frac{i}{\hbar}\sigma\hat{x}+\frac{i}{\hbar}\tau\hat{p}}|x'\rangle &= \langle x|e^{\frac{i}{\hbar}\sigma\hat{x}}e^{\frac{i}{\hbar}\tau\hat{p}}e^{\frac{1}{\hbar^2}\sigma\tau i\frac{\hbar}{2}}|x'\rangle, \\
&= \langle x|e^{\frac{i}{\hbar}\sigma\hat{x}}e^{\frac{i}{\hbar}\tau\hat{p}}|x'\rangle e^{\frac{i}{2\hbar}\sigma\tau}, \\
&= e^{\frac{i}{\hbar}\sigma x}\langle x+\tau|x'\rangle e^{\frac{i}{2\hbar}\sigma\tau}, \\
&= e^{\frac{i}{\hbar}\sigma x}\delta(\tau-(x'-x)e^{\frac{i}{2\hbar}\sigma\tau}.
\end{aligned} \tag{194}$$

Substituting into Eq. (193), we obtain:

$$\begin{aligned}
\langle x|\Omega_1(\hat{x},\hat{p})|x'\rangle &= \frac{1}{2\pi\hbar}\int d\sigma \int d\tau \alpha(\sigma,\tau) e^{\frac{i}{\hbar}\sigma x}\delta(\tau-(x'-x))e^{\frac{i}{2\hbar}\sigma\tau}, \\
&= \frac{1}{2\pi\hbar}\int d\sigma \alpha(\sigma,x'-x) e^{\frac{i}{\hbar}\sigma x}e^{\frac{i}{2\hbar}\sigma(x'-x)}, \\
&= \frac{1}{2\pi\hbar}\int d\sigma \alpha(\sigma,x'-x) e^{\frac{i}{2\hbar}\sigma(x'+x)},
\end{aligned} \tag{195}$$

Analogously, we obtain:

$$\langle x'|\Omega_2(\hat{x},\hat{p})|x''\rangle = \frac{1}{2\pi\hbar}\int d\sigma' \beta(\sigma',x''-x') e^{\frac{i}{2\hbar}\sigma'(x''+x')}. \tag{196}$$

Substituting Eqs. (195) and (196) into Eq. (190), we obtain:

$$\begin{aligned}
(\Omega_1\Omega_2)_W(x,p) = \frac{1}{(2\pi\hbar)^2}\int ds dx' e^{\frac{i}{\hbar}sp}\int d\sigma \alpha(\sigma,x'-x+\frac{s}{2}) e^{\frac{i}{2\hbar}\sigma(x'+x-\frac{s}{2})} \\
\times \int d\sigma' \beta(\sigma',x+\frac{s}{2}-x') e^{\frac{i}{2\hbar}\sigma'(x'+x+\frac{s}{2})}
\end{aligned} \tag{197}$$

Introducing the change of variables $\tau = x'-x+\frac{s}{2}$ and $\tau' = x+\frac{s}{2}-x' = x+\frac{s}{2}-(\tau+x-\frac{s}{2}) = s-\tau$,

we obtain:

$$
\begin{aligned}
(\Omega_1\Omega_2)_W &= \frac{1}{(2\pi\hbar)^2}\int d\tau' d\tau e^{\frac{i}{\hbar}p(\tau+\tau')}\int d\sigma\alpha(\sigma,\tau)e^{\frac{i}{2\hbar}\sigma(2x-\tau')}\int d\sigma'\beta(\sigma',\tau')e^{\frac{i}{2\hbar}\sigma'(2\tau+2x-s+\tau')},\\
&= \frac{1}{(2\pi\hbar)^2}\int d\tau d\sigma e^{\frac{i}{\hbar}(p\tau+x\sigma)}\alpha(\sigma,\tau)\int d\tau' d\sigma'\beta(\sigma',\tau')e^{\frac{i}{\hbar}(p\tau'+x\sigma')}e^{\frac{i}{2\hbar}(\sigma'\tau-\sigma\tau')},\\
&= \frac{1}{(2\pi\hbar)^2}\int d\tau d\sigma e^{\frac{i}{\hbar}(p\tau+x\sigma)}\alpha(\sigma,\tau)\int d\tau' d\sigma'\beta(\sigma',\tau')e^{\frac{i}{\hbar}(p\tau'+x\sigma')}\sum_{n=0}^{\infty}\frac{1}{n\,!}\frac{i^n}{2^n\hbar^n}(\sigma'\tau-\sigma\tau')^n,\\
&= \frac{1}{(2\pi\hbar)^2}\int d\tau d\sigma e^{\frac{i}{\hbar}(p\tau+x\sigma)}\alpha(\sigma,\tau)e^{-i\frac{\hbar}{2}\Lambda}\int d\tau' d\sigma'\beta(\sigma',\tau')e^{\frac{i}{\hbar}(p\tau'+x\sigma')},\\
&= (\Omega_1)_W e^{-i\frac{\hbar}{2}\Lambda}(\Omega_2)_W,
\end{aligned}
\tag{198}
$$

with

$$
\Lambda = \frac{\overleftarrow{\partial}}{\partial p}\frac{\partial}{\partial x} - \frac{\overleftarrow{\partial}}{\partial x}\frac{\partial}{\partial p}. \tag{199}
$$

Commutator: The Weyl symbol of a commutator of operators is given, as follows:

$$
\begin{aligned}
[\hat{\Omega}_1,\hat{\Omega}_2]_W &= (\Omega_1)_W e^{-i\frac{\hbar}{2}\Lambda}(\Omega_2)_W - (\Omega_2)_W e^{-i\frac{\hbar}{2}\Lambda}(\Omega_1)_W,\\
&= (\Omega_1)_W e^{-i\frac{\hbar}{2}\Lambda}(\Omega_2)_W - (\Omega_1)_W e^{i\frac{\hbar}{2}\Lambda}(\Omega_2)_W,\\
&= (\Omega_1)_W\left(e^{-i\frac{\hbar}{2}\Lambda} - e^{i\frac{\hbar}{2}\Lambda}\right)(\Omega_2)_W,\\
&= -i\hbar\frac{2}{\hbar}(\Omega_1)_W sin\left(\frac{\hbar}{2}\Lambda\right)(\Omega_2)_W,\\
&= -i\hbar\{\{(\Omega_1)_W,(\Omega_2)_W\}\},
\end{aligned}
\tag{200}
$$

with

$$
\{\{(\Omega_1)_W,(\Omega_2)_W\}\} = (\Omega_1)_W\frac{2}{\hbar}sin(\frac{\hbar}{2}\Lambda)(\Omega_2)_W, \tag{201}
$$

the *Moyal bracket* of the Weyl symbols of those operators (first introduced by Groenewold). We note that the Moyal bracket becomes the Poisson bracket in the classical limit when $\hbar \to 0$ since in that limit only the sine function can be expanded to first order in $\hbar$.

Equation of Motion: The quantum Liouville equation can be formulated in Weyl-Wigner quantization, as follows:

$$
\begin{aligned}
i\hbar\frac{d}{dt}(|\Psi\rangle\langle\Psi|)_W &= ([\hat{H},|\Psi\rangle\langle\Psi|])_W,\\
&= -i\hbar\{\{H_W(q,p),(|\Psi\rangle\langle\Psi|)_W(q,p)\}\},\\
&= -i\hbar H_W(q,p)\frac{2}{\hbar}sin(\frac{\hbar}{2}\Lambda)(|\Psi\rangle\langle\Psi|)_W(q,p),
\end{aligned}
\tag{202}
$$

with $(|\Psi\rangle\langle\Psi|)_W = 2\pi\hbar\rho_w(p,q)$, with ρ_w the Wigner transform of Ψ. Therefore,

$$
i\frac{\partial}{\partial t}\rho_w = \mathcal{L}\rho_w, \tag{203}
$$

with the Liouvillian defined, as follows:

$$i\mathcal{L} = H_W(q,p)\frac{2}{\hbar}sin(\frac{\hbar}{2}\Lambda). \tag{204}$$

14.3 Stochastic equation of motion

The equation of motion of introduced by Eq. (203), can also be written, according to Eq. (**??**), as follows:

$$\frac{\partial \rho^W(p,q;t)}{\partial t} = \frac{\partial_v \rho^W(p,q;t)}{\partial t} - \frac{p}{m}\frac{\partial \rho^W(p,q;t)}{\partial q}, \tag{205}$$

with

$$\begin{aligned}
\frac{\partial_v \rho^W}{\partial t} &= \int_{-\infty}^{\infty} ds\, \frac{e^{\frac{i}{\hbar}ps}}{2\pi\hbar}\frac{i}{\hbar}\left[V(q+s/2) - V(q-s/2)\right]\Psi^*(q+s/2)\Psi(q-s/2) \\
&= \int_{-\infty}^{\infty} ds\, \frac{e^{\frac{i}{\hbar}ps}}{2\pi\hbar}\frac{i}{\hbar}\Psi^*(q+s/2)\Psi(q-s/2)\int_{-\infty}^{\infty} dy\left[V(q+y/2) - V(q-y/2)\right]\delta(s-y) \\
&= \frac{1}{2\pi\hbar}\int dj \int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}(p+j)s}\Psi^*(q+s/2)\Psi(q-s/2)J(q,j) \\
&= \int dj \rho^W(p+j,q;t)J(q,j),
\end{aligned} \tag{206}$$

with

$$\begin{aligned}
J(q,j) &= \frac{i}{\hbar}\frac{1}{2\pi\hbar}\int dy\left[V(q+y/2) - V(q-y/2)\right]e^{-\frac{i}{\hbar}yj}, \\
&= \frac{1}{2\pi\hbar^2}\int dy\left[V(q+y/2) - V(q-y/2)\right]sin(yj/\hbar)
\end{aligned} \tag{207}$$

since $[V(q+y/2) - V(q-y/2)]$ is an odd function of y.

Therefore, the equation of motion of the Wigner transform can also be written, as the following stochastic differential equation:

$$\frac{\partial \rho^W(p,q;t)}{\partial t} = -\frac{p}{m}\frac{\partial \rho^W(p,q;t)}{\partial q} + \int dj \rho^W(p+j,q;t)J(q,j), \tag{208}$$

where $J(q,j)$ defines the probability of a momentum jump of size j at position q.

Another equivalent form is:

$$\frac{\partial \rho^W(p,q;t)}{\partial t} + \frac{p}{m}\frac{\partial \rho^W(p,q;t)}{\partial q} - \frac{\partial V}{\partial q}\frac{\partial \rho^W(p,q;t)}{\partial p} = \int dj \rho^W(p+j,q;t)J(q,j) - \frac{\partial V}{\partial q}\frac{\partial \rho^W(p,q;t)}{\partial p}, \tag{209}$$

or

$$\frac{d\rho^W(p,q;t)}{dt} = \int dj \rho^W(p+j,q;t)\tilde{J}(q,j), \tag{210}$$

with

$$\begin{aligned}
\tilde{J}(q,j) &= \frac{1}{2\pi\hbar^2}\int_{-\infty}^{\infty} dy\,[V(q+y/2) - V(q-y/2) - V'(q)y]\,sin(yj/\hbar), \\
&= \frac{1}{\pi\hbar^2}\sum_{n=1}^{\infty}\frac{V^{(2n+1)}(q)}{n!}\int_{-\infty}^{\infty} dy\, y^{2n+1} sin(yj/\hbar), \\
&= \frac{1}{\pi\hbar^2}\sum_{n=1}^{\infty}\frac{V^{(2n+1)}(q)}{n!}\left(\frac{\hbar}{i}\right)^{2n+1}\frac{2}{2i}\frac{\partial^{2n+1}}{\partial j^{2n+1}}\int_{-\infty}^{\infty} dy\, cos(yj/\hbar), \\
&= \frac{2}{\pi}\sum_{n=1}^{\infty}(-1)^{n+1}\frac{V^{(2n+1)}(q)}{n!}\hbar^{2n-1}\frac{\partial^{2n+1}}{\partial j^{2n+1}}\int_{0}^{\infty} dy\, cos(yj/\hbar), \\
&= \frac{2}{\pi}\sum_{n=1}^{\infty}(-1)^{n+1}\frac{V^{(2n+1)}(q)}{n!}\hbar^{2n}\frac{\partial^{2n+1}}{\partial j^{2n+1}}\lim_{y\to\infty}\frac{sin(jy/\hbar)}{j},
\end{aligned} \tag{211}$$

15 Asymmetric Wigner-Weyl Representation

A simple phase-space representation can be introduced by the following symbol of an operator $\hat{O}$:

$$O(x,p) = 2\pi\hbar\langle x|\hat{O}|p\rangle\langle p|x\rangle. \tag{212}$$

In particular, the symbol for the density operator $\hat{\rho} = |\Psi\rangle\langle\Psi|$ is defined, as follows:

$$\rho_H(x,p) = 2\pi\hbar\langle x|\Psi\rangle\langle\Psi|p\rangle\langle p|x\rangle. \tag{213}$$

giving the density in coordinates or momenta upon integration over p or x, respectively, as follows: $(2\pi\hbar)^{-1}\int dp\rho_H(x,p) = \langle x|\Psi\rangle\langle\Psi|x\rangle$ and $(2\pi\hbar)^{-1}\int dx\rho_H(x,p) = \langle p|\Psi\rangle\langle\Psi|p\rangle$.

Next, we obtain the symbol of a product of operators

$$\begin{aligned}\left(\hat{A}\hat{B}\right)_H &= 2\pi\hbar\langle x|\hat{A}\hat{B}|p\rangle\langle p|x\rangle, \\ &= 2\pi\hbar\int dx'dx''\langle x|\hat{A}|x'\rangle\langle x'|\hat{B}|x''\rangle\langle x''|p\rangle\langle p|x\rangle,\end{aligned} \tag{214}$$

To substitute $\langle x|\hat{A}|x'\rangle$ and $\langle x'|\hat{B}|x''\rangle$, we introduce the Fourier expansion of the operators $A(\hat{x},\hat{p})$ and $B(\hat{x},\hat{p})$, as follows:

$$A(\hat{x},\hat{p}) = \frac{1}{2\pi\hbar}\int d\sigma\int d\tau\alpha(\sigma,\tau)e^{i(\sigma\hat{x}+\tau\hat{p})/\hbar}, \tag{215}$$

and

$$B(\hat{x},\hat{p}) = \frac{1}{2\pi\hbar}\int d\sigma\int d\tau\beta(\sigma,\tau)e^{i(\sigma\hat{x}+\tau\hat{p})/\hbar}, \tag{216}$$

we can be evaluated in coordinate representation, as follows:

$$\langle x|A(\hat{x},\hat{p})|x'\rangle = \frac{1}{2\pi\hbar}\int d\sigma\int d\tau\alpha(\sigma,\tau)\langle x|e^{i(\sigma\hat{x}+\tau\hat{p})/\hbar}|x'\rangle. \tag{217}$$

Using the Baker-Hausdorff theorem $e^{\hat{A}+\hat{B}} = e^{\hat{A}}e^{\hat{B}}e^{-[\hat{A},\hat{B}]/2}$, with $[\hat{x},\hat{p}] = i\hbar$, we obtain

$$\begin{aligned}\langle x|e^{\frac{i}{\hbar}\sigma\hat{x}+\frac{i}{\hbar}\tau\hat{p}}|x'\rangle &= \langle x|e^{\frac{i}{\hbar}\sigma\hat{x}}e^{\frac{i}{\hbar}\tau\hat{p}}e^{\frac{1}{\hbar^2}\sigma\tau i\frac{\hbar}{2}}|x'\rangle, \\ &= \langle x|e^{\frac{i}{\hbar}\sigma\hat{x}}e^{\frac{i}{\hbar}\tau\hat{p}}|x'\rangle e^{\frac{i}{2\hbar}\sigma\tau}, \\ &= e^{\frac{i}{\hbar}\sigma x}\langle x+\tau|x'\rangle e^{\frac{i}{2\hbar}\sigma\tau}, \\ &= e^{\frac{i}{\hbar}\sigma x}\delta(\tau-(x'-x)e^{\frac{i}{2\hbar}\sigma\tau}.\end{aligned} \tag{218}$$

Substituting into Eq. (217), we obtain:

$$\begin{aligned}\langle x|A(\hat{x},\hat{p})|x'\rangle &= \frac{1}{2\pi\hbar}\int d\sigma\int d\tau\alpha(\sigma,\tau)e^{\frac{i}{\hbar}\sigma x}\delta(\tau-(x'-x))e^{\frac{i}{2\hbar}\sigma\tau}, \\ &= \frac{1}{2\pi\hbar}\int d\sigma\alpha(\sigma,x'-x)e^{\frac{i}{\hbar}\sigma x}e^{\frac{i}{2\hbar}\sigma(x'-x)}, \\ &= \frac{1}{2\pi\hbar}\int d\sigma\alpha(\sigma,x'-x)e^{\frac{i}{2\hbar}\sigma(x'+x)},\end{aligned} \tag{219}$$

Analogously, we obtain:

$$\langle x'|B(\hat{x},\hat{p})|x''\rangle = \frac{1}{2\pi\hbar}\int d\sigma'\beta(\sigma', x''-x')e^{\frac{i}{2\hbar}\sigma'(x''+x')}. \tag{220}$$

Substituting Eqs. (219) and (220) into Eq. (221), we obtain:

$$\left(\hat{A}\hat{B}\right)_H = \frac{1}{2\pi\hbar}\int dx'dx''d\sigma d\sigma'\alpha(\sigma, x'-x)e^{\frac{i}{2\hbar}\sigma(x'+x)}\beta(\sigma', x''-x')e^{\frac{i}{2\hbar}\sigma'(x''+x')}\langle x''|p\rangle\langle p|x\rangle, \tag{221}$$

Introducing the change of variables $\tau = x' - x$ and $\tau' = x'' - x' = x'' - \tau - x$, we obtain:

$$\begin{aligned}
\left(\hat{A}\hat{B}\right)_H &= \frac{1}{2\pi\hbar}\int d\tau d\tau' d\sigma d\sigma'\alpha(\sigma,\tau)e^{\frac{i}{2\hbar}\sigma(\tau+2x)}\beta(\sigma',\tau')e^{\frac{i}{2\hbar}\sigma'(\tau'+2x')}\langle\tau'+\tau+x|p\rangle\langle p|x\rangle,\\
&= \frac{1}{2\pi\hbar}\int d\tau d\tau' d\sigma d\sigma'\alpha(\sigma,\tau)e^{\frac{i}{2\hbar}\sigma(\tau+2x)}\beta(\sigma',\tau')e^{\frac{i}{2\hbar}\sigma'(\tau'+2x')}\langle\tau+\tau'|p\rangle,\\
&= \frac{1}{(2\pi\hbar)^2}\int d\tau d\tau' d\sigma d\sigma'\alpha(\sigma,\tau)e^{\frac{i}{2\hbar}\sigma(\tau+2x)}\beta(\sigma',\tau')e^{\frac{i}{2\hbar}\sigma'(\tau'+2x')}e^{\frac{i}{\hbar}\tau p}e^{\frac{i}{\hbar}\tau' p},\\
&= \frac{1}{(2\pi\hbar)^2}\int d\tau d\sigma\alpha(\sigma,\tau)e^{\frac{i}{\hbar}(\sigma x+p\tau)}e^{\frac{i}{2\hbar}\sigma\tau}\int d\tau' d\sigma'\beta(\sigma',\tau')e^{\frac{i}{\hbar}(\sigma' x'+\tau' p)}e^{\frac{i}{2\hbar}\sigma'\tau'},\\
&= \frac{1}{(2\pi\hbar)^2}\int d\tau d\sigma\alpha(\sigma,\tau)e^{\frac{i}{\hbar}(\sigma x+p\tau)}e^{\frac{i}{2\hbar}\sigma\tau}\int d\tau' d\sigma'\beta(\sigma',\tau')e^{\frac{i}{\hbar}(\sigma' x+\tau' p)}e^{\frac{i}{2\hbar}\sigma'\tau'}e^{\frac{i}{\hbar}\sigma'\tau},
\end{aligned} \tag{222}$$

which gives

$$\left(\hat{A}\hat{B}\right)_H = A_H e^{\frac{\hbar}{i}\frac{\overleftarrow{\partial}}{\partial p}\frac{\overrightarrow{\partial}}{\partial x}}B_H, \tag{223}$$

since $A_H = 2\pi\hbar\int dx'\langle x|\hat{A}|x'\rangle\langle x'|p\rangle\langle p|x\rangle$, and according to Eq. (219),

$$\begin{aligned}
A_H &= (2\pi\hbar)^{-1}\int d\sigma\int d\tau\alpha(\sigma,\tau)e^{\frac{i}{2\hbar}\sigma\tau}e^{\frac{i}{\hbar}(p\tau+\sigma x)},\\
B_H &= (2\pi\hbar)^{-1}\int d\sigma'\int d\tau'\beta(\sigma',\tau')e^{\frac{i}{2\hbar}\sigma'\tau'}e^{\frac{i}{\hbar}(p\tau'+\sigma' x)},
\end{aligned} \tag{224}$$

and

$$\begin{aligned}
e^{\frac{i}{\hbar}p\tau}e^{\frac{i}{\hbar}\sigma'\tau}e^{\frac{i}{\hbar}\sigma' x} &= e^{\frac{i}{\hbar}p\tau}\sum_{n=0}^{\infty}\frac{1}{n!}\frac{i^n}{\hbar^n}\sigma'^n\tau^n e^{\frac{i}{\hbar}\sigma' x},\\
&= e^{\frac{i}{\hbar}p\tau}e^{\frac{\hbar}{i}\frac{\overleftarrow{\partial}}{\partial p}\frac{\overrightarrow{\partial}}{\partial x}}e^{\frac{i}{\hbar}\sigma' x}.
\end{aligned} \tag{225}$$

Equation 223 can be used to obtain the Liouville equation in phase-space representation, as follows:

$$i\hbar\frac{\partial\hat{\rho}}{\partial t} = [\hat{H},\hat{\rho}], \tag{226}$$

giving

$$i\hbar\frac{\partial\rho_H}{\partial t} = H_H e^{\frac{\hbar}{i}\frac{\overleftarrow{\partial}}{\partial p}\frac{\overrightarrow{\partial}}{\partial x}}\rho_H - \rho_H e^{\frac{\hbar}{i}\frac{\overleftarrow{\partial}}{\partial p}\frac{\overrightarrow{\partial}}{\partial x}}H_H. \tag{227}$$

Expanding the exponentials, we obtain

$$\begin{aligned} i\hbar\frac{\partial\rho_H}{\partial t} &= H_H\sum_{n=0}^{\infty}\frac{1}{n!}\frac{\hbar^n}{i^n}\frac{\overleftarrow{\partial^n}}{\partial p^n}\frac{\overrightarrow{\partial^n}}{\partial x^n}\rho_H - \rho_H\sum_{n=0}^{\infty}\frac{1}{n!}\frac{\hbar^n}{i^n}\frac{\overleftarrow{\partial^n}}{\partial p^n}\frac{\overrightarrow{\partial^n}}{\partial x^n}H_H, \\ &= H_H\rho_H + \frac{\hbar}{i}\frac{p}{m}\frac{\partial\rho_H}{\partial x} - \hbar^2\frac{1}{2m}\frac{\partial^2\rho_H}{\partial x^2} - H_H\rho_H - \frac{\partial}{\partial x}V(x)\frac{\hbar}{i}\frac{\partial\rho_H}{\partial p} - \sum_{n=2}^{\infty}\frac{1}{n!}\frac{\partial^n}{\partial x^n}V(x)\frac{\hbar^n}{i^n}\frac{\partial^n\rho_H}{\partial p^n}, \\ &= \frac{\hbar}{i}\frac{p}{m}\frac{\partial\rho_H}{\partial x} - \hbar^2\frac{1}{2m}\frac{\partial^2\rho_H}{\partial x^2} - \frac{\partial V}{\partial x}\frac{\hbar}{i}\frac{\partial\rho_H}{\partial p} - \sum_{n=2}^{\infty}\frac{1}{n!}\frac{\partial^n V}{\partial x^n}\frac{\hbar^n}{i^n}\frac{\partial^n\rho_H}{\partial p^n}, \end{aligned} \tag{228}$$

or

$$i\hbar\left(\frac{\partial\rho_H}{\partial t} + \frac{p}{m}\frac{\partial\rho_H}{\partial x} - \frac{\partial V}{\partial x}\frac{\partial\rho_H}{\partial p}\right) = -\frac{\hbar^2}{2m}\frac{\partial^2\rho_H}{\partial x^2} - \sum_{n=2}^{\infty}\frac{1}{n!}\frac{\partial^n V}{\partial x^n}\frac{\hbar^n}{i^n}\frac{\partial^n\rho_H}{\partial p^n}. \tag{229}$$

15.1 Traces

Furthermore, we can show that $Tr[\hat{\rho}\,\hat{p}^m\hat{x}^n] = (2\pi\hbar)^{-1}\int dxdp\, p^m x^n\rho_H$, as follows:

$$\begin{aligned} (2\pi\hbar)^{-1}\int dxdp\, p^m x^n\rho_H &= \int dxdp\, p^m x^n\langle x|\Psi\rangle\langle p|x\rangle\langle\Psi|p\rangle, \\ &= \int dxdp\langle x|\hat{x}^n|\Psi\rangle\langle p|x\rangle\langle\Psi|\hat{p}^m|p\rangle, \\ &= \int dx\langle x|\hat{x}^n|\Psi\rangle\langle\Psi|\hat{p}^m|x\rangle, \\ &= Tr[\hat{x}^n|\Psi\rangle\langle\Psi|\hat{p}^m], \\ &= Tr[\hat{\rho}\,\hat{p}^m\hat{x}^n]. \end{aligned} \tag{230}$$

More generally, $Tr[\hat{p}^m\hat{x}^n] = (2\pi\hbar)^{-1}\int dxdp\, p^m x^n$, since $\hat{1} = \sum_j |\Psi_j\rangle\langle\Psi_j|$ and

$$\begin{aligned} Tr[\hat{p}^m\hat{x}^n] &= \sum_j Tr[\hat{x}^n|\Psi_j\rangle\langle\Psi_j|\hat{p}^m] \\ &= (2\pi\hbar)^{-1}\sum_j\int dxdp\, p^m x^n\rho_H^j, \\ &= \int dxdp\, p^m x^n\langle x|\sum_j|\Psi_j\rangle\langle\Psi_j|p\rangle\langle p|x\rangle, \\ &= \int dxdp\, p^m x^n\langle x|p\rangle\langle p|x\rangle, \\ &= (2\pi\hbar)^{-1}\int dxdp\, p^m x^n. \end{aligned} \tag{231}$$

15.2 Equation of Motion

The equation of motion of $\rho_H(x,p)$ can be obtained, as follows:

$$\frac{i}{2\pi}\frac{\partial}{\partial t}\rho_H(x,p) = \langle x|\hat{H}|\Psi\rangle\langle\Psi|p\rangle\langle p|x\rangle - \langle x|\Psi\rangle\langle\Psi|\hat{H}|p\rangle\langle p|x\rangle, \tag{232}$$

giving

$$\begin{aligned}\frac{i}{2\pi}\frac{\partial}{\partial t}\rho_H(x,p) &= V(x)\langle x|\Psi\rangle\langle\Psi|p\rangle\langle p|x\rangle - \sum_{n=0}^{\infty}\frac{1}{n!}V^{(n)}(x)\langle x|\Psi\rangle\frac{\hbar^n}{i^n}\frac{\partial^n}{\partial p^n}\int dx'\langle\Psi|x'\rangle e^{\frac{i}{\hbar}p(x'-x)} \\ &+ \int d\tilde{p}\frac{\tilde{p}^2}{2m}\langle x|\tilde{p}\rangle\langle\tilde{p}|\Psi\rangle\langle\Psi|p\rangle\langle p|x\rangle - \langle x|\Psi\rangle\langle\Psi|p\rangle\langle p|x\rangle\frac{p^2}{2m},\end{aligned} \tag{233}$$

Expanding the kinetic energy in powers of $\tilde{p}-p$, we obtain:

$$\begin{aligned}\frac{i}{2\pi}\frac{\partial}{\partial t}\rho_H(x,p) &= V(x)\langle x|\Psi\rangle\langle\Psi|p\rangle\langle p|x\rangle - \sum_{n=0}^{\infty}\frac{1}{n!}V^{(n)}(x)\langle x|\Psi\rangle\frac{\hbar^n}{i^n}\frac{\partial^n}{\partial p^n}\int dx'\langle\Psi|x'\rangle e^{\frac{i}{\hbar}p(x'-x)} \\ &+ \frac{1}{2m}\int d\tilde{p}(-p^2+\tilde{p}^2)\langle x|\tilde{p}\rangle\langle\tilde{p}|\Psi\rangle\langle\Psi|p\rangle\langle p|x\rangle,\end{aligned} \tag{234}$$

or

$$\frac{i}{2\pi}\frac{\partial}{\partial t}\rho_H(x,p) = -\sum_{n=1}^{\infty}\frac{1}{n!}V^{(n)}(x)\frac{\hbar^n}{i^n}\frac{\partial^n\rho_H}{\partial p^n}\frac{1}{2\pi\hbar} - \frac{p^2}{2m}\frac{\rho_H}{2\pi\hbar} - \hbar^2\frac{\langle\Psi|p\rangle\langle p|x\rangle}{2m}\frac{\partial^2}{\partial x^2}\langle x|\Psi\rangle. \tag{235}$$

We note that

$$\frac{\partial}{\partial x}\rho_H(x,p) = 2\pi\hbar\langle x|\frac{\partial\Psi}{\partial x}\rangle\langle\Psi|p\rangle\langle p|x\rangle - \rho_H\frac{i}{\hbar}p, \tag{236}$$

and

$$\begin{aligned}\frac{\partial^2}{\partial x^2}\rho_H(x,p) &= 2\pi\hbar\langle x|\frac{\partial^2\Psi}{\partial x^2}\rangle\langle\Psi|p\rangle\langle p|x\rangle - \rho_H\frac{1}{\hbar^2}p^2 - i4\pi p\langle x|\frac{\partial\Psi}{\partial x}\rangle\langle\Psi|p\rangle\langle p|x\rangle, \\ &= 2\pi\hbar\langle x|\frac{\partial^2\Psi}{\partial x^2}\rangle\langle\Psi|p\rangle\langle p|x\rangle - \rho_H\frac{1}{\hbar^2}p^2 - \frac{i}{\hbar}2p(\frac{\partial}{\partial x}\rho_H(x,p) + \rho_H\frac{i}{\hbar}p).\end{aligned} \tag{237}$$

Therefore,

$$-2\pi\hbar\frac{\hbar^2}{2m}\langle\Psi|p\rangle\langle p|x\rangle\langle x|\frac{\partial^2\Psi}{\partial x^2}\rangle - \rho_H\frac{p^2}{2m} = -\frac{\hbar^2}{2m}\frac{\partial^2\rho_H}{\partial x^2} - i\hbar\frac{p}{m}\frac{\partial\rho_H}{\partial x}. \tag{238}$$

Substituting the expression above into Eq. (235), we obtain:

$$i\hbar\left(\frac{\partial\rho_H}{\partial t} + \frac{p}{m}\frac{\partial\rho_H}{\partial x} - \frac{\partial V}{\partial x}\frac{\partial\rho_H}{\partial p}\right) = -\sum_{n=2}^{\infty}\frac{1}{n!}V^{(n)}(x)\frac{\hbar^n}{i^n}\frac{\partial^n\rho_H}{\partial p^n} - \frac{\hbar^2}{2m}\frac{\partial^2\rho_H}{\partial x^2}, \tag{239}$$

which can also be written, as follows:

$$\rho_H(t) = \rho_H(0)exp(-\frac{i}{\hbar}\int_0^t dt'\left[-\sum_{n=2}^{\infty}\frac{1}{n!}V^{(n)}(x)\frac{\hbar^n}{i^n}\frac{1}{\rho_H}\frac{\partial^n\rho_H}{\partial p^n} - \frac{\hbar^2}{2m}\frac{1}{\rho_H}\frac{\partial^2\rho_H}{\partial x^2}\right]). \tag{240}$$

15.3 Stochastic Equation of Motion

The equation of motion of $\rho_H(x,p)$ can also be written as a stochastic equation with momentum jumps, as follows.

$$\begin{aligned}\frac{i}{2\pi}\frac{\partial}{\partial t}\rho_H(x,p) &= \langle x|\hat{H}|\Psi\rangle\langle\Psi|p\rangle\langle p|x\rangle - \langle x|\Psi\rangle\langle\Psi|\hat{H}|p\rangle\langle p|x\rangle, \\ &= \frac{i}{2\pi}\frac{\partial_v}{\partial t}\rho_H(x,p) + \int d\tilde{p}\frac{\tilde{p}^2}{2m}\langle x|\tilde{p}\rangle\langle\tilde{p}|\Psi\rangle\langle\Psi|p\rangle\langle p|x\rangle - \langle x|\Psi\rangle\langle\Psi|p\rangle\langle p|x\rangle\frac{p^2}{2m},\end{aligned} \tag{241}$$

where

$$\begin{aligned}\frac{i}{2\pi}\frac{\partial_v}{\partial t}\rho_H(x,p) &= V(x)\langle x|\Psi\rangle\langle\Psi|p\rangle\langle p|x\rangle - \int dsV(s)\langle x|\Psi\rangle\langle\Psi|s\rangle\langle s|p\rangle\langle p|x\rangle, \\ &= \frac{V(x)}{2\pi\hbar}\rho_H(x,p) - \int dsV(s)\langle x|\Psi\rangle\langle\Psi|s\rangle\langle s|p\rangle\langle p|x\rangle, \\ &= \frac{V(x)}{2\pi\hbar}\rho_H(x,p) - \int dyV(y)\int ds\delta(s-y)\langle x|\Psi\rangle\langle\Psi|s\rangle\langle s|p\rangle\langle p|x\rangle, \\ &= \frac{V(x)}{2\pi\hbar}\rho_H(x,p) - \frac{1}{2\pi\hbar}\int dj\int dyV(y)\int ds e^{\frac{i}{\hbar}j(s-y)}\langle x|\Psi\rangle\langle\Psi|s\rangle\langle s|p\rangle\langle p|x\rangle, \\ &= \frac{V(x)}{2\pi\hbar}\rho_H(x,p) - \frac{1}{2\pi\hbar}\int dj\int dyV(y)e^{-\frac{i}{\hbar}jy}\int ds e^{\frac{i}{\hbar}js}\langle x|\Psi\rangle\langle\Psi|s\rangle\langle s|p\rangle\langle p|x\rangle, \\ &= \frac{V(x)}{2\pi\hbar}\rho_H(x,p) - \frac{1}{2\pi\hbar}\int dj\int dyV(y)e^{-\frac{i}{\hbar}jy}\int ds\langle x|\Psi\rangle\langle\Psi|s\rangle\langle s|p+j\rangle\langle p|x\rangle,\end{aligned} \tag{242}$$

so

$$\begin{aligned}\frac{i}{2\pi}\frac{\partial_v}{\partial t}\rho_H(x,p) &= \frac{V(x)}{2\pi\hbar}\rho_H(x,p) - \frac{1}{2\pi\hbar}\int dj\int dyV(y)e^{-\frac{i}{\hbar}jy}\langle x|\Psi\rangle\langle\Psi|p+j\rangle\langle p|x\rangle, \\ &= \frac{V(x)}{2\pi\hbar}\rho_H(x,p) - \frac{1}{\sqrt{2\pi\hbar}}\int dj\tilde{V}(j)\langle x|\Psi\rangle\langle\Psi|p+j\rangle\langle p|x\rangle, \\ &= \frac{V(x)}{2\pi\hbar}\rho_H(x,p) - \frac{1}{\sqrt{2\pi\hbar}}\int dj\tilde{V}(j)\langle x|\Psi\rangle\langle\Psi|p+j\rangle\langle p+j|x\rangle\langle p|x\rangle\langle x|p+j\rangle, \\ &= \frac{V(x)}{2\pi\hbar}\rho_H(x,p) - \frac{1}{\sqrt{2\pi\hbar}}\int dj\tilde{V}(j)\langle x|\Psi\rangle\langle\Psi|p+j\rangle\langle p+j|x\rangle\langle x|j\rangle, \\ &= \frac{V(x)}{2\pi\hbar}\rho_H(x,p) - \int djJ(j,x)\frac{\rho_H(x,p+j)}{2\pi\hbar},\end{aligned} \tag{243}$$

with $J(j,x)$ the 'probability' of a momentum jump of size j at position x, defined as follows:

$$\begin{aligned}J(j,x) &= \frac{1}{2\pi\hbar}\tilde{V}(j)e^{\frac{i}{\hbar}xj} \\ &= (2\pi\hbar)^{-3/2}\int dyV(y)e^{\frac{i}{\hbar}(x-y)j}\end{aligned} \tag{244}$$

Therefore,

$$i\hbar\left(\frac{\partial\rho_H}{\partial t}+\frac{p}{m}\frac{\partial\rho_H}{\partial x}\right)=-\frac{\hbar^2}{2m}\frac{\partial^2\rho_H}{\partial x^2}+V(x)\rho_H-\int djJ(j,x)\rho_H(x,p+j), \tag{245}$$

Completing the Poisson bracket, we obtain

$$\begin{aligned} i\hbar\frac{d\rho_H}{dt}&=-\frac{\hbar^2}{2m}\frac{\partial^2\rho_H}{\partial x^2}+V(x)\rho_H-i\hbar V'(x)\frac{\partial\rho_H}{\partial p}-\int djJ(j,x)\rho_H(x,p+j),\\ &=-\frac{\hbar^2}{2m}\frac{\partial^2\rho_H}{\partial x^2}+V(x)\rho_H-i\hbar V'(x)\frac{\partial\rho_H}{\partial p}-\int djJ(j,x)\rho_H(x,p+j). \end{aligned} \tag{246}$$

We note that

$$\begin{aligned} \frac{\partial\rho_H}{\partial p}&=2\pi\hbar\langle x|\Psi\rangle\frac{\partial}{\partial p}\langle\Psi|p\rangle\langle p|x\rangle,\\ &=2\pi\hbar\langle x|\Psi\rangle\int ds\langle\Psi|s\rangle\frac{\partial}{\partial p}\langle s|p\rangle\langle p|x\rangle,\\ &=\langle x|\Psi\rangle\int ds\langle\Psi|s\rangle\frac{\partial}{\partial p}e^{\frac{i}{\hbar}p(s-x)},\\ &=\frac{i}{\hbar}\langle x|\Psi\rangle\int ds\langle\Psi|s\rangle(s-x)e^{\frac{i}{\hbar}p(s-x)},\\ &=2\pi\hbar\frac{i}{\hbar}\langle x|\Psi\rangle\langle p|x\rangle\int ds\langle\Psi|s\rangle s\langle s|p\rangle-\frac{i}{\hbar}x\rho_H,\\ &=2\pi\hbar\frac{i}{\hbar}\langle x|\Psi\rangle\langle p|x\rangle\int dyy\int ds\delta(s-y)\langle\Psi|s\rangle\langle s|p\rangle-\frac{i}{\hbar}x\rho_H,\\ &=\frac{i}{\hbar}\langle x|\Psi\rangle\langle p|x\rangle\int dj\int dyy\int dse^{\frac{i}{\hbar}(s-y)j}\langle\Psi|s\rangle\langle s|p\rangle-\frac{i}{\hbar}x\rho_H,\\ &=\frac{i}{\hbar}\langle x|\Psi\rangle\langle p|x\rangle\int dj\int dyye^{-\frac{i}{\hbar}yj}\int ds\langle\Psi|s\rangle\langle s|p+j\rangle-\frac{i}{\hbar}x\rho_H,\\ &=\frac{i}{\hbar}\int dj\int dyye^{-\frac{i}{\hbar}yj}\langle x|\Psi\rangle\langle\Psi|p+j\rangle\langle p+j|x\rangle\langle x|j\rangle-\frac{i}{\hbar}x\rho_H,\\ &=\frac{i}{\hbar}\int dj\int dyy\frac{e^{-\frac{i}{\hbar}(y-x)j}}{\sqrt{2\pi\hbar}}\langle x|\Psi\rangle\langle\Psi|p+j\rangle\langle p+j|x\rangle-\frac{i}{\hbar}x\rho_H,\\ &=\frac{i}{\hbar}\int dj\int dyy\frac{e^{-\frac{i}{\hbar}(y-x)j}}{\sqrt{2\pi\hbar}}\frac{\rho_H(x,p+j)}{2\pi\hbar}-\frac{i}{\hbar}x\rho_H,\\ &=\frac{i}{\hbar}\int dj\tilde{J}(x,j)\rho_H(x,p+j)-\frac{i}{\hbar}x\rho_H, \end{aligned} \tag{247}$$

with

$$\tilde{J}(x,j)=\int dyye^{-\frac{i}{\hbar}(y-x)j}(2\pi\hbar)^{-3/2}. \tag{248}$$

Substituting into Eq. (246), we obtain

$$\begin{aligned} i\hbar\frac{d\rho_H}{dt} &= -\frac{\hbar^2}{2m}\frac{\partial^2\rho_H}{\partial x^2} + V(x)\rho_H - i\hbar V'(x)\frac{\partial\rho_H}{\partial p} - \int djJ(j,x)\rho_H(x,p+j), \\ &= -\frac{\hbar^2}{2m}\frac{\partial^2\rho_H}{\partial x^2} + V(x)\rho_H - i\hbar V'(x)(\frac{i}{\hbar}\int dj\tilde{J}(x,j)\rho_H(x,p+j) - \frac{i}{\hbar}x\rho_H) - \int djJ(j,x)\rho_H(x,p+j), \\ &= -\frac{\hbar^2}{2m}\frac{\partial^2\rho_H}{\partial x^2} + [V(x) - V'(x)x]\rho_H + \int dj[V'(x)\tilde{J}(x,j) - J(j,x)]\rho_H(x,p+j), \\ &= -\frac{\hbar^2}{2m}\frac{\partial^2\rho_H}{\partial x^2} + [V(x) - V'(x)x]\rho_H + \int dj\bar{J}(x,j)\rho_H(x,p+j), \end{aligned} \tag{249}$$

with

$$\begin{aligned} \tilde{J}(x,j) &= \int dy[V'(x)y - V(y)]e^{-\frac{i}{\hbar}(y-x)j}(2\pi\hbar)^{-3/2}, \\ &= \int dy[V'(x)(y+x) - V(y+x)]e^{-\frac{i}{\hbar}yj}(2\pi\hbar)^{-3/2}. \end{aligned} \tag{250}$$

15.4 Splitting integration of Hyperbolic PDEs

As an example of a hyperbolic transport equation, we consider the following PDE:

$$\begin{aligned} \frac{\partial f}{\partial t} &= -a\frac{\partial f}{\partial x} + D\frac{\partial^2 f}{\partial x^2}, \\ &= \left(-a\frac{\partial}{\partial x} + D\frac{\partial^2}{\partial x^2}\right) f, \end{aligned} \tag{251}$$

with initial condition $f(x,0) = f_0(x)$. The formal solution is a $f(x,t) = e^{\left(-a\frac{\partial}{\partial x} + D\frac{\partial^2}{\partial x^2}\right)t} f_0(x) = e^{-a\frac{\partial}{\partial x}t}\tilde{f}(x,t) = \tilde{f}(x-at,t)$, with $\tilde{f}(x,t) = e^{D\frac{\partial^2}{\partial x^2}t} f_0(x)$ the solution of the equation,

$$\frac{\partial\tilde{f}}{\partial t} = D\frac{\partial^2\tilde{f}}{\partial x^2}. \tag{252}$$

The solution is $\tilde{f}(x,t) = \int dy f_0(y)(4\pi t)^{-1/2}e^{-\xi(x-y)^2/t}$, since $\frac{\partial\tilde{f}}{\partial t} = \int dy f_0(y)(4\pi t)^{-1/2}e^{-\xi(x-y)^2/t}(\xi(x-y)^2/t^2 - 1/(2t))$ and $D\frac{\partial^2\tilde{f}}{\partial x^2} = \int dy f_0(y)(4\pi t)^{-1/2}e^{-\xi(x-y)^2/t}D((-2\xi(x-y)/t)^2 - 2\xi/t)$, with $D\xi = 1$. Therefore, the solution can be written, as follows: $\tilde{f}(x,t) = \int dy f_0(y)(4\pi t)^{-1/2}e^{-(x-y)^2/(Dt)} = \int dy f_0(x-y)(4\pi t)^{-1/2}e^{-y^2/(Dt)}$. Therefore, the solution of Eq. (251) is: $f(x,t) = \int dy f_0(x - at - y)(4\pi t)^{-1/2}e^{-y^2/(Dt)}$.

Analogously, we can integrate Eq. (249), as follows:

$$\begin{aligned} \frac{d\rho_H}{dt} &= \frac{i}{\hbar}\frac{\hbar^2}{2m}\frac{\partial^2\rho_H}{\partial x^2} - \frac{i}{\hbar}[V(x) - V'(x)x]\rho_H - \frac{i}{\hbar}\int dj\bar{J}(x,j)\rho_H(x,p+j), \\ &= \frac{i}{\hbar}\frac{\hbar^2}{2m}\frac{\partial^2\rho_H}{\partial x^2} - \frac{i}{\hbar}f(x,p)\rho_H, \end{aligned} \tag{253}$$

with

$$f(x,p) = V(x) - V'(x)x + \int dj \bar{J}(x,j) \frac{\rho_H(x,p+j)}{\rho_H(x,p)}, \tag{254}$$

The formal solution is $\rho_H(x,p,\tau) = e^{\frac{i}{\hbar}\left(\hbar\mathcal{L} + D\frac{\partial^2}{\partial x^2}\right)\tau}\tilde{\rho}_H$, where $D = \frac{\hbar^2}{2m}$ and $\tilde{\rho}_H = e^{-\frac{i}{\hbar}f(x,p)\tau}\rho_H(x,p,0)$, while $i\mathcal{L}$ is the classical Liouvillian $i\mathcal{L} = \frac{p}{m}\frac{\partial}{\partial x} - \frac{\partial V}{\partial x}\frac{\partial}{\partial p}$.

The solution is $\rho(x_\tau, p_\tau, \tau) = \int dy \tilde{\rho}_H(y, p_0, 0)(4\pi t)^{-1/2} e^{\frac{i}{\hbar}2m(x_0-y)^2/t}$, with x_0 and p_0 the coordinates and momenta obtained by backward classical propagation of x_τ and p_τ.

16 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:

$$c_f(t) = -\frac{i}{\hbar}\int_0^t dt' \langle f|\hat{W}(t')|i\rangle e^{\frac{i}{\hbar}(E_f - E_i)t'}, \tag{255}$$

Assuming that the perturbation involves a single frequency component, $\hat{W}(t') = \hat{A}e^{-iwt'}$, we obtain:

$$\begin{aligned} c_f(t) &= \langle f|\hat{A}|i\rangle \frac{[1 - e^{i(w_{fi} - w)t}]}{\hbar(w_{fi} - \omega)}, \\ &= -\frac{i}{\hbar} t \langle f|\hat{A}|i\rangle e^{i(w_{fi} - w)t/2} \frac{\sin[(w_{fi} - w)t/2]}{(w_{fi} - \omega)t/2}. \end{aligned} \tag{256}$$

Therefore, the probability of observing the system in the final state is

$$P_{fi}(t) = \frac{t^2}{\hbar^2}|\langle f|\hat{A}|i\rangle|^2 \frac{\sin^2[(w_{fi} - w)t/2]}{[(w_{fi} - \omega)t/2]^2}. \tag{257}$$

To compute the survival probability that the system remains in the initial state, we must add up the probability over all possible final states,

$$\begin{aligned} P(t) &= 1 - \frac{t^2}{\hbar^2}\sum_f |\langle f|\hat{A}|i\rangle|^2 \frac{\sin^2[(w_{fi} - w)t/2]}{[(w_{fi} - \omega)t/2]^2} \\ &= 1 - \frac{t^2}{\hbar^2}\int_{-\infty}^{\infty} dE_f \rho(E_f)|\langle f|\hat{A}|i\rangle|^2 \frac{\sin^2[(w_{fi} - w)t/2]}{[(w_{fi} - \omega)t/2]^2} \end{aligned} \tag{258}$$

If the very short time limit, $P(t) = exp(-\alpha t^2) \approx 1 - \alpha t^2 + \cdots$, where

$$\begin{aligned} \alpha &= \lim_{t\to 0} \frac{1}{\hbar^2}\int_{-\infty}^{\infty} dE_f |\langle f|\hat{A}|i\rangle|^2 \rho(E_f) \frac{\sin^2[(E_f - E_i - \hbar w)t/(2\hbar)]}{[(E_f - E_i - \hbar w)t/(2\hbar)]^2}, \\ &= \frac{1}{\hbar^2}\int_{-\infty}^{\infty} dE_f |\langle f|\hat{A}|i\rangle|^2 \rho(E_f), \end{aligned} \tag{259}$$

In the longer time limit, the kernel of Eq. (258) is approximated as the delta function to obtain:

$$\begin{aligned} P(t) &= 1 - \frac{t}{\hbar^2}\int_{-\infty}^{\infty} d(tE_f)\rho(E_f)|\langle f|\hat{A}|i\rangle|^2 \pi\delta((E_f t - (E_i + \hbar w)t)/(2\hbar)) \\ &= 1 - t\frac{2\pi}{\hbar}\int_{-\infty}^{\infty} d\xi \rho(\xi 2\hbar/t)|\langle f|\hat{A}|i\rangle|^2 \delta(\xi - (E_i + \hbar w)t/(2\hbar)) \\ &= 1 - t\frac{2\pi}{\hbar}\rho(E_i + \hbar w)|\langle E_i + \hbar w|\hat{A}|i\rangle|^2 \end{aligned} \tag{260}$$

so $P(t) = exp(-\Gamma t) \approx 1 - \Gamma t + \cdots$, where

$$\begin{aligned}\Gamma &= \frac{2\pi}{\hbar}\rho(E_i + \hbar w)|\langle E_i + \hbar w|\hat{A}|i\rangle|^2, \\ &= \frac{2\pi}{\hbar}\int_{-\infty}^{\infty} dE_f \rho(E_f)|\langle f|\hat{A}|i\rangle|^2 \delta(E_f - (E_i + \hbar w)),\end{aligned} \tag{261}$$

or as a discrete sum over states,

$$\Gamma = \frac{2\pi}{\hbar}\sum_f |\langle f|\hat{A}|i\rangle|^2 \delta(E_f - E_i - \hbar w), \tag{262}$$

which is known as Fermi's Golden rule.[2]

Substituting the delta function in Eq. (262) by its integral form, we obtain:

$$\begin{aligned}\Gamma_{fi} &= \frac{1}{\hbar^2}\int_{-\infty}^{\infty} dt \langle f|\hat{A}|i\rangle\langle i|\hat{A}|f\rangle e^{\frac{i}{\hbar}(E_f - E_i - \hbar w)t}, \\ &= \frac{1}{\hbar^2}\int_{-\infty}^{\infty} dt \langle f|e^{\frac{i}{\hbar}\hat{H}t}\hat{A}e^{-\frac{i}{\hbar}\hat{H}t}|i\rangle\langle i|\hat{A}|f\rangle e^{-iwt}, \\ &= \frac{1}{\hbar^2}\int_{-\infty}^{\infty} dt e^{-iwt} A_{fi}(t)A_{if}(0), \\ &= \frac{1}{\hbar^2}\int_{-\infty}^{\infty} dt e^{-iwt} A_{fi}(t)A_{if}(0).\end{aligned} \tag{265}$$

[2]Without introducing the approximation of the kernel of Eq. (258), we obtain:

$$\begin{aligned}P(t) &= 1 - \frac{t^2}{\hbar^2}\int_{-\infty}^{\infty} dE_f \rho(E_f)|\langle f|\hat{A}|i\rangle|^2 \frac{\sin^2[(w_{fi} - w)t/2]}{[(w_{fi} - \omega)t/2]^2} \\ &= 1 - t\frac{2}{\hbar}\int_{-\infty}^{\infty} dE_f \rho(E_f)\frac{t}{2\hbar}|\langle f|\hat{A}|i\rangle|^2 \frac{\sin^2((E_f - E_i - \hbar w)t/(2\hbar))}{((E_f - E_i - \hbar\omega)t/(2\hbar))^2} \\ &= 1 - t\frac{2}{\hbar}\int_{-\infty}^{\infty} d\xi \rho(\xi 2\hbar/t)|\langle f|\hat{A}|i\rangle|^2 \frac{\sin^2(\xi - (E_i + \hbar w)t/(2\hbar))}{(\xi - (E_i + \hbar\omega)t/(2\hbar))^2} \\ &= 1 - t\frac{2}{\hbar}\int_{-\infty}^{\infty} d\xi \rho(\xi 2\hbar/t)|\langle f|\hat{A}|i\rangle|^2 \frac{\sin^2(\xi)}{\xi^2}\end{aligned} \tag{263}$$

which gives, in the time-range when the decay is exponential (i.e., $P(t) = exp(-\Gamma t) \approx 1 - \Gamma t$),

$$\begin{aligned}\Gamma &= \frac{2}{\hbar}\int_{-\infty}^{\infty} d\xi \rho(\xi 2\hbar/t)|\langle f|\hat{A}|i\rangle|^2 \frac{\sin^2(\xi)}{\xi^2}, \\ &= \frac{2}{\hbar}\int_{-\infty}^{\infty} dE_f \rho(E_f)|\langle f|\hat{A}|i\rangle|^2 \frac{\sin^2((E_f - E_i - \hbar w)t/(2\hbar))}{(E_f - E_i - \hbar\omega)^2 t/(2\hbar)}.\end{aligned} \tag{264}$$

The equilibrium ensemble average is

$$\begin{aligned}\langle\Gamma_{fi}\rangle &= \frac{1}{\hbar^2}\int_{-\infty}^{\infty} dt e^{-iwt}\sum_{\alpha} p_\alpha\langle\alpha|A_{fi}(t)A_{if}(0)|\alpha\rangle,\\ &= \frac{1}{\hbar^2}\int_{-\infty}^{\infty} dt e^{-iwt}\langle A_{fi}(t)A_{if}(0)\rangle, \\ &= \frac{[1+e^{-\beta\hbar\omega}]^{-1}}{\hbar^2}\int_{-\infty}^{\infty} dt e^{-iwt}\langle[A_{fi}(t)A_{if}(0)+A_{fi}(0),A_{if}(t)]\rangle,\end{aligned} \tag{266}$$

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. (266), $\langle\Gamma_{fi}\rangle$ can be written as follows:

$$\langle\Gamma_{fi}\rangle = \frac{[1+e^{-\beta\hbar\omega}]^{-1}}{\hbar^2}\int_{-\infty}^{\infty} dt e^{-iwt}\langle[A_{fi}(t)A_{if}(0)+A_{fi}(0),A_{if}(t)]\rangle, \tag{267}$$

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. (267) 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 step-by-step derivation of the last line of Eq. (267) 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 \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 \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{268}
$$

$$
\begin{aligned}
\langle \Gamma_{fi} \rangle &= \frac{[1+e^{-\beta\hbar\omega}]^{-1}}{\hbar^2} \int_{-\infty}^{\infty} dt e^{-iwt} \left[\sum_{\alpha,\gamma} p_\alpha \langle \alpha | A_{fi}(t) | \gamma \rangle \langle \gamma | A_{if}(0) | \alpha \rangle + \sum_{\alpha,\gamma} p_\gamma \langle \gamma | A_{if}(0) | \alpha \rangle \langle \alpha | A_{fi}(t) | \gamma \rangle \right], \\
&= \frac{[1+e^{-\beta\hbar\omega}]^{-1}}{\hbar^2} \int_{-\infty}^{\infty} dt e^{-iwt} \sum_{\alpha,\gamma} p_\alpha \left[\langle \alpha | A_{fi}(t) | \gamma \rangle \langle \gamma | A_{if}(0) | \alpha \rangle + \langle \alpha | A_{if}(0) | \gamma \rangle \langle \gamma | A_{fi}(t) | \alpha \rangle \right], \\
&= \frac{[1+e^{-\beta\hbar\omega}]^{-1}}{\hbar^2} \int_{-\infty}^{\infty} dt e^{-iwt} \sum_{\alpha} p_\alpha \left[\langle \alpha | A_{fi}(t) A_{if}(0) + A_{if}(0) A_{fi}(t) | \alpha \rangle \right], \\
&= \frac{[1+e^{-\beta\hbar\omega}]^{-1}}{\hbar^2} \int_{-\infty}^{\infty} dt e^{-iwt} \langle [A_{fi}(t) A_{if}(0) + A_{if}(0) A_{fi}(t)] \rangle, \\
&= \frac{[1+e^{-\beta\hbar\omega_{fi}}]^{-1}}{\hbar^2} \int_{-\infty}^{\infty} dt e^{-iwt} \langle [A_{fi}(t) A_{if}(0) + A_{if}(0) A_{fi}(t)] \rangle,
\end{aligned}
\tag{269}
$$

17 WKB Approximation

The goal of this section is to introduce the WKB approximation proposed by Wentzel, Kramers and Brillouin to solve the time-independent Schrödinger equation:

$$\begin{aligned} -\frac{\hbar^2}{2m}\frac{d^2\psi}{dx^2} + V(x)\psi &= E\psi, \\ \frac{d^2\psi}{dx^2} + \frac{2m}{\hbar^2}(E - V(x))\psi &= 0, \\ \frac{d^2\psi}{dx^2} + k(x)^2\psi &= 0, \end{aligned} \tag{270}$$

or

$$\frac{d^2\psi}{dx^2} + k(x)^2\psi = 0, \tag{271}$$

with $k(x) = \sqrt{\frac{2m}{\hbar^2}(E - V)}$ when $E > V$ and $k(x) = i\sqrt{\frac{2m}{\hbar^2}(V - E)}$ when $V > E$.

When $k(x) = k(x_0) = k_0$, the solution of Eq. (271) is

$$\psi(x) = \psi(x_0)e^{\pm ik_0(x-x_0)}, \tag{272}$$

which can also be written, as follows:

$$\psi(x) = \psi(x_0)e^{\pm i\int_{x_0}^{x} dr k(r)}. \tag{273}$$

Substituting Eq. (273) into Eq. (271), we obtain:

$$\begin{aligned} \frac{d}{dx}(\pm ik\psi) + k^2\psi &= 0, \\ (-k^2 \pm ik')\psi(x) + k^2\psi &= 0, \end{aligned} \tag{274}$$

showing that the solution introduced by Eq. (273) that is valid not only when $k = k_0$ but also when $|k'| \ll |k^2|$. That solution (Eq. (273)) is the so-called WKB approximation.

More generally, the WKB approximation can be written, as follows:

$$\psi(x,t) = \psi(x_0)e^{\frac{i}{\hbar}S(x)}e^{-\frac{i}{\hbar}Et}, \tag{275}$$

with $S(x) = \int_{x_0}^{x} dr\hbar k(r)$. Substituting Eq. (275) into the time-independent Schrödinger equation, we obtain:

$$\begin{aligned} -\frac{\hbar^2}{2m}(-\frac{1}{\hbar^2}S'^2 + \frac{i}{\hbar}S'') + (V(x) - E) &= 0, \\ (-\frac{1}{\hbar^2}S'^2 + \frac{i}{\hbar}S'') + \frac{2m}{\hbar^2}(E - V(x)) &= 0, \\ -S'^2 + i\hbar S'' + 2m(E - V(x)) &= 0, \\ -S'^2 + i\hbar S'' + \hbar^2k^2 &= 0. \end{aligned} \tag{276}$$

Considering the expansion $S = S_0 + \hbar S_1 + \hbar^2 S_2 + \cdots$ and substituting into Eq. (276), we obtain:

$$-S_0'^2 - \hbar^2 S_1'^2 - 2\hbar S_0' S_1' + \cdots + i\hbar S_0'' + i\hbar^2 S_1'' + 2m(E-V) = 0, \tag{277}$$

and making equal terms of equal power of $\hbar$, we obtain: $S_0' = \sqrt{2m(E-V)}$ or $S_0(x) = \int^x dx' \hbar k(x')$ and $S_1' = \frac{i}{2} dln S_0'$, or $S_1(x) = \frac{i}{2} ln S_0' = \frac{i}{2} ln \hbar k(x)$.

Therefore, the WKB approximation to second order in $\hbar^2$ is:

$$\begin{aligned} \psi(x,t) &= \psi(x_0) e^{\frac{i}{\hbar}\int_{x_0}^x dx' \hbar k(x') + \frac{i}{\hbar}\hbar\frac{i}{2} ln \hbar k(x)} e^{-\frac{i}{\hbar}Et}, \\ &= \psi(x_0) e^{\frac{i}{\hbar}\int_{x_0}^x dx' \hbar k(x') + ln\frac{1}{\sqrt{\hbar k(x)}}} e^{-\frac{i}{\hbar}Et}, \\ &= \frac{\psi(x_0)}{\sqrt{\hbar k(x)}} e^{i\int_{x_0}^x dx' k(x')} e^{-\frac{i}{\hbar}Et}. \end{aligned} \tag{278}$$

17.1 Connection Formula

We now consider the WKB solution on both sides of a classical turning point x_c, where $V(x_c) = E$. We expand the potential around x_c, as follows: $V(x) = E + V'(x_c)(x - x_c)$, for example with $V'(x_c) > 0$ satisfying the Schrödinger equation:

$$\begin{aligned} \frac{d^2\psi}{dx^2} + k^2\psi &= 0, \\ \frac{d^2\psi}{dx^2} - (x - x_c)\alpha^3\psi &= 0, \end{aligned} \tag{279}$$

where $k^2 = \frac{2m}{\hbar^2}(E-V)$ and $\alpha^3 = \frac{2m}{\hbar^2}V'(x_c) > 0$.

The WKB solution of Eq. (279) is

$$\psi(x) = \begin{cases} c_1 e^{i\int_x^{x_c} drk(r)} + c_2 e^{-i\int_x^{x_c} drk(r)}, & \text{for } x < x_c \\ c_3 e^{-\int_{x_c}^x drk(r)}, & \text{for } x > x_c. \end{cases} \tag{280}$$

with $k(x)^2 = \alpha^3(x_c - x)$, or

$$\psi(x) = \begin{cases} c_1 e^{i\int_x^{x_c} dr\alpha^{3/2}(x_c-r)^{1/2}} + c_2 e^{-i\int_x^{x_c} dr\alpha^{3/2}(x_c-r)^{1/2}}, & \text{for } x < x_c \\ c_3 e^{-\int_{x_c}^x dr\alpha^{3/2}(r-x_c)^{1/2}}, & \text{for } x > x_c, \end{cases} \tag{281}$$

giving

$$\psi(z) = \begin{cases} c_1 e^{i\frac{2}{3}(-z)^{3/2}} + c_2 e^{-i\frac{2}{3}(-z)^{3/2}}, & \text{for } z < 0 \\ c_3 e^{-\frac{2}{3}z^{3/2}}, & \text{for } z > 0, \end{cases} \tag{282}$$

with $z = \alpha(x - x_c)$. Introducing this change of variables, with $dz/dx = \alpha$, we find that Eq. (279) can be written, as follows:

$$\left(\frac{d^2}{dz^2} - z\right)\psi(z) = 0, \tag{283}$$

which is the Airy equation (or Stokes equation) whose solution is the Airy function,

$$\psi(z) = Ai(z), \tag{284}$$

defined, as follows:

$$\begin{aligned} Ai(z) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} dt \;\; e^{i(zt+t^3/3)}, \\ &= \frac{1}{2\pi}\int_{0}^{\infty} dt \;\; cos(zt+t^3/3), \end{aligned} \tag{285}$$

with asymptotic expansions

$$Ai(z) = \begin{cases} \frac{1}{2i\sqrt{\pi}}(-z)^{-1/4}\left\{e^{i\frac{\pi}{4}+i\frac{2}{3}(-z)^{3/2}} - e^{-i\frac{\pi}{4}-i\frac{2}{3}(-z)^{3/2}}\right\}, \;\; \text{when} \;\; z \to -\infty \\ \frac{1}{2\sqrt{\pi}}z^{-1/4}e^{-\frac{2}{3}z^{3/2}}, \;\; \text{when} \;\; z \to \infty. \end{cases} \tag{286}$$

Comparing the WKB solution, introduced by Eq. (282), with the asymptotic expansions of the exact solution, introduced by Eq. (286), we see that the coefficients c_1, c_2 and c_3 can be defined, as follows:

$$\begin{aligned} c_1 &= \frac{\alpha^{-1/4}}{2i\sqrt{\pi}}(x_c - x)^{-1/4}e^{i\frac{\pi}{4}}, \\ c_2 &= \frac{\alpha^{-1/4}}{2i\sqrt{\pi}}(x_c - x)^{-1/4}e^{i\frac{\pi}{4}}, \\ c_3 &= \frac{\alpha^{-1/4}}{2\sqrt{\pi}}(x - x_c)^{-1/4}, \end{aligned} \tag{287}$$

giving

$$\psi(x) = \begin{cases} \frac{1}{\sqrt{\pi}(x_c - x)^{1/4}\alpha^{1/4}}\sin\left(\frac{\pi}{4} + \frac{2}{3}(x_c - x)^{3/2}\alpha^{3/2}\right), \;\; \text{when} \;\; x < x_c \\ \frac{1}{2\sqrt{\pi}(x - x_c)^{1/4}\alpha^{1/4}}e^{-\frac{2}{3}(x-x_c)^{3/2}\alpha^{3/2}}, \;\; \text{when} \;\; x > x_c, \end{cases} \tag{288}$$

or

$$\psi(x) = \begin{cases} \frac{1}{\sqrt{\pi}(x_c - x)^{1/4}\alpha^{1/4}}\cos\left(-\frac{\pi}{4} + \frac{2}{3}(x_c - x)^{3/2}\alpha^{3/2}\right), \;\; \text{when} \;\; x < x_c, \;\; (i.e., V(x) < E), \\ \frac{1}{2\sqrt{\pi}(x - x_c)^{1/4}\alpha^{1/4}}e^{-\frac{2}{3}(x-x_c)^{3/2}\alpha^{3/2}} \;\; \text{when} \;\; x > x_c \;\; (i.e., V(x) > E). \end{cases} \tag{289}$$

which can also be written in terms of $k(x) = \sqrt{2m(E-V)/\hbar^2} = \sqrt{\alpha^3(x_c - x)}$, as the WKB solution:

$$\psi(x) = \begin{cases} \dfrac{\sqrt{\alpha}}{\sqrt{\pi k(x)}} \cos\left(-\dfrac{\pi}{4} + \displaystyle\int_x^{x_c} dx' k(x')\right), & \text{when } V(x) < E, \\ \dfrac{\sqrt{\alpha}}{2\sqrt{\pi |k(x)|}} e^{-\int_{x_c}^{x} |k(x')| dx'} & \text{when } V(x) > E. \end{cases} \tag{290}$$

The WKB solution for $V(x) > E$ can be used to estimate tunneling probabilities through potential energy barriers (*e.g.*, $V(x) > E$ when $a < x < b$), as follows:

$$\begin{aligned} P &= \frac{|\psi(b)|^2}{|\psi(a)|^2}, \\ &= e^{-2\int_a^b |k(x')| dx'}, \end{aligned} \tag{291}$$

which provide a description of tunneling currents, as shown in Sec. 20.

17.2 Normalization

The bound states of a potential well with $V(x) < E$ in the classically allowed region with left and right classical turning points x_l and x_r, respectively, can be approximated according to Eq. (290), as follows:

$$\psi_n(x) = \frac{N}{\sqrt{k_n(x)}} \cos\left(-\frac{\pi}{4} + \int_x^{x_r} dx' k_n(x')\right), \tag{292}$$

where $N = \sqrt{2m\omega/\hbar\pi}$ is the normalization constant.

To obtain the normalization constant N, we first note that the solution must be the same when written in terms of either of the two turning points,

$$\begin{aligned} \psi_n(x) &= \frac{N}{\sqrt{k_n(x)}} \cos\left(-\frac{\pi}{4} + \int_x^{x_r} dx' k_n(x')\right), \\ &= \frac{N}{\sqrt{k_n(x)}} \cos\left(-\frac{\pi}{4} + \int_{x_l}^{x} dx' k_n(x')\right), \end{aligned} \tag{293}$$

so

$$\begin{aligned}
\psi_n(x) &= \frac{N}{\sqrt{k_n(x)}}\cos\left(-\frac{\pi}{4}+\int_{x_l}^{x} dx' k_n(x')\right), \\
&= \frac{N}{\sqrt{k_n(x)}}\cos\left(-\frac{\pi}{4}-\int_{x}^{x_r} dx' k_n(x')+\int_{x_l}^{x_r} dx' k_n(x')\right), \\
&= \frac{N}{\sqrt{k_n(x)}}\left\{\cos\left(-\frac{\pi}{4}-\int_{x}^{x_r} dx' k_n(x')\right)\cos\left(\int_{x_l}^{x_r} dx' k_n(x')\right)\right. \\
&\left.-\sin\left(-\frac{\pi}{4}-\int_{x}^{x_r} dx' k_n(x')\right)\sin\left(\int_{x_l}^{x_r} dx' k_n(x')\right)\right\}, \\
&= \frac{N}{\sqrt{k_n(x)}}\left\{\cos\left(-\frac{\pi}{4}-\int_{x}^{x_r} dx' k_n(x')\right)\cos\left(\int_{x_l}^{x_r} dx' k_n(x')\right)\right. \\
&\left.+\cos\left(-\frac{\pi}{4}+\int_{x}^{x_r} dx' k_n(x')\right)\sin\left(\int_{x_l}^{x_r} dx' k_n(x')\right)\right\}.
\end{aligned} \tag{294}$$

To make the last row of Eq. (294) equal to the first row of Eq. (293), we must have:

$$\int_{x_l}^{x_r} dx' \hbar k_n(x') = (n+\frac{1}{2})\hbar\pi, \tag{295}$$

introducing the energy quantization of bound states (*e.g.*, $E_n = \hbar\omega(\frac{1}{2}+n)$ for the harmonic oscillator), as shown in Sec. 17.4.

Next, we obtain the normalization constant N of ψ_n, as introduced in Eq. (292), as follows:

$$\begin{aligned}
1 &= \langle\psi_n|\psi_n\rangle, \\
&= N^2\int dx\frac{1}{4k(x)}\left\{e^{-i\frac{\pi}{4}+i\int_{x_l}^{x} kdx'}+e^{i\frac{\pi}{4}-i\int_{x_l}^{x} kdx'}\right\}\left\{e^{i\frac{\pi}{4}-i\int_{x_l}^{x} kdx'}+e^{-i\frac{\pi}{4}+i\int_{x_l}^{x} kdx'}\right\}, \\
&= N^2\int dx\frac{1}{2k(x)}+N^2\int dx\frac{1}{4k(x)}\left\{-e^{i2\int_{x_l}^{x} kdx'}+e^{-i2\int_{x_l}^{x} kdx'}\right\}, \\
&= N^2\int_{x_l}^{x_r} dx\frac{\hbar}{2\sqrt{2m(E_n-m\omega^2x^2/2)}}+N^2\int dx\frac{1}{4k(x)}\left\{-e^{i2\int^{x} kdx'}+e^{-i2\int^{x} kdx'}\right\}, \\
&= N^2\int_{m\omega x_l/\sqrt{2mE_n}}^{m\omega x_r/\sqrt{2mE_n}} dr\frac{\hbar}{2\sqrt{2mE_n}\sqrt{(1-r^2)}}+N^2\int_{x_l}^{x_r} dx\frac{1}{4k(x)}\left\{-e^{i2\int^{x} kdx'}+e^{-i2\int^{x} kdx'}\right\}, \\
&= \frac{N^2}{2\sqrt{2mE_n}}\int_{-1}^{1} dr\frac{\hbar}{\sqrt{(1-r^2)}}+N^2\int_{x_l}^{x_r} dx\frac{1}{4k(x)}\left\{-e^{i2\int^{x} kdx'}+e^{-i2\int^{x} kdx'}\right\}, \\
&= \frac{\hbar\pi N^2}{2\sqrt{2mE_n}} = \frac{\hbar\pi N^2}{2m\omega},
\end{aligned} \tag{296}$$

giving $N = \sqrt{2m\omega/\hbar\pi}$, where we have introduced the change of variables $r = m\omega x/\sqrt{2mE_n} = cos(\theta)$, with $\int_{-1}^{1} dcos(\theta)/sin(\theta) = \int_0^{\pi} d\theta = \pi$, and the turning points $x_r = \sqrt{2E_nm}/(m\omega)$ and

$x_l = -\sqrt{2E_n m}/(m\omega)$. We note that the integral in the second term of Eq. (296) vanishes when $E_n = \hbar\omega(\frac{1}{2}+n) = \frac{1}{2}m\omega^2 x_l^2$.

17.3 Overlaps

The WKB bound states for the harmonic oscillator are defined, as follows:

$$\psi_n(x) = \sqrt{\frac{2m\omega}{\pi\hbar k_n(x)}}\cos\left(-\frac{\pi}{4} + \hbar^{-1}\int_{x_l}^{x} dx'\hbar k_n(x')\right), \tag{297}$$

with $\hbar k_n = \sqrt{2m\hbar\omega(\frac{1}{2}+n) - m^2\omega^2x^2}$, which are very good approximations to the actual eigenstates, and can be used to approximate the overlaps $\langle\psi_i|\psi_f\rangle = \int dx\psi_i^*(x)\psi_f(x-x_{if})$. introduced by Eq. (309), as follows:

$$\begin{aligned}\langle\psi_i|\psi_f\rangle = \int dx \frac{m\omega}{2\pi\hbar\sqrt{k_i(x)k_f(x-x_{if})}} &\left\{e^{\frac{i\pi}{4}-\frac{i}{\hbar}\int_{x_l}^{x}dx'\hbar k_i(x')} + e^{-\frac{i\pi}{4}+\frac{i}{\hbar}\int_{x_l}^{x}dx'\hbar k_i(x')}\right\} \\ \times &\left\{e^{-\frac{i\pi}{4}+\frac{i}{\hbar}\int_{x_l}^{x}dx'\hbar k_f(x'-x_{if})} + e^{\frac{i\pi}{4}-\frac{i}{\hbar}\int_{x_l}^{x}dx'\hbar k_f(x'-x_{if})}\right\},\end{aligned} \tag{298}$$

Neglecting highly oscillatory terms, we obtain:

$$\begin{aligned}\langle\psi_i|\psi_f\rangle &= \int dx \frac{m\omega}{2\pi\hbar\sqrt{k_i(x)k_f(x-x_{if})}}\left\{e^{-\frac{i}{\hbar}\int_{x_l}^{x}dx'\hbar[k_i(x')-k_f(x'-x_{if})]} + c.c.\right\} \\ &= \int dx \frac{m\omega}{2\pi\hbar\sqrt{k_i(x)k_f(x-x_{if})}}\left\{e^{-\frac{i}{\hbar}\phi(x)} + c.c.\right\}\end{aligned} \tag{299}$$

with $\phi(x) = \int_{x_l}^{x} dx'\hbar[k_i(x') - k_f(x'-x_{if})]$. To evaluate the integral by the stationary phase approximation method, we find the point x_* such that $\phi'(x_*) = 0$ and we expand the phase to second order relative to that point, as follows: $\phi(x) \approx \phi(x_*) + \frac{1}{2}\phi''(x_*)(x-x_*)^2$.

Note that $\hbar k_i(x_*) = \hbar k_f(x_* - x_{if})$, or $\sqrt{2mE_i - m^2\omega^2x_*^2} = \sqrt{2mE_f - m^2\omega^2(x_*-x_{if})^2} = \hbar k_*$. Furthermore, $\phi''(x_*) = \hbar k_i'(x_*) - \hbar k_f'(x_* - x_{if}) = \frac{1}{2}(2mE_i - m^2\omega^2x_*^2)^{-1/2}(-2m^2\omega^2x_*) - \frac{1}{2}(2mE_f - m^2\omega^2(x_*-x_{if})^2)^{-1/2}(-2m^2\omega^2(x_*-x_{if})) = \frac{1}{2}(\hbar k_*)^{-1}(-2m^2\omega^2x_* + 2m^2\omega^2(x_* - x_{if})) = -(\hbar k_*)^{-1}m^2\omega^2x_{if}$.

Evaluating the resulting Gaussian integral, given by the stationary phase approximation, we obtain:

$$\begin{aligned}\langle\psi_i|\psi_f\rangle &= I_* + c.c., \\ |\langle\psi_i|\psi_f\rangle|^2 &= I_*I_*^* + I_*I_* + I_*^*I_* + I_*^*I_*^*, \\ &\approx 2I_*I_*^*,\end{aligned} \tag{300}$$

where

$$
\begin{aligned}
I_* &= \frac{m\omega}{2\pi\hbar k_*} e^{-\frac{i}{\hbar}\phi(x_*)} \int dx e^{-\frac{i}{2\hbar}\phi''(x_*)(x-x_*)^2}, \\
&= \frac{m\omega}{2\pi\hbar k_*} e^{-\frac{i}{\hbar}\phi(x_*)} \sqrt{\frac{2\hbar\pi}{-i\phi''(x_*)}}, \\
&= \frac{m\omega}{2\pi\hbar k_*} e^{-\frac{i}{\hbar}\phi(x_*)} \sqrt{\frac{2\hbar\pi}{i(\hbar k_*)^{-1} m^2\omega^2 x_{if}}}, \\
&= e^{-\frac{i}{\hbar}\phi(x_*)} \sqrt{\frac{1}{i2\pi k_* x_{if}}},
\end{aligned}
\tag{301}
$$

so

$$
|\langle\psi_i|\psi_f\rangle|^2 = 2I_* I_*^* \approx \frac{1}{\pi k_* x_{if}}. \tag{302}
$$

The overlap introduced by Eq. (302) can be expressed in terms of the vibrational quantum numbers n_i and n_f by considering that

$$
\begin{aligned}
\frac{\hbar^2 k_*^2}{2m} &= \hbar\omega(\frac{1}{2} + n_i) - \frac{1}{2} m\omega^2 x_*^2, \\
&= \hbar\omega(\frac{1}{2} + \tilde{n}_f) - \frac{1}{2} m\omega^2 (x_* - x_{if})^2 - \Delta E, \\
&= \hbar\omega(\frac{1}{2} + n_f) - \frac{1}{2} m\omega^2 (x_* - x_{if})^2,
\end{aligned}
\tag{303}
$$

where $n_f = \tilde{n}_f - \Delta E/(\hbar\omega)$. Solving for x_*, we obtain:

$$
x_* = \frac{2\hbar\omega(n_i - n_f) + m\omega^2 x_{if}^2}{m\omega^2 2x_{if}}, \tag{304}
$$

and substituting into Eq. (303), using $n_\lambda\hbar = \frac{1}{2} m\omega x_{if}^2$, we obtain:

$$
\begin{aligned}
\frac{\hbar^2 k_*^2}{2m} &= \hbar\omega(\frac{1}{2} + n_i) - \frac{1}{2} m\omega^2 (\frac{2\hbar\omega(n_i - n_f) + m\omega^2 x_{if}^2}{m\omega^2 2x_{if}})^2, \\
x_{if}^2\hbar^2 k_*^2 &= m x_{if}^2 \hbar\omega + 2m x_{if}^2 \hbar\omega n_i - (\hbar^2(n_i - n_f)^2 + \frac{m^2\omega^2}{4} x_{if}^4 + \hbar(n_i - n_f) m\omega x_{if}^2), \\
&= m x_{if}^2 \hbar\omega + m x_{if}^2 \hbar\omega(n_i + n_f) - (\hbar^2(n_i - n_f)^2 + \frac{m^2\omega^2}{4} x_{if}^4), \\
&= 2n_\lambda\hbar^2 + 2n_\lambda\hbar^2(n_i + n_f) - (\hbar^2(n_i - n_f)^2 + n_\lambda^2\hbar^2), \\
x_{if}^2 k_*^2 &= 2n_\lambda + 2n_\lambda(n_i + n_f) - (n_i - n_f)^2 - n_\lambda^2, \\
&= 2n_\lambda + 2n_\lambda n_i + 2n_\lambda n_f - n_i^2 - n_f^2 + 2n_i n_f - n_\lambda^2, \\
&\approx (n_f + n_\lambda + 2\sqrt{n_f n_\lambda} - n_i)(n_i - n_f - n_\lambda + 2\sqrt{n_f n_\lambda}),
\end{aligned}
\tag{305}
$$

where in the last line we have neglected n_λ as compared to the other terms. Substituting Eq. (305) into Eq. (494), we obtain:

$$|\langle\psi_i|\psi_f\rangle|^2 \approx \frac{1}{\pi\sqrt{(n_{max}-n_i)(n_i-n_{min})}}, \tag{306}$$

with $n_{max} = n_f + n_\lambda + 2\sqrt{n_f n_\lambda}$ and $n_{min} = n_f + n_\lambda - 2\sqrt{n_f n_\lambda}$.

17.4 Energy Quantization

In this section, we show that the WKB approximation predicts that the energy levels of the harmonic oscillator $V(x) = \frac{1}{2}m\omega^2$ are discretized, as follows: $E_n = \hbar\omega(\frac{1}{2}+n)$. According to Eq. (295),

$$\begin{aligned}
\int_{x_l}^{x_r} dx'\hbar k_n(x') &= (n+\frac{1}{2})\hbar\pi, \\
\int_{x_l}^{x_r} dx'\sqrt{2mE_n - m^2\omega^2 x'^2} &= (n+\frac{1}{2})\hbar\pi, \\
\oint p(x)dx &= (n+\frac{1}{2})h,
\end{aligned} \tag{307}$$

where the contour integral in the last row corresponds to a complete loop in phase-space integrating the classical momentum $p(x)$ from $x = x_l$ to $x = x_r$ and back to $x = x_l$.

The integer numbers $n = 0, 1, 2, \cdots$ introduce the energy quantization, since they define the possible discrete values of E_n associated with the bound states. To show that, we Introduce the change of variables $r = m\omega x'/\sqrt{2mE_n} = cos(\theta)$ and we obtain:

$$\begin{aligned}
\int_{-1}^{1} dcos(\theta)\frac{2}{\omega}E_n\sqrt{(1-cos^2(\theta))} &= (n+\frac{1}{2})\hbar\pi, \\
\int_0^{\pi} d\theta sin^2(\theta)\frac{2}{\omega}E_n &= (n+\frac{1}{2})\hbar\pi, \\
\frac{\pi}{2}\frac{2}{\omega}E_n &= (n+\frac{1}{2})\hbar\pi, \\
E_n &= (n+\frac{1}{2})\hbar\omega.
\end{aligned} \tag{308}$$

17.5 Computational Problem WKB Approximation

(a) Write a program to obtain the WKB approximation of the eigenstate of the harmonic oscillator with $\nu = 20$ and compare it to the exact solution.
(b) Write a program to compare the WKB approximation of the vibrational overlap factors for two displaced harmonic oscillators, as described by Eqs. (302) and (306) and compare them to the exact numerical values of the squared overlaps.
Solution:

The link item (a) and link item (b) provide Matlab solutions to items (a) and (b), respectively.

18 Electron Transfer: Marcus Theory

This subsection shows that the WKB approximation of the bound states of a Harmonic oscillator can be used to estimate the non-radiative transitions between two weakly coupled harmonic oscillators, as described by the Golden rule of first order time-dependent perturbation theory:

$$\begin{aligned}\Gamma_i &\approx \frac{2\pi}{\hbar}\sum_f |\langle f|\hat{A}|i\rangle|^2\delta(E_f-\hbar(w_i+w)),\\ &\approx \frac{2\pi}{\hbar}|H_{if}|^2\sum_f |\langle\psi_f|\psi_i\rangle|^2\delta(E_f-E_i-\Delta E),\end{aligned} \tag{309}$$

where $\langle\psi_f|\psi_i\rangle$ are the overlaps of harmonic vibrational states, $H_{if}=\langle e_f|\hat{A}|e_i\rangle$ is the coupling between electronic states, assumed to be independent of vibrational coordinates, and $\Delta E=\hbar\omega$ is the change in vibrational energy.

Substituting the WKB approximation of $|\langle\psi_f|\psi_i\rangle|^2=(\pi k_* x_{if})^{-1}$, introduced by Eq. (302), into Eq. (309) using an approximate density of states of $1/(\hbar\omega)$ when $E_i, E_f > E^*$, with $\hbar k^* = \sqrt{2m(E_i-E^*)}$ and $x_{if}=\sqrt{2E_\lambda/(m\omega^2)}$, we obtain:

$$\begin{aligned}\Gamma_i &\approx \frac{2\pi}{\hbar}|H_{if}|^2\frac{1}{\hbar\omega\pi k_* x_{if}},\\ &\approx \frac{1}{\hbar}|H_{if}|^2\frac{1}{\sqrt{(E_i-E^*)}\sqrt{E_\lambda}}.\end{aligned} \tag{310}$$

Computing the thermal average over all initial states, we obtain the overall rate, as follows:

$$\begin{aligned}\Gamma &\approx |H_{if}|^2\frac{1}{\hbar\sqrt{E_\lambda}}\frac{\int dE_i exp(-\beta E_i)\frac{1}{\sqrt{(E_i-E^*)}}}{\int dE_i exp(-\beta E_i)}\\ &\approx |H_{if}|^2\frac{1}{\hbar\sqrt{E_\lambda}}exp(-\beta E^*)\frac{\int dE exp(-\beta E)\frac{1}{\sqrt{E}}}{\int dE exp(-\beta E)}\\ &\approx |H_{if}|^2\frac{1}{\hbar\sqrt{E_\lambda}}exp(-\beta E^*)\frac{\int_0^\infty dp 2p exp(-\beta p^2)\frac{1}{p}}{\int dE exp(-\beta E)}\\ &\approx |H_{if}|^2\frac{1}{\hbar\sqrt{E_\lambda}}exp(-\beta E^*)\frac{\int_{-\infty}^\infty dp exp(-\beta p^2)}{\int dE exp(-\beta E)}\\ &\approx |H_{if}|^2\frac{1}{\hbar\sqrt{E_\lambda}}exp(-\beta E^*)\frac{\sqrt{\pi/\beta}}{\beta^{-1}}\\ &\approx |H_{if}|^2\sqrt{\frac{\pi\beta}{\hbar^2 E_\lambda}}exp(-\beta E^*)\\ &\approx |H_{if}|^2\frac{\sqrt{2\pi\beta/m}}{\hbar\omega x_{if}}exp(-\beta E^*)\end{aligned} \tag{311}$$

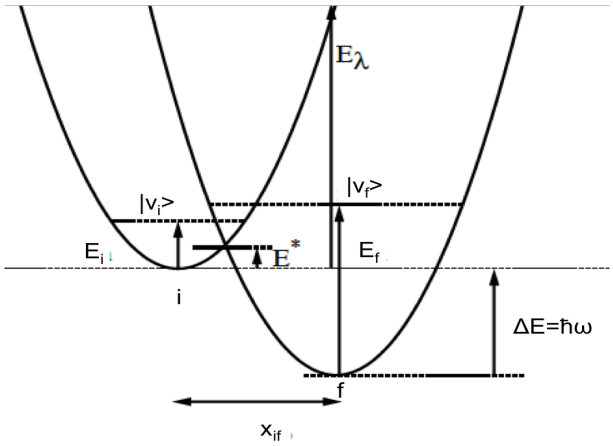


Figure 18 shows a schematic description of the energy diagram, assuming that the potential energy surfaces along the vibronic coordinates are parabolas displaced in $x_{fi} = x_f - x_i$ with frequency ω, we have $V_i = 1/2m\omega^2(x - x_i)^2$ and $V_f = 1/2m\omega^2(x - x_f)^2 - \Delta E$, which cross at x^* with energy

$$E^* = (E_\lambda - \Delta E)^2/(4E_\lambda). \tag{312}$$

To derive Eq. (312), 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: $E_\lambda = m\omega^2(x^* - x_i)x_{fi} + \Delta E = m\omega^2\sqrt{2E^*/(m\omega^2)}\sqrt{2E_\lambda/(m\omega^2)} + \Delta E$.

19 Landau-Zener Equation

The goal of this section is to derive the famous Landau-Zener formula,

$$P_{LZ} = e^{-\frac{\pi\Delta^2}{2\hbar\epsilon}}, \tag{313}$$

for the probability that a system, initially prepared on a diabatic state, undergoes a nonadiabatic transition and is found on the other diabatic state at an asymptotically long time when described by the following time-dependent model Hamiltonian:

$$\hat{H} = \frac{\epsilon}{2}t\left(|\phi_1\rangle\langle\phi_1| - |\phi_2\rangle\langle\phi_2|\right) + \frac{\Delta}{2}(|\phi_1\rangle\langle\phi_2| + |\phi_2\rangle\langle\phi_1|).$$

The Landau-Zener formula was published by Clarence Zener in 1932, as the exact solution to a one-dimensional semi-classsical model for nonadiabatic transitions [Zener, C. *Proc. R. Soc. London A* 1932, **137**, 696] of a model where nuclear motion was treated classically with constant velocity v. Therefore, the parameter $\epsilon = vF_{12}$ corresponds to the velocity v times the difference of the forces on the two diabatic surfaces F_{12}. If the surfaces are parallel to each other, or the system enters the electronic transition with low speed, the transition probability is small and the system

remains on the same diabatic surface. The parameters of the Hamiltonian are constant, assuming that they change over distances that are large compared to the interaction region near the crossing point. The effect of the interaction vanishes when $t \rightarrow \infty$, far from the crossing point, since the energy difference between diabatic states is much larger than the coupling (*i.e.*, $\epsilon t \gg \Delta$). As Landau had formulated and solved the same model independently (albeit in the perturbative limit and with an error of a factor of 2π) [Landau, L. D. *Phys. Z.* 1932, **2**, 46], the equation came to be known as the Landau-Zener model.

We follow the derivation by Ho and Chibotaru who implemented the contour integration method proposed by Wittig [*J. Phys. Chem. B* 2005, **109**, 8428-8430] to find the transition probability without having to solve the Schrödinger equation.

We consider the dynamics of the wave packet for the 2-level system, evolving according to the time-dependent Schrödinger equation,

$$i\hbar\frac{\partial\psi}{\partial t} = \hat{H}\psi, \tag{314}$$

as described by the time-dependent wavefunction,

$$|\psi(t)\rangle = c_1(t)|\phi_1\rangle + c_2(t)|\phi_2\rangle. \tag{315}$$

Substituting $|\psi(t)\rangle$ and $\hat{H}$ into Eq. (314), we obtain:

$$\begin{aligned} i\hbar\dot{c}_1(t) &= c_1(t)\frac{\epsilon}{2}t + c_2(t)\frac{\Delta}{2}, \\ i\hbar\dot{c}_2(t) &= -c_2(t)\frac{\epsilon}{2}t + c_1(t)\frac{\Delta}{2}, \end{aligned} \tag{316}$$

which gives

$$\begin{aligned} i\hbar\ddot{c}_1(t) &= \dot{c}_1(t)\frac{\epsilon}{2}t + \dot{c}_2(t)\frac{\Delta}{2} + c_1(t)\frac{\epsilon}{2}, \\ i\hbar\ddot{c}_1(t) &= \left(c_1(t)\frac{\epsilon}{2}t + c_2(t)\frac{\Delta}{2}\right)\frac{\epsilon}{i2\hbar}t + \left(-c_2(t)\frac{\epsilon}{2}t + c_1(t)\frac{\Delta}{2}\right)\frac{\Delta}{i2\hbar} + c_1(t)\frac{\epsilon}{2}, \\ \ddot{c}_1(t) &= -c_1(t)\left(\frac{\epsilon^2 t^2}{4\hbar^2} + \frac{\Delta^2}{4\hbar^2} + \frac{i\epsilon}{2\hbar}\right). \end{aligned} \tag{317}$$

The desired quantity is the value of $c_1(t \rightarrow \infty)$ after the interaction is over. Remarkably, $c_1(t \rightarrow \infty)$ can be found directly, so it is not necessary to find $c_1(t)$ by solving the Schrödinger equation. This is accomplished by showing that,

$$\int_{-\infty}^{\infty} dt\frac{\dot{c}_1}{c_1} = \lim_{t\rightarrow\infty} ln\left(\frac{c_1(t)}{c_1(-t)}\right) = \pm\frac{\pi\Delta^2}{4\hbar\epsilon}, \tag{318}$$

so assuming that $\lim_{t\rightarrow-\infty} c_1(t) = 1$, we obtain $\lim_{t\rightarrow\infty} |c_1(t)|^2 = exp(-\frac{\pi\Delta^2}{2\epsilon})$, as the only possible physical solution, which is the familar Landau-Zener formula introduced by Eq. (313).

To derive Eq. (318), we assume that $\frac{\dot{c}_1}{c_1} = \frac{d}{dt} ln(c_1)$, which is a well-behaved function in the real axis of time, can be analytically continued into the complex plane to use the Cauchy integral theorem, as follows:

$$\begin{aligned} \oint_\gamma dz \frac{\dot{c}_1}{c_1} &= 0, \\ \int_{-\infty}^{\infty} dt \frac{\dot{c}_1}{c_1} + \int_\Omega dz \frac{\dot{c}_1}{c_1} &= 0, \\ \int_{-\infty}^{\infty} dt \frac{\dot{c}_1}{c_1} &= -\int_\Omega dz \frac{\dot{c}_1}{c_1}, \end{aligned} \tag{319}$$

where γ is the closed curve defined by the real axis from $-R$ to R and the semicircle Ω of radius R from R to $-R$ in the complex plane when $R \to \infty$. Equation (319) allows us to by-pass the need of solving the Schrödinger equation to find $\frac{\dot{c}_1}{c_1}$ in the real axis when we can evaluate the integral of its analytically continued form over the semicircle $\Omega : z = Re^{i\theta}$ in the complex plane.

To evaluate the integral over the semicircle, we show below that

$$\lim_{t \to \pm\infty} \dot{c}_1/c_1 = -i \left(\frac{1}{2\hbar} \epsilon t + \frac{1}{4\hbar} \frac{\Delta^2}{\epsilon t} \right) \tag{320}$$

giving

$$\lim_{|z| \to \pm\infty} \dot{c}_1/c_1 = -i \left(\frac{1}{2\hbar} \epsilon z + \frac{1}{4\hbar} \frac{\Delta^2}{\epsilon z} \right). \tag{321}$$

Therefore,

$$\int_\Omega dz \left(\frac{i}{2\hbar} \epsilon z + \frac{i}{4\hbar} \frac{\Delta^2}{\epsilon z} \right) = \lim_{R \to \infty} \int_0^{\pm\pi} d\theta i \left(\frac{i}{2\hbar} \epsilon R^2 e^{i2\theta} + \frac{i}{4\hbar} \frac{\Delta^2}{\epsilon} \right) = \pm \frac{\pi \Delta^2}{4\hbar\epsilon}. \tag{322}$$

To obtain Eq. (320), we note that at $t \to \pm\infty$ the $|c_1(t)|$ is constant, so $c_1(t) = |c_1| e^{-i\varphi(t)}$ and

$$\frac{\dot{c}_1}{c_1} = -i\dot{\varphi}(t). \tag{323}$$

Substituting into Eq. (317), we obtain:

$$-i\ddot{\varphi} - \dot{\varphi}^2 = -\left(\frac{\epsilon^2 t^2}{4\hbar^2} + \frac{\Delta^2}{4\hbar^2} + \frac{i\epsilon}{2\hbar} \right). \tag{324}$$

Separating real and imaginary parts, we obtain:

$$\begin{aligned} \ddot{\varphi} &= \frac{\epsilon}{2\hbar}, \\ \dot{\varphi}^2 &= \frac{\epsilon^2 t^2}{4\hbar^2} + \frac{\Delta^2}{4\hbar^2}. \end{aligned} \tag{325}$$

Therefore,

$$\dot{\varphi} = \pm\frac{\epsilon}{2\hbar}|t|\sqrt{1+\frac{\Delta^2}{\epsilon^2 t^2}} \tag{326}$$

and expanding the square root in the limit when $t \to \pm\infty$, we obtain:

$$\begin{aligned}\dot{\varphi} &= \pm\frac{\epsilon}{2\hbar}|t|(1+\frac{\Delta^2}{2\epsilon^2 t^2})+\cdots \\ &\approx \frac{\epsilon}{2\hbar}t(1+\frac{\Delta^2}{2\epsilon^2 t^2}),\end{aligned} \tag{327}$$

which can be substituted into Eq. (323) to obtain Eq. (320).

19.1 Marcus Formula

The Landau-Zener formula, introduced by Eq. (313), can be used to compute the probability of electron transfer, as follows:

$$\begin{aligned}P_{if} &= 1-e^{-\frac{2\pi(\Delta/2)^2}{\hbar\epsilon}}, \\ &= 1-exp\left(-\frac{\pi|H_{if}|^2}{\hbar\frac{d}{dx}\left(V_i-V_f\right)\dot{x}}\right),\end{aligned} \tag{328}$$

and can be expanded to first order when $|H_{if}|$ is sufficiently small (weak coupling limit), as follows:

$$P_{if} = \frac{2\pi|H_{if}|^2}{\hbar\frac{d}{dx}\left(V_i-V_f\right)\dot{x}}, \tag{329}$$

where $\dot{x} = \hbar k_*/m$ is the velocity at the crossing point and

$$\begin{aligned}\frac{d}{dx}\left(V_i-V_f\right)|_{x=x_{if}} &= m\omega^2(x-(x-x_{if})), \\ &= m\omega^2 x_{if},\end{aligned} \tag{330}$$

giving

$$P_{if} = \frac{2\pi|H_{if}|^2}{\hbar\omega^2 x_{if}\hbar k_*}, \tag{331}$$

Considering that the vibrational period is $2\pi/\omega$ and that the crossing is reached in half a period, we obtain the rate (*i.e.*, transition probability per unit time), as follows:

$$\begin{aligned}\Gamma_{if} &= \frac{\omega}{\pi}\frac{2\pi|H_{if}|^2}{\hbar\omega^2 x_{if}\hbar k_*}, \\ &= \frac{2\pi}{\hbar}\frac{|H_{if}|^2}{\pi\omega x_{if}\hbar k_*}.\end{aligned} \tag{332}$$

20 Tunneling Current

This subsection shows that the WKB tunneling probability can be used to estimate tunneling currents. We consider a 1-dimensional electron tunneling

$$\hat{H}\psi = \left[-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + eV(x)\right]\psi, \tag{333}$$

problem described by the Hamiltonian

$$\hat{H} = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + eV(x), \tag{334}$$

where e is the charge of the electron and

$$V(x) = \begin{cases} V_l & \text{if} \quad x < 0, \\ V_b & \text{if} \quad 0 < x < a, \\ V_r & \text{if} \quad x > a, \end{cases} \tag{335}$$

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:

$$\psi_l(x) = \begin{cases} \phi_l^+ + r_l\phi_l^- & \text{if} \quad x < 0 \\ Ae^{ik_b x} + Be^{-ik_b x} & \text{if} \quad 0 < x < a \\ t_r\phi_r^+ & \text{if} \quad x > a \end{cases} \tag{336}$$

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:

$$\begin{aligned} k_l^{-1/2} + k_l^{-1/2}r_l &= A + B, \\ Ae^{ik_b a} + Be^{-ik_b a} &= k_r^{-1/2}t_r e^{ik_r a}, \\ k_l^{1/2}(1 - r_l) &= k_b(A - B), \\ k_b(Ae^{ik_b a} - Be^{-ik_b a}) &= k_r^{1/2}t_r e^{ik_r a}. \end{aligned} \tag{337}$$

The transmission amplitude t_r, reflection amplitude r_l and coefficients A and B can be obtained by solving for them from Eq. (337).

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:

$$j_i(x,t) = \frac{\hbar}{2mi}\left(\psi_i^*(x,t)\frac{\partial \psi_i(x,t)}{\partial x} - \psi_i(x,t)\frac{\partial \psi_i^*(x,t)}{\partial x}\right), \tag{338}$$

or

$$\begin{aligned} j_i(x) &= \frac{1}{2}\psi_i^*(x,t)\left(-i\frac{\hbar}{m}\frac{\partial}{\partial x}\right)\psi_i(x,t) + c.c., \\ &= \frac{1}{2}\psi_i^*(x,t)\frac{\hat{p}}{m}\psi_i(x,t) + c.c., \\ &= \text{Re}[\psi_i^*(x,t)\hat{v}\psi_i(x,t)] = \frac{\hbar}{m}. \end{aligned} \tag{339}$$

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:

$$\begin{aligned} j_t(x) &= \frac{1}{2}\psi_t^*(x,t)\left(-i\frac{\hbar}{m}\frac{\partial}{\partial x}\right)\psi_t(x,t) + c.c., \\ &= |t_r|^2\frac{\hbar}{m}. \end{aligned} \tag{340}$$

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:

$$\psi_r(x) = \begin{cases} \phi_r^- + r_r\phi_r^+ & \text{if} \quad x > a \\ Ae^{ik_b x} + Be^{-ik_b x} & \text{if} \quad 0 < x < a \\ t_l\phi_l^- & \text{if} \quad x < 0 \end{cases} \tag{341}$$

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: $\mathbf{c}_{out} = \mathbf{S}\mathbf{c}_{in}$:

$$\begin{pmatrix} c_{out}^l \\ c_{out}^r \end{pmatrix} = \begin{pmatrix} r_l & t_l \\ t_r & r_r \end{pmatrix}\begin{pmatrix} c_{in}^l \\ c_{in}^r \end{pmatrix} \tag{342}$$

Due to the conservation of probability, the S-matrix must be unitary: $\mathbf{S}^{-1} = \mathbf{S}^\dagger$. Therefore, $\mathbf{S}\mathbf{S}^\dagger = 1$:

$$\begin{pmatrix} r_l & t_l \\ t_r & r_r \end{pmatrix}\begin{pmatrix} r_l^\dagger & t_r^\dagger \\ t_l^\dagger & r_r^\dagger \end{pmatrix} = 1 \tag{343}$$

which gives

$$r_l r_l^\dagger + t_l t_l^\dagger = 1. \tag{344}$$

In addition, $\mathbf{S}^\dagger\mathbf{S} = 1$:

$$\begin{pmatrix} r_l^\dagger & t_r^\dagger \\ t_l^\dagger & r_r^\dagger \end{pmatrix} \begin{pmatrix} r_l & t_l \\ t_r & r_r \end{pmatrix} = 1 \tag{345}$$

which gives

$$t_l^\dagger t_l + r_r^\dagger r_r = 1 \tag{346}$$

Therefore, according to Eqs. (344) and (346), 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:

$$|r_l|^2 = |r_r|^2 = R. \tag{347}$$

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 $x > a$:

$$(1 - |r_l|^2) = |t_r|^2, \tag{348}$$

and

$$(1 - |r_r|^2) = |t_l|^2. \tag{349}$$

Dividing Eq. (348) by Eq. (349) and using (347), we obtain:

$$t_l = t_r. \tag{350}$$

Therefore,

$$\begin{aligned} T_l(E) &= |t_r|^2, \\ &= |t_l|^2 = T_r(E), \end{aligned} \tag{351}$$

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:

$$\begin{aligned} I &= 2e \int_0^\infty dk_l n_l(k_l) \frac{\hbar k_l}{m} T_l - 2e \int_0^\infty dk_r n_r(k_r) \frac{\hbar k_r}{m} T_r, \\ &= \frac{2e}{2\pi} \int_0^\infty dE\, T(E) \left(n_l(E) \frac{\hbar k_l}{m} \left| \frac{\partial k_l}{\partial E} \right| - n_r(E) \frac{\hbar k_r}{m} \left| \frac{\partial k_r}{\partial E} \right| \right), \\ &= \frac{2e}{2\pi} \int_0^\infty dE\, T(E) \left(n_l(E) \frac{\hbar k_l}{m} \left| \frac{m}{\hbar^2 k_l} \right| - n_r(E) \frac{\hbar k_r}{m} \left| \frac{m}{\hbar^2 k_r} \right| \right), \\ &= \frac{2e}{h} \int_0^\infty dE\, T(E) \left(n_l(E) - n_r(E) \right), \end{aligned} \tag{352}$$

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. (352) 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:

$$n(E) = \frac{1}{e^{\beta(E-E_F)}+1}, \tag{353}$$

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:

$$\begin{aligned} n_l(E) &= n(E - E_F) + \frac{\partial n(E)}{\partial E} eV_l + \cdots, \\ n_r(E) &= n(E - E_F) + \frac{\partial n(E)}{\partial E} eV_r + \cdots, \end{aligned} \tag{354}$$

and write the Landauer formula, giving the current in the form of the Ohm's lAF, as follows:

$$\begin{aligned} I &= \frac{2e}{h} \int_0^\infty dE\, T(E)\, (n_l(E) - n_r(E))\,, \\ &= \frac{2e^2}{h} \int dE\, T(E) \frac{\partial n(E)}{\partial E} \Delta V, \\ &= G(E)\Delta V, \end{aligned} \tag{355}$$

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 $G_0 = \frac{2e^2}{h} = [12.906\ \mathrm{k\Omega}]^{-1}$ 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

$$\begin{aligned} I &= \frac{2e^2}{h} \int dE\, T(E) \delta(E_F - E) \Delta V, \\ &= \frac{2e^2}{h}\, T(E_F) \Delta V. \end{aligned} \tag{356}$$

In this low-temperature limit, the conductance is the transmission times the quantum of conductance, $G(E) = \frac{2e^2}{h}\, T(E_F)$.

The transmission coefficient $T(E)$ can be estimated, under the WKB approximation, as follows:

$$T(E) = e^{-2\int_0^a dx \sqrt{2m|E-\xi(x)|/\hbar^2}}, \tag{357}$$

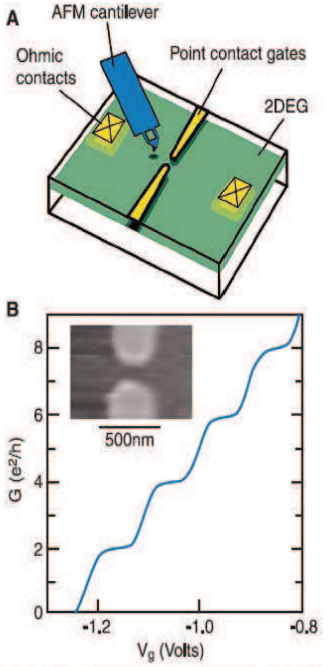


Fig. 1. (A) Schematic diagram of the experimental setup. QPC conductance is measured as a function of AFM tip position. (B) Point contact conductance G versus gate voltage V_g with no tip present at temperature $T = 1.7$ K. Plateaus at integer multiples of $2\ e^2/h$ are clearly seen. The inset shows a topographic image of the point contact device.

where $\xi(x) = V_b$ describes the tunneling barrier according to Eq. (335).

To derive Eq. (357), we consider the WKB approximate solution of Eq. (333), with the following functional form:

$$\psi(x) = \psi_0 e^{i\int_0^x k(x')dx'}, \tag{358}$$

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. (358), into Eq. (333), we obtain:

$$-\frac{\hbar^2}{2m}\frac{\partial^2\psi(x)}{\partial x^2} + V(x)\psi(x) = E\psi(x) - \Delta, \tag{359}$$

with $\Delta = ik'(x)\frac{\hbar^2}{2m}\psi(x)$. Therefore, the WKB solution is a good approximation when $|k'(x)| \ll k(x)^2$.

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:

$$\psi(x) = \psi(0)e^{-\int_0^x dx'\sqrt{2m|E-\xi(x')|/\hbar^2}}, \tag{360}$$

since $E < \xi(x)$. In particular, at $x = a$, the probability density is

$$|\psi(x)|^2 = |\psi(0)|^2 e^{-2\int_0^a dx' |k(x')|}. \tag{361}$$

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

$$\begin{aligned} T(E) &= \frac{|\psi(a)|^2}{|\psi(0)|^2}, \\ &= e^{-2\int_0^a dx \sqrt{2m|E-\xi(x)|/\hbar^2}}. \end{aligned} \tag{362}$$

21 Grover Algorithm

The goal of this section is to explain the Grover algorithm as applied first to a system of 2 qubits in a uniform superposition with $N = 2^2$ states. We also discuss the straightforward generalization to a system with an arbitrary number qubits and for applications to optimization problems.

Algorithm: Given a so-called *oracle* $\hat{O}$ that changes the sign of a target state (*e.g.*, $|11\rangle$) in a uniform superposition,[3]

$$|s\rangle = \frac{1}{\sqrt{N}}(|00\rangle + |10\rangle + |01\rangle + |11\rangle), \tag{363}$$

as follows:

$$\hat{O}|s\rangle = \frac{1}{\sqrt{N}}(|00\rangle + |10\rangle + |01\rangle - |11\rangle), \tag{364}$$

the Grover algorithm reveals the nature of that target state by rotating the uniform superposition $|s\rangle$ to align it with that target state $|11\rangle$. The rotation is performed by applying the product of operators $\hat{D}\hat{O}$, where $\hat{D}$ is the so-called Grover's *diffusion operator* that changes the sign of the component perpendicular to the initial superposition state, as follows:

$$|\hat{D}\rangle = 2|s\rangle\langle s| - I. \tag{365}$$

Note that in the basis of qubit states (*i.e.,* the *computational basis*), the matrix elements of $\hat{D}$ are defined, as follows: $\langle j|\hat{D}|k\rangle = \frac{2}{N} - \delta_{j,k}$. Full alignment can be verified by checking that the oracle changes the sign of the resulting state. The nature of the target state is revealed by making a measurement giving the expectation value corresponding to the target state.

Analogous to $\hat{D}$, the oracle can also be represented as a Householder transformation $\hat{O} = I - 2|11\rangle\langle 11|$, or as a matrix in the basis of qubit states with $\langle j|\hat{O}|k\rangle = \delta_{jk}(1 - \frac{2}{N}\delta_{j,11})$. In general, the oracle can also be represented, as follows: $\hat{O} = e^{i\pi f(j)}$, where $f(j) = 1$ for the target state (*e.g.*, $|j\rangle = |11\rangle$) and $f(j) = 0$, otherwise.

[3] An *oracle* in quantum computing is a 'black-box' operator that takes n q-bits and performs a unitary transformation, analogous to the oracles of classical computations that take as an input an n-bit number x and output a function $f(x)$.

The uniform superposition can be prepared by starting with all qubits in the ground state and applying the Hadamard operator to each qubit, as follows: $\hat{H}_2|00\rangle = \hat{H}_1 \otimes \hat{H}_1|00\rangle$, since the Hadamard operator $\hat{H}_1 = \frac{1}{\sqrt{2}}(\hat{\sigma}_z + \hat{\sigma}_x)$ transforms $|0\rangle$ into the symmetric linear combination $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$.

Remarkably, the state $\hat{D}\hat{O}|s\rangle$ is more aligned with the target state $|11\rangle$ than the initial state $|s\rangle$, as shown geometrically in the figure below. The procedure can be repeated n times to maximize the alignment, with n determined by the number of qubits as explained below.

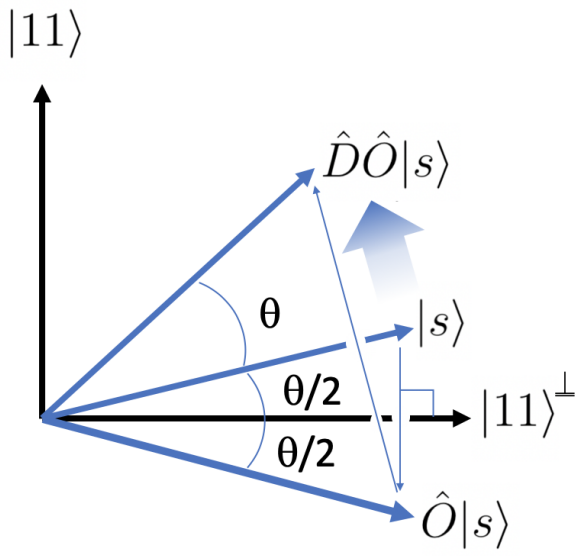
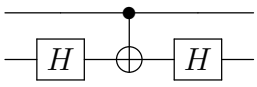


Figure 1: Geometric representation of the first iteration of Grover's algorithm. The initial superposition state $|s\rangle$ is rotated by an angle θ towards the target vector $|11\rangle$ by first applying the oracle $\hat{O}$ that inverts it along the direction of the target state, and then the diffusion operator $\hat{D}$ that inverts it relative to the direction of the superposition state.

Oracle operator: To explain how to construct $\hat{O}$ in terms of unitary operators for the example described above, we note that

$$-|1\rangle = \hat{H}_1\hat{\sigma}_x\hat{H}_1|1\rangle. \tag{366}$$

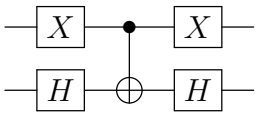
since the Hadamard operator $\hat{H}_1 = \frac{1}{\sqrt{2}}(\hat{\sigma}_z + \hat{\sigma}_x)$ transforms $|1\rangle$ into the antisymmetric linear combination $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, the NOT operator $\hat{x}$ changes the sign of the state $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ (by changing $|0\rangle$ into $|1\rangle$ and $|1\rangle$ into $|0\rangle$), and applying the Hadamard operator to $|-\rangle$ returns the original state $|1\rangle$. By using CNOT instead of NOT, as shown in the circuit, below:



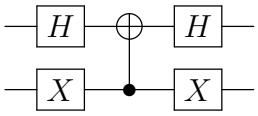
we change the sign of $|1\rangle$ in the second qubit only when it is preceded by the control qubit $|1\rangle$, as necessary when applying $\hat{O}$ to the uniform superposition, according to Eq. (364).

Diffusion operator: The diffusion operator changes the sign of all of the terms orthogonal to the uniform superposition. Therefore, it is possible to implement it by first orienting the state along a convenient direction where it is easier to change the sign of all of the orthogonal state and the rotate it back to its original orientation. For example, rotating the state $\hat{O}|s\rangle$ so that its component along the direction of the uniform superposition points along one of the computational states (*e.g.,*, along the $|00\rangle$ direction) so one can change the sign of all of the terms that are orthogonal to that direction (*e.g.*, $|00\rangle$) and then rotate the resulting state back so that the superposition state component points along its original direction.

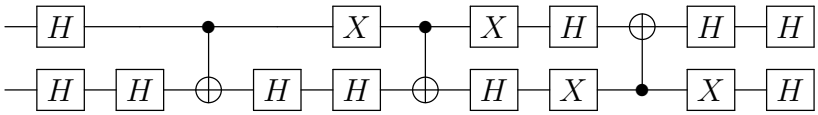
The circuit given above can be used to change the sign of the component along the $|11\rangle$ direction. In addition, the analogous circuit but with a NOT gate previously applied to the first qubit (and subsequently applied as well to avoid modifying that qubit) would change the sign of the term along $|01\rangle$, as follows:



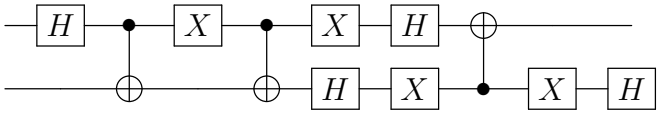
and the same circuit but with exchanged the control and target qubits would change the sign of the term associated with the direction $|10\rangle$, as follows:



so, the complete diffusion operator $\hat{D}$ can be implemented, as follows:



which can be simplified, as follows:



since $\hat{H}\hat{H} = I$.

Number of steps n: Here, we show that $n = 1$ for a system of 2 qubits, thus outperforming a classical search that would need at least 2 steps to search a state component in logarithmic time –*i.e.*, $\log_2(\#qubits)$. We note that

$$\langle 11|s\rangle = \frac{1}{\sqrt{N}}, \tag{367}$$

so

$$\hat{O}|s\rangle = |s\rangle - \frac{2}{\sqrt{N}}|11\rangle, \tag{368}$$

and

$$\hat{D}\left(|s\rangle - \frac{2}{\sqrt{N}}|11\rangle\right) = (2|s\rangle\langle s| - I)\left(|s\rangle - \frac{2}{\sqrt{N}}|11\rangle\right). \tag{369}$$

Therefore,

$$\hat{D}\hat{O}|s\rangle = (2|s\rangle\langle s|)\left(|s\rangle - \tfrac{2}{\sqrt{N}}|11\rangle\right) - \left(|s\rangle - \tfrac{2}{\sqrt{N}}|11\rangle\right) \quad (370)$$

$$= \left(2|s\rangle - \tfrac{4}{N}|s\rangle\right) - |s\rangle + \tfrac{2}{\sqrt{N}}|11\rangle \quad (371)$$

$$= \tfrac{N-4}{N}|s\rangle + \tfrac{2}{\sqrt{N}}|11\rangle \quad (372)$$

so for a system with 2 qubits, $N = 4$, making $(\hat{D}\hat{O})^n|s\rangle = |11\rangle$ when $n = 1$.

More generally we note that, according to the figure given above, the projection of $(\hat{D}\hat{O})^n|s\rangle$ along the direction of $|11\rangle$ is $\sin((2n+1)\frac{\theta}{2})$. To maximize the projection, we need to have $(2n+1)\frac{\theta}{2} = \frac{\pi}{2}$ with $\frac{\theta}{2} = \frac{1}{\sqrt{N}}$. So, $(2n+1)\frac{1}{\sqrt{N}} = \frac{\pi}{2}$ and when n is large, $2n\frac{1}{\sqrt{N}} \approx \frac{\pi}{2}$, giving $n \approx \frac{\pi\sqrt{N}}{4}$. So, remarkably, $n \approx \sqrt{N}$ when N is large –*i.e.*, quadratically faster than the classical search where n is of order N.

IBM Experience: The YouTube videos 24 and 25 show how to implement the Grover algorithm for 2 qubits, as described above, on the IBM Quantum computer.

Matlab function: The gsa.m Matlab function simulates the Grover algorithm.

Grover Optimization: Grover's quantum computational search procedure can provide the basis for implementing adaptive global optimization algorithms. An example of such methods is the *Grover adaptive search* (GAS) algorithm where the global minimum of a cost function V is iteratively searched for with an adaptive oracle, as follows. Given an initial state $|j_0\rangle$ and its corresponding expectation value $V_0 = \langle j_0|V|j_0\rangle$, the oracle $\hat{O} = e^{i\pi f(j)}$ is defined with $f(j) = 1$ for states $|j\rangle$ with expectation value $\langle j|V|j\rangle < V_0$. Applying the Grover algorithm to a uniform superposition, we find a state $|j_1\rangle$ whose expectation value $V_1 < V_0$ after $r = \pi\sqrt{N}/4$ rotations. The oracle is then adapted with $f(j) = 1$ for states $|j\rangle$ with $\langle j|V|j\rangle < V_1$, and the process is iterated m times until convergence to find the global minimum state $|m\rangle$ with $V_m < V_{m-1} < \cdots < V_0$.

As an example of Grover minimization, we consider the problem of finding the configuration of a conjugated polyene chain with Cartesian atomic coordinates $x_1, \cdots x_n$, assuming that bond-lengths and bending angles are known but the 1-4 dihedrals are yet to be determined since their π or $-\pi$ (*cis* or *trans*) configurations must fulfill a constraining set S of interatomic distances $d_{ij} = \|x_i - x_j\|$ determined by NMR. If all interatomic distances are determined, then the problem is trivial and can be solved in n steps. However, the problem is NP-hard when only some of the distances are known.

Therefore, we need to find the coordinates $x_1, \cdots x_n$ that minimize the following cost function:

$$g(x_1, x_2, \cdots, x_n) = \sum_{(i,j)\epsilon S} (d_{ij} - \|x_i - x_j\|)^2, \quad (373)$$

and thus make $g = 0$. We note that only $n-3$ dihedrals have to be specified to define all interatomic distances, since all bond-lengths and bending angles are given and the positions of the first 3 atoms are defined by the bond-lengths and bending angles. Since each dihedral can be either *cis* or *trans*, we have a total of 2^{n-3} possible configurations, with only some of them satisfying Eq. (373).

To implement the Grover optimization algorithm, we prepare a register with $n-3$ qubits in a uniform superposition, where the state $|0\rangle$ of the j-th qubit corresponds to the *cis* state of the j-th

dihedral and $|1\rangle$ corresponds to the *trans* state of that dihedral. The oracle $\hat{O} = e^{i\pi f(j)}$ is defined so that $f(j) = 1$ if state $|j\rangle$ satisfies Eq. (373) and $f(j) = 0$, otherwise.

22 Second Quantization

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 $A_{\nu_j,\nu_k} = \langle \nu_j | \hat{A} | \nu_k \rangle$.

22.1 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

$$\sum_{\nu_j} \psi_{\nu_j}(\mathbf{r}')^* \psi_{\nu_j}(\mathbf{r}) = \delta(\mathbf{r}' - \mathbf{r}), \tag{374}$$

and

$$\int d\mathbf{r}\, \psi_{\nu_j}(\mathbf{r})^* \psi_{\nu_k}(\mathbf{r}) = \delta_{\nu_j \nu_k}. \tag{375}$$

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:

$$\Psi(\mathbf{r}_1, \mathbf{r'}_2, \cdots, \mathbf{r'}_N) = \sum_{\nu_1} \psi_{\nu_1}(\mathbf{r}_1) \int d\mathbf{r'}_1 \psi_{\nu_1}(\mathbf{r'}_1)^* \Psi(\mathbf{r'}_1, \mathbf{r'}_2, \cdots, \mathbf{r'}_N), \tag{376}$$

and then we proceed analogously with the other coordinates, so we obtain:

$$\Psi(\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_N) = \sum_{\nu_1, \cdots, \nu_N} c_{\nu_1, \cdots, \nu_N} \prod_{j=1}^{N} \psi_{\nu_j}(\mathbf{r}_j), \tag{377}$$

with

$$c_{\nu_1, \cdots, \nu_N} = \int d\mathbf{r'}_1 \psi_{\nu_1}(\mathbf{r'}_1)^* \cdots \int d\mathbf{r'}_N \psi_{\nu_N}(\mathbf{r'}_N)^* \Psi(\mathbf{r'}_1, \mathbf{r'}_2, \cdots, \mathbf{r'}_N). \tag{378}$$

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*):

$$\begin{aligned}\hat{S}_{\pm}\prod_{j=1}^{N}\psi_{\nu_j}(\mathbf{r}_j) &= \frac{1}{\prod_\nu \sqrt{n_\nu!}}\frac{1}{\sqrt{N!}}\begin{vmatrix}\psi_{\nu_1}(\mathbf{r}_1) & \psi_{\nu_1}(\mathbf{r}_2) & \cdots & \psi_{\nu_1}(\mathbf{r}_N)\\ \psi_{\nu_2}(\mathbf{r}_1) & \psi_{\nu_2}(\mathbf{r}_2) & \cdots & \psi_{\nu_2}(\mathbf{r}_N)\\ \cdots & \cdots & \cdots & \cdots\\ \psi_{\nu_N}(\mathbf{r}_1) & \psi_{\nu_N}(\mathbf{r}_2) & \cdots & \psi_{\nu_N}(\mathbf{r}_N)\end{vmatrix}_{\pm}, \\ &= \langle \mathbf{r}|\psi_{\nu_1}\psi_{\nu_2}\cdots\psi_{\nu_N}\rangle,\end{aligned} \tag{379}$$

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 sign-less.

22.2 Occupation Number Basis

The product states, introduced by Eq. (379), 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,

$$\hat{n}_{\nu_k}|n_{\nu_1}, n_{\nu_2}, n_{\nu_3}, \cdots\rangle = n_{\nu_k}|n_{\nu_1}, n_{\nu_2}, n_{\nu_3}, \cdots\rangle. \tag{380}$$

For fermions, $n_{\nu_k} = 0, 1$ while for bosons $n_{\nu_k} = 0, 1, 2, \cdots$ is a positive integer.

22.3 Creation and Annihilation Operators

Bosons: The creation and annihilation 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:

$$\hat{n}_{\nu_j}|n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots\rangle = n_{\nu_j}|n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots\rangle, \tag{381}$$

and raise or lower the occupation of that state, as follows:

$$\begin{aligned}\hat{b}_j^\dagger|n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots\rangle &= B_+(n_{\nu_j})|n_{\nu_1}, n_{\nu_2}, \cdots (n_{\nu_j}+1), \cdots\rangle,\\ \hat{b}_j|n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots\rangle &= B_-(n_{\nu_j})|n_{\nu_1}, n_{\nu_2}, \cdots (n_{\nu_j}-1), \cdots\rangle,\end{aligned} \tag{382}$$

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

$$\begin{aligned}\hat{b}_j^\dagger|n_{\nu_1}, n_{\nu_2}, \cdots 0, \cdots\rangle &= |n_{\nu_1}, n_{\nu_2}, \cdots 1, \cdots\rangle,\\ \hat{b}_j|n_{\nu_1}, n_{\nu_2}, \cdots 1, \cdots\rangle &= |n_{\nu_1}, n_{\nu_2}, \cdots 0, \cdots\rangle.\end{aligned} \tag{383}$$

Therefore,

$$\begin{aligned}\hat{b}_j\hat{b}_j^\dagger|n_{\nu_1}, n_{\nu_2}, \cdots 0, \cdots\rangle &= |n_{\nu_1}, n_{\nu_2}, \cdots 0, \cdots\rangle, \\ \hat{b}_j^\dagger\hat{b}_j|n_{\nu_1}, n_{\nu_2}, \cdots 0, \cdots\rangle &= 0,\end{aligned} \tag{384}$$

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. (381), as follows:

$$\begin{aligned}\langle n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots|\hat{b}_j^\dagger\hat{b}_j|n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots\rangle &= n_{\nu_j}, \\ \langle n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots|\hat{b}_j^\dagger\hat{b}_j|n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots\rangle &= B_-(n_{\nu_j})^2,\end{aligned} \tag{385}$$

so $B_-(n_{\nu_j}) = \sqrt{n_{\nu_j}}$. Analogously, we obtain

$$\begin{aligned}\langle n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots|\hat{b}_j\hat{b}_j^\dagger|n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots\rangle &= B_+(n_{\nu_j})^2, \\ (n_{\nu_j} + 1) &= B_+(n_{\nu_j})^2,\end{aligned} \tag{386}$$

$B_+(n_{\nu_j}) = \sqrt{n_{\nu_j} + 1}$. Therefore,

$$(\hat{b}_j^\dagger)^{n_\nu}|n_{\nu_1}, n_{\nu_2}, \cdots 0, \cdots\rangle = \sqrt{n_\nu!}|n_{\nu_1}, n_{\nu_2}, \cdots n_\nu, \cdots\rangle. \tag{387}$$

or

$$|n_{\nu_1}, n_{\nu_2}, n_{\nu_3}, \cdots\rangle = \prod_j \frac{(\hat{b}_j^\dagger)^{n_{\nu_j}}}{\sqrt{n_\nu!}}|0, 0, 0, \cdots\rangle. \tag{388}$$

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 $\hat{c}_j|1\rangle = 0$, $\hat{c}_j^\dagger|0\rangle = |1\rangle$, $\hat{c}_j|0\rangle = 0$, 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 $\hat{c}_j\hat{c}_j^\dagger + \hat{c}_j^\dagger\hat{c}_j = 0$.

22.4 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 $\hat{T}_k = \frac{\hat{p}_k^2}{2m_k}$:

$$\begin{aligned}
\langle \mathbf{r}|\hat{T}|\psi_{\nu_1}\psi_{\nu_2}\cdots\psi_{\nu_N}\rangle &= \sum_{\nu_j}\langle \mathbf{r}|\psi_{\nu_j}\rangle\langle\psi_{\nu_j}|\hat{T}|\psi_{\nu_1}\psi_{\nu_2}\cdots\psi_{\nu_N}\rangle \\
&= \sum_{\nu_j}\langle \mathbf{r}|\psi_{\nu_j}\rangle \sum_{k=1}^{N}\langle\psi_{\nu_j}|\hat{T}_k|\psi_{\nu_1}\psi_{\nu_2}\cdots\psi_{\nu_N}\rangle \\
&= \sum_{\nu_j}\langle \mathbf{r}|\psi_{\nu_j}\rangle \sum_{k=1}^{N}\langle\psi_{\nu_j}|\hat{T}_k|\psi_{\nu_k}\rangle\langle\mathbf{r}|\hat{b}_{\nu_k}|\psi_{\nu_1}\psi_{\nu_2}\cdots\psi_{\nu_N}\rangle \\
&= \sum_{\nu_j,\nu_l}\sum_{k=1}^{N}\langle \mathbf{r}|\psi_{\nu_j}\rangle\delta_{\nu_l,\nu_k}T_{\nu_j,\nu_l}\langle\mathbf{r}|\hat{b}_{\nu_k}|\psi_{\nu_1}\psi_{\nu_2}\cdots\psi_{\nu_N}\rangle \\
&= \sum_{\nu_j,\nu_l}\sum_{k=1}^{N}\delta_{\nu_l,\nu_k}T_{\nu_j,\nu_l}\langle\mathbf{r}|\hat{b}^\dagger_{\nu_j}\hat{b}_{\nu_k}|\psi_{\nu_1}\psi_{\nu_2}\cdots\psi_{\nu_N}\rangle
\end{aligned} \tag{389}$$

Therefore, summing over k we obtain:

$$\hat{T} = \sum_{\nu_j,\nu_l} T_{\nu_j,\nu_l}\hat{b}^\dagger_{\nu_j}\hat{b}_{\nu_l}. \tag{390}$$

Analogously, any 2-particle operator $\hat{V}$ such as the pair-wise additive potential,

$$\hat{V} = \frac{1}{2}\sum_{j=1}^{N}\sum_{k\neq j} V(x_j, x_k), \tag{391}$$

can be written in second quantization, as follows:

$$\hat{V} = \sum_{\nu_j,\nu_i,\nu_l,\nu_k} V_{\nu_j,\nu_i,\nu_l,\nu_k}\hat{b}^\dagger_{\nu_j}\hat{b}^\dagger_{\nu_i}\hat{b}_{\nu_l}\hat{b}_{\nu_k} \tag{392}$$

where $V_{\nu_j,\nu_i,\nu_l,\nu_k} = \langle\psi_{\nu_j}\psi_{\nu_i}|V(x_1,x_2)|\psi_{\nu_l}\psi_{\nu_k}\rangle$.

22.5 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:

$$|\psi_{\mu_j}\rangle = \sum_k |\psi_{\nu_k}\rangle\langle\psi_{\nu_k}|\psi_{\mu_j}\rangle, \tag{393}$$

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,

$$\hat{a}_{\mu_j}|0\rangle = \sum_k \langle\psi_{\nu_k}|\psi_{\mu_j}\rangle \hat{a}_{\nu_k}|0\rangle, \tag{394}$$

or

$$\hat{a}_{\mu_j} = \sum_k \langle\psi_{\nu_k}|\psi_{\mu_j}\rangle \hat{a}_{\nu_k}, \tag{395}$$

and

$$\hat{a}^\dagger_{\mu_j} = \sum_k \langle\psi_{\nu_k}|\psi_{\mu_j}\rangle^* \hat{a}^\dagger_{\nu_k}, \tag{396}$$

22.6 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 $\tilde{p}_{\nu_j} = \frac{i}{\sqrt{2}}[\hat{b}^\dagger_{\nu_j} - \hat{b}_{\nu_j}]$, we obtain:

$$\hat{b}^\dagger_{\nu_j} = \frac{1}{\sqrt{2}}\left[\tilde{x}_{\nu_j} - i\tilde{p}_{\nu_j}\right], \tag{397}$$

and

$$\hat{b}_{\nu_j} = \frac{1}{\sqrt{2}}\left[\tilde{x}_{\nu_j} + i\tilde{p}_{\nu_j}\right]. \tag{398}$$

Therefore,

$$\begin{aligned}
\hat{n}_{\nu_j} &= \frac{1}{2}(\tilde{x}_{\nu_j} - i\tilde{p}_{\nu_j})(\tilde{x}_{\nu_j} + i\tilde{p}_{\nu_j}) \\
&= \frac{1}{2}(\tilde{x}^2_{\nu_j} + i[\tilde{x}_{\nu_j}, \tilde{p}_{\nu_j}] + \tilde{p}^2_{\nu_j}) \\
&= \frac{1}{2}(\tilde{x}^2_{\nu_j} + \tilde{p}^2_{\nu_j} - \hbar)
\end{aligned} \tag{399}$$

Substituting into Eq. (400), we obtain:

$$\begin{aligned}
\hat{T} &= \frac{1}{2}\sum_{\nu_j,\nu_l} T_{\nu_j,\nu_l}\left[\tilde{x}_{\nu_j} - i\tilde{p}_{\nu_j}\right]\left[\tilde{x}_{\nu_l} + i\tilde{p}_{\nu_l}\right], \\
&= \frac{1}{2}\sum_{\nu_j} T_{\nu_j,\nu_j}(\tilde{x}^2_{\nu_j} + \tilde{p}^2_{\nu_j} - \hbar) + \frac{1}{2}\sum_{\nu_j}\sum_{\nu_l\neq\nu_j} T_{\nu_j,\nu_l}\left[\tilde{x}_{\nu_j} - i\tilde{p}_{\nu_j}\right]\left[\tilde{x}_{\nu_l} + i\tilde{p}_{\nu_l}\right] \\
&= \frac{1}{2}\sum_{\nu_j} T_{\nu_j,\nu_j}(\tilde{x}^2_{\nu_j} + \tilde{p}^2_{\nu_j} - \hbar) + \frac{1}{2}\sum_{\nu_j}\sum_{\nu_l\neq\nu_j} T_{\nu_j,\nu_l}\left[\tilde{x}_{\nu_j}\tilde{x}_{\nu_l} + \tilde{p}_{\nu_j}\tilde{p}_{\nu_l}\right]
\end{aligned} \tag{400}$$

since $[\tilde{x}_{\nu_j}, \tilde{p}_{\nu_l}] = i\hbar\delta_{\nu_l,\nu_j}$ while $[\tilde{x}_{\nu_j}, \tilde{x}_{\nu_l}] = 0$ and $[\tilde{p}_{\nu_j}, \tilde{p}_{\nu_l}] = 0$.

22.7 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:

$$H^{MM}(\tilde{\mathbf{x}}, \tilde{\mathbf{p}}) = \frac{1}{2} \sum_j \sum_k (\tilde{x}_j \tilde{x}_k + \tilde{p}_j \tilde{p}_k) H_{jk} - \frac{1}{2} \sum_k H_{kk} \tag{401}$$

The Meyer-Miller Hamiltonian, introduced by Eq. (401), can be obtained by first writing the Hamiltonian in second quantization, as follows:

$$\begin{aligned} \hat{H} &= \sum_{j=1}^{N} \sum_{k=1}^{N} H_{jk} |j\rangle\langle k|, \\ &= \sum_{j=1}^{N} \sum_{k=1}^{N} H_{jk} \hat{a}_j^\dagger \hat{a}_k, \end{aligned} \tag{402}$$

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

$$\begin{aligned} \hat{H} &= \frac{1}{2} \sum_{j=1}^{N} \sum_{k=1}^{N} H_{jk} (\tilde{x}_j - i\tilde{p}_j)(\tilde{x}_k + i\tilde{p}_k), \\ &= \frac{1}{2} \sum_{j=1}^{N} \sum_{k=1}^{N} H_{jk} (\tilde{x}_j \tilde{x}_k + \tilde{p}_j \tilde{p}_k - \delta_{jk}). \end{aligned} \tag{403}$$

Note that the delta function, in Eq. (403), is introduced by the commutator $[\tilde{x}_j, \tilde{p}_k] = i\delta_{j\,k}$. Therefore, the 'zero-point energy' term effectively introduces the commutation relation between the coordinates and momenta,

$$\begin{aligned} \tilde{x}_j &= \frac{1}{\sqrt{2}} [\hat{a}_j^\dagger + \hat{a}_j], \\ \tilde{p}_j &= \frac{i}{\sqrt{2}} [\hat{a}_j^\dagger - \hat{a}_j], \end{aligned} \tag{404}$$

with

$$\begin{aligned} \tilde{x}_j &= \hat{x}_j \sqrt{\frac{m\omega}{\hbar}}, \\ \tilde{p}_j &= \hat{p}_j \sqrt{\frac{1}{m\omega\hbar}}, \end{aligned} \tag{405}$$

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

$$\begin{aligned} H_j &= \frac{\hat{p_j}^2}{2m} + \frac{1}{2}m\omega^2\hat{x}_j^2, \\ &= \frac{\tilde{p_j}^2}{2m}m\omega\hbar + \frac{1}{2}m\omega^2\frac{\hbar}{m\omega}\tilde{x_j}^2, \\ &= \frac{\hbar\omega}{2}\left[\tilde{p}_j^2 + \tilde{x}_j^2\right]. \end{aligned} \tag{406}$$

Considering that

$$\begin{aligned} \hat{n}_j &= \hat{a}_j^\dagger\hat{a}_j, \\ &= \frac{1}{2}(\tilde{x}_j - i\tilde{p}_j)(\tilde{x}_j + i\tilde{p}_j), \\ &= \frac{1}{2}(\tilde{x}_j^2 + i[\tilde{x}_j, \tilde{p}_j] + \tilde{p}_j^2), \\ &= \frac{1}{2}(\tilde{x}_j^2 + \tilde{p}_j^2 - 1), \end{aligned} \tag{407}$$

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

$$H_j = \hbar\omega\left(\hat{n}_j + \frac{1}{2}\right), \tag{408}$$

with eigenstates

$$\begin{aligned} |0\rangle &= \left(\frac{m\omega}{\pi\hbar}\right)^{1/4}\exp(-\frac{m\omega}{2\hbar}x_j^2), \\ |\nu+1\rangle &= \frac{1}{\sqrt{\nu+1}}\hat{a}_j^\dagger|\nu\rangle, \\ |1\rangle &= \frac{1}{\sqrt{2}}\left(\hat{x}_j\sqrt{\frac{m\omega}{\hbar}} - i\frac{1}{\sqrt{m\omega\hbar}}\hat{p}_j\right)\left(\frac{m\omega}{\pi\hbar}\right)^{1/4}\exp(-\frac{m\omega}{2\hbar}x_j^2), \\ |1\rangle &= \frac{1}{\sqrt{2}}\left(\hat{x}_j\sqrt{\frac{m\omega}{\hbar}} - \hbar\frac{1}{\sqrt{m\omega\hbar}}\frac{\partial}{\partial x_j}\right)\left(\frac{m\omega}{\pi\hbar}\right)^{1/4}\exp(-\frac{m\omega}{2\hbar}x_j^2), \\ |1\rangle &= \frac{1}{\sqrt{2}}\left(\hat{x}_j\sqrt{\frac{m\omega}{\hbar}} + \frac{m\omega}{\hbar}\hbar\frac{x_j}{\sqrt{m\omega\hbar}}\right)\left(\frac{m\omega}{\pi\hbar}\right)^{1/4}\exp(-\frac{m\omega}{2\hbar}x_j^2), \\ |1\rangle &= \frac{1}{\sqrt{2}}\sqrt{\frac{m\omega}{\hbar}}2\hat{x}_j\left(\frac{m\omega}{\pi\hbar}\right)^{1/4}\exp(-\frac{m\omega}{2\hbar}x_j^2), \end{aligned} \tag{409}$$

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.

22.8 Wigner Transform Propagation based on the MM Hamiltonian

The equation of motion for propagation of the Wigner transform

$$\rho^W(\mathbf{R},\mathbf{P},\mathbf{x},\mathbf{p}) = \int \frac{d\mathbf{S}e^{i2\mathbf{S}\cdot\mathbf{P}}}{\pi} \int \frac{ds_1 e^{i2s_1p_1}}{\pi} \int \frac{ds_2 e^{i2s_2p_2}}{\pi} \psi^*(\mathbf{z}+\mathbf{s})\psi(\mathbf{z}-\mathbf{s}), \tag{410}$$

with $\mathbf{z} = (\mathbf{R}, x_1, x_2)$ and $\mathbf{s} = (\mathbf{S}, s_1, s_2)$ according the Meyer-Miller Hamiltonian,

$$\begin{aligned} \hat{H} &= \frac{\hat{P}^2}{2M} + \frac{1}{2}\sum_{j=1}^{2}\sum_{k=1}^{2} H_{jk}(R)(\tilde{x}_j\tilde{x}_k + \tilde{p}_j\tilde{p}_k - \delta_{jk}), \\ &= \frac{\hat{P}^2}{2M} + \frac{1}{2}\sum_{j=1}^{2}\sum_{k=1}^{2} \hbar H_{jk}(R)(x_jx_k + p_jp_k - \delta_{jk}) \end{aligned} \tag{411}$$

can be obtained as follows:

$$\frac{\partial\rho^W}{\partial t} = \int \frac{d\mathbf{S}e^{i2\mathbf{S}\cdot\mathbf{P}}}{\pi} \int \frac{d\mathbf{s}e^{i2\mathbf{s}\cdot\mathbf{p}}}{\pi} \left(\frac{\partial\psi^*(\mathbf{z}+\mathbf{s})}{\partial t}\psi(\mathbf{z}-\mathbf{s}) + \psi^*(\mathbf{z}+\mathbf{s})\frac{\partial\psi(\mathbf{z}-\mathbf{s})}{\partial t} \right). \tag{412}$$

Considering that

$$\begin{aligned} i\psi(\mathbf{z}-\mathbf{s})\frac{\partial\psi^*(\mathbf{z}+\mathbf{s})}{\partial t} &= -\frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N} H_{jk}(\mathbf{R}+\mathbf{S})(x_jx_k + p_jp_k - \delta_{jk})\psi^*\psi \\ &- \sum_{i=1}^{N} s_i\frac{\partial}{\partial x_i}\frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N} H_{jk}(\mathbf{R}+\mathbf{S})(x_jx_k - \delta_{jk})\psi^*\psi \\ &- \sum_{i=1}^{N}\sum_{l=1}^{N} s_is_l\frac{\partial}{\partial x_l}\frac{\partial}{\partial x_i}\frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N} H_{jk}(\mathbf{R}+\mathbf{S})(x_jx_k - \delta_{jk})\psi^*\psi \end{aligned} \tag{413}$$

and

$$\begin{aligned} i\psi^*(\mathbf{z}+\mathbf{s})\frac{\partial\psi(\mathbf{z}-\mathbf{s})}{\partial t} &= \frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N} H_{jk}(\mathbf{R}-\mathbf{S})(x_jx_k + p_jp_k - \delta_{jk})\psi\psi^* \\ &- \sum_{i=1}^{N} s_i\frac{\partial}{\partial x_i}\frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N} H_{jk}(\mathbf{R}-\mathbf{S})(x_jx_k - \delta_{jk})\psi\psi^* \\ &+ \sum_{i=1}^{N}\sum_{l=1}^{N} s_is_l\frac{\partial}{\partial x_l}\frac{\partial}{\partial x_i}\frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N} H_{jk}(\mathbf{R}-\mathbf{S})(x_jx_k - \delta_{jk})\psi\psi^* \end{aligned} \tag{414}$$

gives

$$\frac{\partial \rho^W}{\partial t} = \int \frac{d\mathbf{S} e^{i2\mathbf{S}\cdot\mathbf{P}}}{\pi} \int \frac{d\mathbf{s} e^{i2\mathbf{s}\cdot\mathbf{p}}}{\pi} i \sum_{j=1}^{N}\sum_{k=1}^{N} (\frac{H_{jk}}{2} + \frac{H''_{jk}}{4}\mathbf{S}^2) \left[\psi p_j p_k \psi^* - \psi^* p_j p_k \psi + (s_j x_k + s_k x_j)\psi\psi^*\right]$$
$$+ \int \frac{d\mathbf{S} e^{i2\mathbf{S}\cdot\mathbf{P}}}{\pi} \int \frac{d\mathbf{s} e^{i2\mathbf{s}\cdot\mathbf{p}}}{\pi} i \sum_{j=1}^{N}\sum_{k=1}^{N} \frac{H'_{jk}}{2}\mathbf{S} \left[2\psi(x_j x_k - \delta_{jk})\psi^* + \psi p_j p_k \psi^* + \psi^* p_j p_k \psi + 2\psi^* s_j s_k \psi\right], \tag{415}$$

where the first bracket gives

$$\begin{aligned}
[\cdots] &= \left[-4\left(\psi \frac{\partial}{\partial s_j}\frac{\partial}{\partial s_k}\psi^* - \psi^* \frac{\partial}{\partial s_j}\frac{\partial}{\partial s_k}\psi\right) + (s_j x_k + s_k x_j)\psi\psi^*\right], \\
&= \left[-4\frac{\partial}{\partial s_j}\left(\psi \frac{\partial}{\partial s_k}\psi^* - \psi^* \frac{\partial}{\partial s_k}\psi\right) + (s_j x_k + s_k x_j)\psi\psi^*\right], \\
&= \left[-2\frac{\partial}{\partial s_j}\left(\psi \frac{\partial}{\partial x_k}\psi^* + \psi^* \frac{\partial}{\partial x_k}\psi\right) + (s_j x_k + s_k x_j)\psi\psi^*\right], \\
&= \left[-2\frac{\partial}{\partial s_j}\frac{\partial}{\partial x_k}(\psi\psi^*) + (s_j x_k + s_k x_j)\psi\psi^*\right],
\end{aligned} \tag{416}$$

$$\begin{aligned}
\frac{\partial \rho^W}{\partial t} &= -\frac{\partial H}{\partial P}\frac{\partial \rho^W}{\partial R} + \sum_{j=1}^{N}\sum_{k=1}^{N}(H_{jk} - \frac{\hbar^2}{2}H''_{jk}\frac{\partial^2}{\partial P^2})\left[-p_j\frac{\partial \rho^W}{\partial x_k} + \frac{\partial \rho^W}{\partial p_k}x_j\right] + \cdots, \\
&= -\frac{\partial H}{\partial P}\frac{\partial \rho^W}{\partial R} + \sum_{k=1}^{N} -\frac{\partial H}{\partial p_k}\frac{\partial \rho^W}{\partial x_k} + \frac{\partial \rho^W}{\partial p_k}\frac{\partial H}{\partial x_k} - \sum_{j=1}^{N}\sum_{k=1}^{N}\frac{\hbar^2}{2}H''_{jk}\frac{\partial^2}{\partial P^2}\left[-p_j\frac{\partial \rho^W}{\partial x_k} + \frac{\partial \rho^W}{\partial p_k}x_j\right] + \cdots,
\end{aligned} \tag{417}$$

22.9 Wigner Transform Propagation for the electronic MM Hamiltonian

The equation of motion for the propagation of the Wigner transform

$$\rho^W(\mathbf{x},\mathbf{p}) = \int \frac{ds_1 e^{is_1 p_1}}{2\pi} \cdots \int \frac{ds_N e^{is_N p_N}}{2\pi} \psi^*(\mathbf{x} + \mathbf{s}/2)\psi(\mathbf{x} - \mathbf{s}/2), \tag{418}$$

according the Meyer-Miller Hamiltonian without nuclear coordinates,

$$\begin{aligned}
\hat{H} &= \frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N} H_{jk}(\tilde{x}_j\tilde{x}_k + \tilde{p}_j\tilde{p}_k - \delta_{jk}), \\
&= \frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N} \hbar H_{jk}(x_j x_k + p_j p_k - \delta_{jk})
\end{aligned} \tag{419}$$

can be obtained as follows:

$$\frac{\partial \rho^W}{\partial t} = \int \frac{ds_1 e^{is_1 p_1}}{2\pi} \cdots \int \frac{ds_N e^{is_N p_N}}{2\pi} \left(\frac{\partial \psi^*(\mathbf{x}+\mathbf{s}/2)}{\partial t} \psi(\mathbf{x}-\mathbf{s}/2) + \psi^*(\mathbf{x}+\mathbf{s}/2) \frac{\partial \psi(\mathbf{x}-\mathbf{s}/2)}{\partial t} \right). \tag{420}$$

Considering that

$$\begin{aligned} i\psi(\mathbf{x}-\mathbf{s}/2)\frac{\partial \psi^*(\mathbf{x}+\mathbf{s}/2)}{\partial t} &= -\frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N} H_{jk}(x_j x_k + p_j p_k - \delta_{jk})\psi^*\psi \\ &- \sum_{i=1}^{N} \frac{s_i}{2}\frac{\partial}{\partial x_i}\frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N} H_{jk}(x_j x_k - \delta_{jk})\psi^*\psi \\ &- \sum_{i=1}^{N}\sum_{l=1}^{N} \frac{s_i}{2}\frac{s_l}{2}\frac{\partial}{\partial x_l}\frac{\partial}{\partial x_i}\frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N} H_{jk}(x_j x_k - \delta_{jk})\psi^*\psi \end{aligned} \tag{421}$$

and

$$\begin{aligned} i\psi^*(\mathbf{x}+\mathbf{s}/2)\frac{\partial \psi(\mathbf{x}-\mathbf{s}/2)}{\partial t} &= \frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N} H_{jk}(x_j x_k + p_j p_k - \delta_{jk})\psi\psi^* \\ &- \sum_{i=1}^{N} \frac{s_i}{2}\frac{\partial}{\partial x_i}\frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N} H_{jk}(x_j x_k - \delta_{jk})\psi\psi^* \\ &+ \sum_{i=1}^{N}\sum_{l=1}^{N} \frac{s_i}{2}\frac{s_l}{2}\frac{\partial}{\partial x_l}\frac{\partial}{\partial x_i}\frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N} H_{jk}(x_j x_k - \delta_{jk})\psi\psi^* \end{aligned} \tag{422}$$

gives

$$\frac{\partial \rho^W}{\partial t} = \int \frac{ds_1 e^{is_1 p_1}}{2\pi} \cdots \int \frac{ds_N e^{is_N p_N}}{2\pi} i \sum_{j=1}^{N}\sum_{k=1}^{N} \frac{H_{jk}}{2} \left[\psi p_j p_k \psi^* - \psi^* p_j p_k \psi + (s_j x_k + s_k x_j)\psi\psi^* \right], \tag{423}$$

where

$$\begin{aligned} [\cdots] &= \left[-4\left(\psi \frac{\partial}{\partial s_j}\frac{\partial}{\partial s_k}\psi^* - \psi^* \frac{\partial}{\partial s_j}\frac{\partial}{\partial s_k}\psi \right) + (s_j x_k + s_k x_j)\psi\psi^* \right], \\ &= \left[-4\frac{\partial}{\partial s_j}\left(\psi \frac{\partial}{\partial s_k}\psi^* - \psi^* \frac{\partial}{\partial s_k}\psi \right) + (s_j x_k + s_k x_j)\psi\psi^* \right], \\ &= \left[-2\frac{\partial}{\partial s_j}\left(\psi \frac{\partial}{\partial x_k}\psi^* + \psi^* \frac{\partial}{\partial x_k}\psi \right) + (s_j x_k + s_k x_j)\psi\psi^* \right], \\ &= \left[-2\frac{\partial}{\partial s_j}\frac{\partial}{\partial x_k}(\psi\psi^*) + (s_j x_k + s_k x_j)\psi\psi^* \right], \\ &= \left[-2p_j \frac{\partial}{\partial x_k}\rho^W + (\frac{\partial}{\partial p_j}x_k + \frac{\partial}{\partial p_k}x_j)\rho^W \right], \end{aligned} \tag{424}$$

$$\begin{aligned}\frac{\partial \rho^W}{\partial t} &= \sum_{j=1}^{N}\sum_{k=1}^{N} H_{jk}\left[-p_j\frac{\partial \rho^W}{\partial x_k} + \frac{\partial \rho^W}{\partial p_k}x_j\right], \\ &= \sum_{k=1}^{N} -\frac{\partial H}{\partial p_k}\frac{\partial \rho^W}{\partial x_k} + \frac{\partial \rho^W}{\partial p_k}\frac{\partial H}{\partial x_k}.\end{aligned} \tag{425}$$

Equation (425) 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:

$$P_k(t) = (2\pi\hbar)^N \int d\mathbf{p} \int d\mathbf{x} \left(\rho_{kk}^W(\mathbf{x},\mathbf{p})\right)^* \rho^W(\mathbf{x},\mathbf{p};t), \tag{426}$$

where

$$\begin{aligned}\rho_{kk}^W(\mathbf{x},\mathbf{p}) &= \frac{1}{(2\pi\hbar)^N}\int d\mathbf{s}' e^{\frac{i}{\hbar}\mathbf{p}\cdot\mathbf{s}'}\langle \mathbf{x}-\frac{\mathbf{s}'}{2}|k\rangle\langle k|\mathbf{x}+\frac{\mathbf{s}'}{2}\rangle, \\ \rho^W(\mathbf{x},\mathbf{p};t) &= \frac{1}{(2\pi\hbar)^N}\int d\mathbf{s} e^{\frac{i}{\hbar}\mathbf{p}\cdot\mathbf{s}}\langle \mathbf{x}-\frac{\mathbf{s}}{2}|\Psi(t)\rangle\langle\Psi(t)|\mathbf{x}+\frac{\mathbf{s}}{2}\rangle, \\ &= \frac{1}{(2\pi\hbar)^N}\sum_j\sum_{j'} c_j^* c_{j'}\int d\mathbf{s} e^{\frac{i}{\hbar}\mathbf{p}\cdot\mathbf{s}}\langle \mathbf{x}-\frac{\mathbf{s}}{2}|j'\rangle\langle j|\mathbf{x}+\frac{\mathbf{s}}{2}\rangle.\end{aligned} \tag{427}$$

We note that

$$\rho_{00}^W = \frac{1}{(2\pi\hbar)^N}\int d\mathbf{s} e^{\frac{i}{\hbar}\mathbf{p}\cdot\mathbf{s}}\langle \mathbf{x}-\frac{\mathbf{s}}{2}|0\rangle\langle 0|\mathbf{x}+\frac{\mathbf{s}}{2}\rangle, \tag{428}$$

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^W_{kk} &= \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}^\dagger_k|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{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{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{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{429}$$

Note that by substituting Eq. (427) into Eq. (426), we obtain:

$$\begin{aligned}
P_k(t) = (2\pi\hbar)^N\sum_j\sum_{j'}c_j^* c_{j'}\frac{1}{(2\pi\hbar)^N}\int d\mathbf{s}\int d\mathbf{p}\int d\mathbf{x}\frac{1}{(2\pi\hbar)^N}\int d\mathbf{s}' e^{\frac{i}{\hbar}\mathbf{p}(\mathbf{s}-\mathbf{s}')} \\
\times\langle\mathbf{x}+\frac{\mathbf{s}'}{2}|k\rangle\langle k|\mathbf{x}-\frac{\mathbf{s}'}{2}\rangle\langle\mathbf{x}-\frac{\mathbf{s}}{2}|j'\rangle\langle j|\mathbf{x}+\frac{\mathbf{s}}{2}\rangle.
\end{aligned} \tag{430}$$

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

$$\begin{aligned}
P_k(t) &= \sum_j\sum_{j'}c_j^* c_{j'}\int d\mathbf{s}\int d\mathbf{x}\int d\mathbf{s}'\delta(\mathbf{s}-\mathbf{s}')\langle\mathbf{x}+\frac{\mathbf{s}'}{2}|k\rangle\langle k|\mathbf{x}-\frac{\mathbf{s}'}{2}\rangle\langle\mathbf{x}-\frac{\mathbf{s}}{2}|j'\rangle\langle j|\mathbf{x}+\frac{\mathbf{s}}{2}\rangle, \\
&= \sum_j\sum_{j'}c_j^* c_{j'}\int d\mathbf{s}\int d\mathbf{x}\langle\mathbf{x}+\frac{\mathbf{s}}{2}|k\rangle\langle k|\mathbf{x}-\frac{\mathbf{s}}{2}\rangle\langle\mathbf{x}-\frac{\mathbf{s}}{2}|j'\rangle\langle j|\mathbf{x}+\frac{\mathbf{s}}{2}\rangle.
\end{aligned} \tag{431}$$

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

$$P_k(t) = \sum_j\sum_{j'}c_j^* c_{j'}\int d\mathbf{s}\int d\mathbf{r}\langle\mathbf{r}+\mathbf{s}|k\rangle\langle k|\mathbf{r}\rangle\langle\mathbf{r}|j'\rangle\langle j|\mathbf{r}+\mathbf{s}\rangle, \tag{432}$$

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

$$\begin{aligned} P_k(t) &= \sum_j \sum_{j'} c_j^* c_{j'} \int d\mathbf{r}' \int d\mathbf{r} \langle \mathbf{r}'|k\rangle\langle k|\mathbf{r}\rangle\langle \mathbf{r}|j'\rangle\langle j|\mathbf{r}'\rangle, \\ &= \sum_j \sum_{j'} c_j^* c_{j'} \int d\mathbf{r} \langle k|\mathbf{r}\rangle\langle \mathbf{r}|j'\rangle \int d\mathbf{r}' \langle j|\mathbf{r}'\rangle\langle \mathbf{r}'|k\rangle, \\ &= \sum_j \sum_{j'} c_j^* c_{j'} \langle k|j'\rangle\langle j|k\rangle, \\ &= \sum_j \sum_{j'} c_j^* c_{j'} \delta_{k,j'}\delta_{j,k}, \\ &= c_k^* c_k, \end{aligned} \tag{433}$$

which proves Eq. (426).

22.10 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. (419), with $H_{11} = -H_{22} = H_{12} = H_{21} = 1.0$, initialized in one of the 2 states according to Eq. (429) and evolving by Velocity Verlet according to Eq. (425). Compare the Rabi oscillations of the time-dependent survival probability to the corresponding results obtained by SOFT quantum propagation.

Solutions in Sec. 40.21

23 Spin-Boson Model

Consider the spin-boson Hamiltonian

$$\hat{H} = \frac{2}{\hbar}\left[\epsilon \hat{S}_z + J\hat{S}_x + \hat{S}_z \times f(\mathbf{y})\right] + H_b(\mathbf{y}, \mathbf{p}_y), \tag{434}$$

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:

$$H = \begin{bmatrix} \epsilon & 0 \\ 0 & -\epsilon \end{bmatrix} + \begin{bmatrix} 0 & J \\ J & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \times f(\mathbf{y}) + H_b(\mathbf{y}, \mathbf{p}_y), \tag{435}$$

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:

$$\hat{S}_z|S, m_s\rangle = \hbar m_s |S, m_s\rangle. \tag{436}$$

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: $[\hat{S}_z, \hat{S}^+] = \hbar\hat{S}^+$, $[\hat{S}_z, \hat{S}^-] = -\hbar\hat{S}^-$, $[\hat{S}^+, \hat{S}^-] = 2\hbar\hat{S}_z$.

Matrix elements in the Zeeman basis are defined, as usual:

$$\begin{aligned}
\langle m'_s|\hat{S}_x|m_s\rangle &= \hbar(\delta_{m'_s,m_s+1}+\delta_{m'_s+1,m_s})\frac{1}{2}\sqrt{S(S+1)-m'_s m_s}\\
\langle m'_s|\hat{S}_y|m_s\rangle &= \hbar(\delta_{m'_s,m_s+1}-\delta_{m'_s+1,m_s})\frac{1}{2i}\sqrt{S(S+1)-m'_s m_s}\\
\langle m'_s|\hat{S}_z|m_s\rangle &= \hbar\delta_{m'_s,m_s}m_s\\
\langle m'_s|\hat{S}^+|m_s\rangle &= \hbar\delta_{m'_s,m_s+1}\sqrt{S(S+1)-m'_s m_s}\\
\langle m'_s|\hat{S}^-|m_s\rangle &= \hbar\delta_{m'_s+1,m_s}\sqrt{S(S+1)-m'_s m_s}\\
\langle m'_s|\hat{S}^2|m_s\rangle &= \hbar\delta_{m'_s,m_s}S(S+1)
\end{aligned} \tag{437}$$

so the same Hamiltonian introduced by Eq. (434) 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

$$\begin{aligned}
S^+ &= \hbar\begin{bmatrix}0&\sqrt{5}&0&0&0&0\\0&0&\sqrt{8}&0&0&0\\0&0&0&\sqrt{9}&0&0\\0&0&0&0&\sqrt{8}&0\\0&0&0&0&0&\sqrt{5}\\0&0&0&0&0&0\end{bmatrix},\\
S^- &= \hbar\begin{bmatrix}0&0&0&0&0&0\\\sqrt{5}&0&0&0&0&0\\0&\sqrt{8}&0&0&0&0\\0&0&\sqrt{9}&0&0&0\\0&0&0&\sqrt{8}&0&0\\0&0&0&0&\sqrt{5}&0\end{bmatrix}
\end{aligned} \tag{438}$$

and

$$S_x = \frac{\hbar}{2}\begin{bmatrix} 0 & \sqrt{5} & 0 & 0 & 0 & 0 \\ \sqrt{5} & 0 & \sqrt{8} & 0 & 0 & 0 \\ 0 & \sqrt{8} & 0 & \sqrt{9} & 0 & 0 \\ 0 & 0 & \sqrt{9} & 0 & \sqrt{8} & 0 \\ 0 & 0 & 0 & \sqrt{8} & 0 & \sqrt{5} \\ 0 & 0 & 0 & 0 & \sqrt{5} & 0 \end{bmatrix},$$

$$S_y = \frac{\hbar}{2i}\begin{bmatrix} 0 & \sqrt{5} & 0 & 0 & 0 & 0 \\ -\sqrt{5} & 0 & \sqrt{8} & 0 & 0 & 0 \\ 0 & -\sqrt{8} & 0 & \sqrt{9} & 0 & 0 \\ 0 & 0 & -\sqrt{9} & 0 & \sqrt{8} & 0 \\ 0 & 0 & 0 & -\sqrt{8} & 0 & \sqrt{5} \\ 0 & 0 & 0 & 0 & -\sqrt{5} & 0 \end{bmatrix}, \tag{439}$$

$$S_z = \frac{\hbar}{2}\begin{bmatrix} 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -3 & 0 \\ 0 & 0 & 0 & 0 & 0 & -5 \end{bmatrix},$$

According to the second quantization mapping, the spin-boson Hamiltonian can be written as follows:

$$\hat{H} = H_b(\mathbf{y}, \mathbf{p}_y) + \sum_{m_s, m_s'} \langle m_s'|(\epsilon + f(\mathbf{y}))\frac{2}{\hbar}\hat{S}_z + J\frac{2}{\hbar}\hat{S}_x|m_s\rangle \hat{a}_{m_s'}^{\dagger}\hat{a}_{m_s}. \tag{440}$$

and in Cartesian coordinates,

$$\begin{aligned} \hat{H} = H_b(\mathbf{y}, \mathbf{p}_y) &+ \sum_{m_s} \frac{2}{\hbar}(\epsilon + f(\mathbf{y}))\langle m_s|\hat{S}_z|m_s\rangle[\tilde{x}_{m_s}^2 + \tilde{p}_{m_s}^2 - \hbar] \\ &+ \sum_{m_s}\sum_{m_s' \neq m_s} \frac{2}{\hbar}\langle m_s|\hat{S}_x|m_s'\rangle J\left[\tilde{x}_{m_s}\tilde{x}_{m_s'} + \tilde{p}_{m_s}\tilde{p}_{m_s'}\right] \end{aligned} \tag{441}$$

24 Holstein-Primakoff Mapping

According to the Holstein-Primakoff (HP) transformation, we re-write the spin operators in terms of boson operators, as follows:

$$\hat{S}^{+} = \hbar\sqrt{2S - \hat{N}}\hat{a}, \tag{442}$$

and

$$\hat{S}^{-} = \hat{a}^{\dagger}\hbar\sqrt{2S - \hat{N}}, \tag{443}$$

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,

$$\hat{S}_z = \hbar(S - \hat{N}), \tag{444}$$

and

$$\hat{S}_x = \frac{1}{2}\left[\hat{S}^+ + \hat{S}^-\right] = \frac{\hbar}{2}\left[\sqrt{2S - \hat{N}}\hat{a} + \hat{a}^\dagger\sqrt{2S - \hat{N}}\right], \tag{445}$$

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:

$$\tilde{x} - i\tilde{p} = \sqrt{2}\hat{a}^\dagger, \tag{446}$$

and

$$\tilde{x} + i\tilde{p} = \sqrt{2}\hat{a}. \tag{447}$$

Therefore,

$$\begin{aligned}\hat{N} &= \frac{1}{2}(\tilde{x} - i\tilde{p})(\tilde{x} + i\tilde{p}) \\ &= \frac{1}{2}(\tilde{x}^2 + i[\tilde{x}, \tilde{p}] + \tilde{p}^2) \\ &= \frac{1}{2}(\tilde{x}^2 + \tilde{p}^2 - \hbar)\end{aligned} \tag{448}$$

Therefore, $S_z = \hbar(S - \frac{1}{2}(\tilde{x}^2 + \tilde{p}^2 - \hbar))$. Analogously, we obtain:

$$\hat{S}_x = \frac{\hbar}{2}\left[\sqrt{S - \frac{1}{4}(\tilde{x}^2 + \tilde{p}^2 - \hbar)}(\tilde{x} + i\tilde{p}) + (\tilde{x} - i\tilde{p})\sqrt{S - \frac{1}{4}(\tilde{x}^2 + \tilde{p}^2 - \hbar)}\right]. \tag{449}$$

and

$$\begin{aligned}\hat{H} &= H_b(\mathbf{y}, \mathbf{p}_y) + (\epsilon + f(\mathbf{y}))(2S - (\tilde{x}^2 + \tilde{p}^2 - \hbar)) \\ &+ J\left[\sqrt{S - \frac{1}{4}(\tilde{x}^2 + \tilde{p}^2 - \hbar)}(\tilde{x} + i\tilde{p}) + (\tilde{x} - i\tilde{p})\sqrt{S - \frac{1}{4}(\tilde{x}^2 + \tilde{p}^2 - \hbar)}\right]\end{aligned} \tag{450}$$

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:

$$\hat{H} = H_b(\mathbf{y}, \mathbf{p}_y) + (\epsilon + f(\mathbf{y}))(2S - (\tilde{x}^2 + \tilde{p}^2 - \hbar)) + J\tilde{x}\sqrt{4S - (\tilde{x}^2 + \tilde{p}^2 - \hbar)} \tag{451}$$

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:

$$\begin{aligned}\sqrt{S - \frac{\hat{N}}{2}} &= \sqrt{S} + \frac{1}{2}(2S)^{-1/2}(-\frac{\hat{N}}{2}) + \cdots \\ &\approx \sqrt{S}\end{aligned} \tag{452}$$

Therefore, truncating the expansion introduced by Eq. (452) after the first term and substituting into Eq. (450), we obtain the 'simple spin-wave (SW) theory' expression:

$$\hat{H} = H_b(\mathbf{y}, \mathbf{p}_y) + (\epsilon + f(\mathbf{y}))(2S - (\tilde{x}^2 + \tilde{p}^2 - \hbar)) + 2J\sqrt{S}\tilde{x} \tag{453}$$

25 Schwinger Mapping

Spin operators can also be mapped according to the Schwinger-boson representation, in terms of pairs of (constrained) bosons, as follows:

$$\begin{aligned}\hat{S}^{+} &= \hbar\hat{b}^{\dagger}\hat{a}, \\ \hat{S}^{-} &= \hbar\hat{a}^{\dagger}\hat{b}, \\ \hat{S}_z &= \frac{1}{2}\left[\hat{S}^{+}, \hat{S}^{-}\right] = \frac{\hbar}{2}\left(\hat{b}^{\dagger}\hat{b} - \hat{a}^{\dagger}\hat{a}\right), \\ \hat{S}_x &= \frac{1}{2}\left(\hat{S}^{+} + \hat{S}^{-}\right) = \frac{\hbar}{2}\left(\hat{b}^{\dagger}\hat{a} + \hat{a}^{\dagger}\hat{b}\right), \\ \hat{S}_y &= \frac{1}{2i}\left(\hat{S}^{+} - \hat{S}^{-}\right) = \frac{\hbar}{2i}\left(\hat{b}^{\dagger}\hat{a} - \hat{a}^{\dagger}\hat{b}\right),\end{aligned} \tag{454}$$

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

$$S = \frac{1}{2}(\hat{b}^{\dagger}\hat{b} + \hat{a}^{\dagger}\hat{a}). \tag{455}$$

A specific form of the constraint, introduced by Eq. (455), is when $\hat{b}^{\dagger} = b = \sqrt{2S - \hat{a}^{\dagger}\hat{a}}$, which makes the Schwinger mapping identical to the Holstein-Primakoff transformation, defined by Eqs. (442)–(445).

Substituting the spin operators in Eq. (434), according to the Schwinger mapping, introduced by Eqs. (454), we obtain:

$$\hat{H} = (\epsilon + f(\mathbf{y}))\left(\hat{b}^{\dagger}\hat{b} - \hat{a}^{\dagger}\hat{a}\right) + J\left(\hat{b}^{\dagger}\hat{a} + \hat{a}^{\dagger}\hat{b}\right) + H_b(\mathbf{y}, \mathbf{p}_y), \tag{456}$$

Changing variables, $\hat{b} = \hat{b}_{\nu_1}$ and $\hat{a} = \hat{b}_{\nu_2}$, defined in terms of the Cartesian coordinates, as follows:

$$\hat{b}^{\dagger}_{\nu_j} = \frac{1}{\sqrt{2}}\left[\tilde{x}_{\nu_j} - i\tilde{p}_{\nu_j}\right], \tag{457}$$

and

$$\hat{b}_{\nu_j} = \frac{1}{\sqrt{2}}\left[\tilde{x}_{\nu_j} + i\tilde{p}_{\nu_j}\right], \tag{458}$$

with

$$\hat{n}_{\nu_j} = \frac{1}{2}(\tilde{x}^2_{\nu_j} + \tilde{p}^2_{\nu_j} - \hbar) \tag{459}$$

we obtain

$$\hat{H} = H_b(\mathbf{y}, \mathbf{p}_y) + (\epsilon + f(\mathbf{y}))\left[(\tilde{x}^2_{\nu_1} + \tilde{p}^2_{\nu_1} - \hbar) - (\tilde{x}^2_{\nu_2} + \tilde{p}^2_{\nu_2} - \hbar)\right] + J\left(\tilde{x}_{\nu_1}\tilde{x}_{\nu_2} + \tilde{p}_{\nu_1}\tilde{p}_{\nu_2}\right), \tag{460}$$

which is the same Hamiltonian given by Eq. (441), according to the second quantization mapping, although with the contraint

$$S = \frac{1}{4}\left[(\tilde{x}^2_{\nu_1} + \tilde{p}^2_{\nu_1} - \hbar) + (\tilde{x}^2_{\nu_2} + \tilde{p}^2_{\nu_2} - \hbar)\right]. \tag{461}$$

26 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:

$$\tilde{\rho}(\mathbf{x}, \mathbf{x}'; \epsilon) \approx \sum_{j=1}^{n} c_j \phi_j(\mathbf{x})[\phi'_j(\mathbf{x}')]^*, \tag{462}$$

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

$$\phi_j(\mathbf{x}) \equiv \prod_{k=1}^{N} A_{\phi_j}(k) e^{-\gamma_{\phi_j}(k)\left(x(k) - x_{\phi_j}(k)\right)^2/2}\, e^{i\, p_{\phi_j}(k)\left(x(k) - x_{\phi_j}(k)\right)}, \tag{463}$$

  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. (462), are defined as follows:

$$c_j \equiv \begin{cases} I_j, & \text{when} \quad j = 1, \\ I_j - \sum_{k=1}^{j-1} c_k \langle \phi_j | \phi_k \rangle \langle \phi'_k | \phi'_j \rangle, & \text{for} \quad j = 2 - n, \end{cases} \tag{464}$$

  where the overlap integral I_j is defined as follows,

$$I_j \equiv \int d\mathbf{x}' d\mathbf{x}\; \phi_j(\mathbf{x})\tilde{\rho}(\mathbf{x}, \mathbf{x}'; \epsilon)[\phi'_j(\mathbf{x}')]^*. \tag{465}$$

- 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:

$$\rho(\mathbf{x},\mathbf{x}';\epsilon+i\tau)=\sum_{j=1}^{n}c_j e^{-iV_0(\mathbf{x})\tau/2}\widetilde{\phi}_j(\mathbf{x})[\phi'_j(\mathbf{x}')]^*, \tag{466}$$

where

$$\widetilde{\phi}_j(\mathbf{x})\equiv\prod_{k=1}^{N}A_{\widetilde{\phi}_j}(k)\sqrt{\frac{m}{m+i\tau\gamma_{\widetilde{\phi}_j}(k)}}\exp\left(\frac{\left(\frac{p_{\widetilde{\phi}_j}(k)}{\gamma_{\widetilde{\phi}_j}(k)}-i(x_{\widetilde{\phi}_j}(k)-x(k))\right)^2}{\left(\frac{2}{\gamma_{\widetilde{\phi}_j}(k)}+\frac{i2\tau}{m}\right)}-\frac{p_{\widetilde{\phi}_j}(k)^2}{2\gamma_{\widetilde{\phi}_j}(k)}\right). \tag{467}$$

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. (462).

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. (465). 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. (462) 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. (465). 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.*, $\parallel\varepsilon_1\parallel<\parallel\tilde{\rho}\parallel$.

- 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

$$c_2\equiv\int d\mathbf{x}'d\mathbf{x}\;\phi_2(\mathbf{x})\varepsilon_1(\mathbf{x},\mathbf{x}')[\phi'_2(\mathbf{x}')]^*. \tag{468}$$

  Note that $\parallel\varepsilon_2\parallel<\parallel\varepsilon_1\parallel$, 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.*, $\parallel\varepsilon_n\parallel=(1-\sum_{j=1}^{n}|c_j|^2)^{1/2}<\epsilon$, and the resulting expansion is given by Eq. (462). 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. (463) and (467), 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.

27 Exam 1

Exam 1 CHEM 572a

Advanced Quantum Mechanics

Exercise 1:
(**10 points**) **1.1**: Explain the fundamental principles of Quantum Mechanics.
(**10 points**) **1.2**: Prove that if two eigenfunctions of a hermitian operator have different eigenvalues then they are orthogonal to each other.
(**10 points**) **1.3**: Prove that the Boltzmann operator $exp(-\beta\hat{H})$ commutes with $\hat{H}$.
(**10 points**) **1.4**: Prove that the momentum operator is Hermitian.
(**10 points**) **1.5**: Prove that

$$\langle\Phi_k|\dot{\Phi}_n\rangle = \frac{\langle\Phi_k|\frac{\partial\hat{H}}{\partial t}|\Phi_n\rangle}{E_k - E_n},$$

where Φ_k and Φ_n are eigenfunctions of $\hat{H}$ with eigenvalues E_k and E_n, respectively.
Exercise 2:
2.1. (20 points) Explain how to implement the Split Operator Fourier transform method to integrate the time-dependent Schrödinger equation and propagate the initial state $|\Psi_0\rangle$ in a digital grid-based representation when evolving according to the Hamiltonian

$$\hat{H} = \frac{\hat{p}^2}{2m} + V(\hat{x}) + \epsilon(t).$$

2.2. (20 points) Having obtained the time-evolved wave function at $|\Psi_t\rangle$ in a grid-based representation, as described in item 2.1, explain how to compute the quantum expectation value of the energy, $E_t = \langle\Psi_t|\hat{H}|\Psi_t\rangle$.
2.3. (10 points) Explain how to implement the Velocity-Verlet algorithm and compute the classical time-dependent value of the energy, analogous to the quantum mechanical expression $E_t = \langle\Psi_t|\hat{H}|\Psi_t\rangle$.

27.1 Answer Key

Exercise 1:
(**10 points**) **Exercise 1.1**: See lecture notes (pages 3–5).

$\boxed{Postulate\ 1}$: *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.

$\boxed{Postulate\ 3}$**:** 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 following equation*:

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

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, $E = \frac{p^2}{2m} + V(x)$.

$\boxed{Expansion\ Postulate}$**:**
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*.
(**10 points**) **Exercise 1.2**: See lecture notes (page 4).
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 $\int dx\phi_m^*\phi_n = 0$.
(**10 points**) **Exercise 1.3**: Note that

$$exp(-\beta\hat{H}) = 1 - \beta\hat{H} + \frac{1}{2}\beta^2\hat{H}^2 + ...$$

where each term in the expansion commutes with $\hat{H}$ since it is a power of $\hat{H}$ and $\hat{H}^n\hat{H} = \hat{H}\hat{H}^n$. Therefore, the complete expansion commutes with $\hat{H}$.
(**10 points**) **Exercise 1.4**: In order to prove that the momentum operator is Hermitian, we integrate $\langle\phi_j|\hat{p}|\phi_k\rangle$ by parts as follows:

$$-i\hbar\int_{-\infty}^{\infty} dx\phi_j^*(x)\frac{\partial\phi_k(x)}{\partial x} = -i\hbar\left(\phi_j^*(x)\phi_k(x)|_{-\infty}^{\infty} - \int_{-\infty}^{\infty} dx\phi_k(x)\frac{\partial}{\partial x}\phi_j^*(x)\right),$$

where the first term in the right hand side of the equation above is zero since (according to postulate 1) $\phi_j(x)$ and $\phi_k(x)$ must be square integrable and therefore must vanish at $x = \pm\infty$. Hence, $\langle\phi_j|\hat{p}|\phi_k\rangle$= $\langle\phi_k|\hat{p}|\phi_j\rangle^*$.
(**10 points**) **Exercise 1.5**: Considering that Φ_j and Φ_n are eigenfunctions of $\hat{H}$ with eigenvalues E_j and E_n, respectively, we obtain:

$$\frac{\partial}{\partial t}\langle\Phi_j|\hat{H}|\Phi_n\rangle = 0,$$

since $\hat{H}|\Phi_n\rangle = E_n|\Phi_n\rangle$ and $\langle\Phi_j|\Phi_n\rangle = 0$, when $j \neq n$.

Therefore,

$$\int dx\left(\frac{\partial\Phi_j^*}{\partial t}\hat{H}\Phi_n + \Phi_j^*\frac{\partial\hat{H}}{\partial t}\Phi_n + \Phi_j^*\hat{H}\frac{\partial\Phi_n}{\partial t}\right) = 0.$$

Considering that $\hat{H}$ is hermitian, we obtain:

$$E_n\langle\dot{\Phi}_j|\Phi_n\rangle + \langle\Phi_j|\frac{\partial\hat{H}}{\partial t}|\Phi_n\rangle + E_j\langle\Phi_j|\dot{\Phi}_n\rangle = 0.$$

Finally, note that $\langle\dot{\Phi}_j|\Phi_n\rangle = -\langle\Phi_j|\dot{\Phi}_n\rangle$ since $\frac{\partial}{\partial t}\langle\Phi_k|\Phi_n\rangle = 0$. Thus,

$$(E_n - E_j)\langle\dot{\Phi}_j|\Phi_n\rangle + \langle\Phi_j|\frac{\partial\hat{H}}{\partial t}|\Phi_n\rangle = 0.$$

Exercise 2:
(20 points) Exercise 2.1. See lecture notes (pages 14 and 15). 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 $e^{-i(V(x')+\epsilon(t_k))\tau/2}$ as arrays of numbers $\Psi_{t_k}(x_j)$ and $e^{-i(V(x_j)+\epsilon(t_k))\tau/2}$ 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 $e^{-i(V(x')+\epsilon(t_k))\tau/2}$ to $\Psi_{t_k}(x')$ by simple multiplication of array elements:

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

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 $e^{-i(V(x')+\epsilon(t_k))\tau/2}$ to $\widetilde{\Psi}_{t_k}(x')$ by simple multiplication of array elements,

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

Make $t_k = t_{k+1}$ and go to step 1. Loop the process N times, with $N = t/\tau$.
Exercise 2.2. (20 points) Having Ψ_t represented on a grid as a linear combination of delta functions $\Psi_t(x) = \sum_j \Psi_t(x_j)\delta(x - x_j)$, we compute $E_t = \langle\Psi_t|\hat{H}|\Psi_t\rangle$, by first obtaining

$$\tilde{\Psi}_t(x) = \left(\frac{\hat{p}^2}{2m} + V(x)\right)\Psi_t(x),$$

and then computing the internal product $E_t = \langle\Psi_t|\tilde{\Psi}_t\rangle$, as follows:

$$E_t = \sum_j \Delta\Psi_t^*(x_j)\tilde{\Psi}_t(x_j)$$

.

In order to obtain $\tilde{\Psi}_t$, we proceed as follows:

$$\tilde{\Psi}_t(x) = \int_{-\infty}^{\infty} dpdx'\langle x|p\rangle\frac{p^2}{2m}\langle p|x'\rangle\langle x'|\Psi_t\rangle + V(x)\Psi_t(x),$$

where $\langle p|x\rangle = (2\pi\hbar)^{1/2}e^{-ipx/\hbar}$. Therefore,

$$\tilde{\Psi}_t(x) = \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} dp e^{ipx/\hbar}\frac{p^2}{2m}\int_{-\infty}^{\infty} dx' e^{-ipx'/\hbar}\Psi_t(x') + V(x)\Psi_t(x).$$

Exercise 2.3. (10 points) The coordinates and momenta are evolved according to the Velocity-Verlet algorithm, by repeatetly applying the following transformation:

$$\begin{aligned} p_{j+1} &= p_j + (F(x_j) + F(x_{j+1}))\tau/2 \\ x_{j+1} &= x_j + p_j\tau/m + F(x_j)\tau^2/(2m). \end{aligned} \tag{469}$$

After applying the transformation N times, with $N = t/\tau$, we evaluate the energy of the system as follows: $E_t = p_N^2/(2m) + V(x_N)$.

28 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,

$$HC_j - C_j E_j = 0. \tag{470}$$

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:

$$x_j = x_{min} + j\Delta, \text{ with } \Delta = (x_{max} - x_{min})/N, \tag{471}$$

with j = 1–(N-1).

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

$$H(j,j') = V(x_j)\delta_{jj'} + \frac{\hbar^2}{2m\Delta^2}(-1)^{j-j'}\left(\delta_{jj'}\frac{\pi^2}{3} + (1-\delta_{jj'})\frac{2}{(j-j')^2}\right), \tag{472}$$

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:

$$H(j,j') = V(x_j)\delta_{jj'} + \frac{\hbar^2}{2m\Delta^2}(-1)^{j-j'}\left(\delta_{jj'}\left(\frac{\pi^2}{3} - \frac{1}{2j^2}\right) + (1-\delta_{jj'})\left(\frac{2}{(j-j')^2} - \frac{2}{(j+j')^2}\right)\right). \tag{473}$$

To derive Eq. (472) and Eq. (473), we consider the Hamiltonian,

$$\hat{H} = \hat{T} + V(\hat{x}), \tag{474}$$

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:

$$\begin{aligned} V^{(\delta)}(j,k) &= \langle j|V(\hat{x})|k\rangle = \int dx \delta^*(x - x_j)V(\hat{x})\delta(x - x_k), \\ &= V(x_k)\delta_{j,k}. \end{aligned} \tag{475}$$

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:

$$T^{(\delta)} = \Gamma^{-1} T^{(\phi)} \Gamma, \tag{476}$$

where Γ is the transformation matrix defined by the linear combinations,

$$\phi_k(x) = \sum_j \Gamma(j,k)\delta(x - x_j)\Delta', \tag{477}$$

where

$$\Gamma(j,k) = \phi_k(x_j). \tag{478}$$

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:

$$\phi_k(x) = \sqrt{\frac{2}{x_{max} - x_{min}}} \text{Sin}\left(k \frac{\pi(x - x_{min})}{(x_{max} - x_{min})}\right), \tag{479}$$

with $\phi_k(x_{min}) = 0$ and $\phi_k(x_{max}) = 0$. Therefore,

$$\hat{T}\phi_k(x) = \frac{(\hbar\pi k)^2}{2m}\phi_k(x), \tag{480}$$

and $T^{(\phi)}$ is diagonal with matrix elements,

$$\hat{T}^{(\phi)}(j,k) = \langle \phi_j | \hat{T} | \phi_k \rangle = \frac{(\hbar k)^2}{2m} \frac{\pi^2}{(x_{max} - x_{min})^2} \delta_{jk}. \tag{481}$$

Therefore, substituting Eq. (481) and Eq. (478) into Eq. (476) we obtain,

$$\begin{aligned} T^{(\delta)}(i,i') &= \sum_{j,k=1}^{N-1} \Gamma^{-1}(i,j) T^{(\phi)}(j,k) \Gamma(k,i') = \sum_{j,k=1}^{N-1} \Gamma(j,i) T^{(\phi)}(j,k) \Gamma(k,i'), \\ &= \frac{\Delta\pi^2}{(x_{max} - x_{min})^2} \sum_{j,k=1}^{N-1} \phi_j(x_i) \frac{(\hbar k)^2}{2m} \delta_{jk} \phi_k(x_i') = \frac{\Delta\pi^2}{(x_{max} - x_{min})^2} \sum_{k=1}^{N-1} \phi_k(x_i) \frac{(\hbar k)^2}{2m} \phi_k(x_i'), \\ &= \frac{\Delta\hbar^2\pi^2}{2m(x_{max} - x_{min})^2} \frac{2}{(x_{max} - x_{min})} \sum_{k=1}^{N-1} k^2 \text{Sin}\left(k\pi \frac{(x_i - x_{min})}{(x_{max} - x_{min})}\right) \text{Sin}\left(k\pi \frac{(x_{i'} - x_{min})}{(x_{max} - x_{min})}\right). \end{aligned} \tag{482}$$

Finally, substituting Eq. (471) into Eq. (482) we obtain:

$$T^{(\delta)}(j,j') = \frac{\hbar^2\pi^2}{2m(x_{max} - x_{min})^2} \frac{2}{N} \sum_{k=1}^{N-1} k^2 \text{Sin}\left(\frac{k\pi j}{N}\right) \text{Sin}\left(\frac{k\pi j'}{N}\right). \tag{483}$$

To calculate the finite series introduced by Eq. (483) we first note that,

$$\begin{aligned} 2\text{Sin}\left(\frac{k\pi j}{N}\right)\text{Sin}\left(\frac{k\pi j'}{N}\right) &= \text{Cos}\left(\frac{k\pi(j-j')}{N}\right) - \text{Cos}\left(\frac{k\pi(j+j')}{N}\right), \\ &= Re\left[\text{Exp}\left(i\frac{k\pi(j-j')}{N}\right) - \text{Exp}\left(i\frac{k\pi(j+j')}{N}\right)\right]. \end{aligned} \tag{484}$$

so that Eq. (483) can be written as follows:

$$T^{(\delta)}(j,j') = \frac{\hbar^2\pi^2}{2m(x_{max}-x_{min})^2}\frac{2}{N}\left[\text{Re}\sum_{k=1}^{N-1}k^2\text{Exp}\left(i\frac{k\pi(j-j'}{N}\right) - \text{Re}\sum_{k=1}^{N-1}k^2\text{Exp}\left(i\frac{k\pi(j+j')}{N}\right)\right]. \tag{485}$$

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

$$\begin{aligned} x\frac{\partial}{\partial x}\sum_{k=0}^{N-1}x^k &= \sum_{k=0}^{N-1}kx^k, \\ x^2\frac{\partial^2}{\partial x^2}\sum_{k=0}^{N-1}x^k &= \sum_{k=0}^{N-1}k^2x^k - \sum_{k=0}^{N-1}kx^k, \end{aligned} \tag{486}$$

Therefore,

$$\sum_{k=1}^{N-1}k^2x^k = x^2\frac{\partial^2}{\partial x^2}\left(\frac{(1-x^N)}{(1-x)}\right) + x\frac{\partial}{\partial x}\left(\frac{(1-x^N)}{(1-x)}\right). \tag{487}$$

We evaluate the sums over k in Eq. (485) analytically to obtain:

$$T^{(\delta)}(j,j') = \frac{\hbar^2(-1)^{j-j'}}{2m(x_{max}-x_{min})^2}\frac{\pi^2}{2}\left[\frac{1}{\text{Sin}^2[\pi(j-j')/(2N)]} - \frac{1}{\text{Sin}^2[\pi(j+j')/(2N)]}\right], \tag{488}$$

for $j \neq j'$ and

$$T^{(\delta)}(j,j) = \frac{\hbar^2}{2m(x_{max}-x_{min})^2}\frac{\pi^2}{2}\left[\frac{(2N^2+1)}{3} - \frac{1}{\text{Sin}^2[\pi j/N]}\right]. \tag{489}$$

Equation (472) is obtained from Eq. (488) and Eq. (489), 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 (473) is obtained from Eq. (488) and Eq. (489), 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.

28.1 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. (10) 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. (10) 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.

29 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. 9. 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,

$$H_0(x, p) = \frac{p^2}{2} - \alpha(x^2 - \beta x^4), \tag{490}$$

with $\alpha = 1/2^2$ and $\beta = 1/2^5$. In the absence of an external perturbation, the initial non-stationary state

$$\Phi_0(x) = \pi^{-1/4} e^{-(x-x_0)^2/2}, \tag{491}$$

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. (491)),

$$\Phi_0(x) = \frac{1}{\sqrt{2}} \left(\chi_0(x) + \chi_1(x)\right), \tag{492}$$

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 $\hat{H}|\chi_j\rangle = E_j|\chi_j\rangle$,

$$|\Phi_t\rangle = \frac{1}{\sqrt{2}}\left(\chi_0(x)e^{-\frac{i}{\hbar}E_0 t} + \chi_1(x)e^{-\frac{i}{\hbar}E_1 t}\right), \tag{493}$$

and

$$|\xi(t)|^2 = |\langle\Phi_0|\Phi_t\rangle|^2 = \frac{1}{2} + \frac{1}{2}cos(\Omega t), \tag{494}$$

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. (494) and (494), is that tunneling is the result of interference between the two components of the coherent superposition defined by Eq. (494) 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. (493) we obtain,

$$|\Phi_t\rangle = \frac{1}{\sqrt{2}}\left(\chi_0(x)e^{-\frac{i}{\hbar}E_0 t}e^{i\theta} + \chi_1(x)e^{-\frac{i}{\hbar}E_1 t}\right), \tag{495}$$

and

$$|\xi(t)| = |\langle\Phi_0|\Phi_t\rangle|^2 = \frac{1}{2} + \frac{1}{2}cos(\Omega t + \theta). \tag{496}$$

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.

29.1 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:

$$\hat{U}^{2\theta} = cos\left(\frac{\Gamma\tau}{2}\right)\left(|\Phi_0\rangle\langle\Phi_0| + |\Phi_a\rangle\langle\Phi_a|\right) - isin\left(\frac{\Gamma\tau}{2}\right)\left(|\Phi_0\rangle\langle\Phi_a| + |\Phi_a\rangle\langle\Phi_0|\right), \tag{497}$$

with $\tau = 2\theta/\Gamma$. In particular, when $\theta = \pi$,

$$\hat{U}^{2\pi} = -\left(|\Phi_0\rangle\langle\Phi_0| + |\Phi_a\rangle\langle\Phi_a|\right), \tag{498}$$

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. (498), 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

$$\hat{U}^{2\pi}|\Psi_t\rangle = -c_0(t_0)|\Phi_0\rangle + c_1(t_0)|\Phi_1\rangle + ... \tag{499}$$

The pulse can also be represented as

$$\hat{U}^{2\pi} = 1 - 2|\Phi_0\rangle\langle\Phi_0|, \tag{500}$$

since

$$\hat{U}^{2\pi}|\Psi_{t_0}\rangle = (1 - 2|\Phi_0\rangle\langle\Phi_0|)|\Psi_t\rangle = -c_0(t_0)|\Phi_0\rangle + c_1(t_0)|\Phi_1\rangle + ... \tag{501}$$

The propagation of the system under the influence of N instantaneous 2-π pulses, applied at 2τ intervals, generates the time-evolved state,

$$\begin{aligned} |\Psi_{t+2N\tau}\rangle &= c_0 \left(e^{-\frac{i}{\hbar}\hat{H}\tau}\hat{U}^{2\pi}e^{-\frac{i}{\hbar}\hat{H}\tau}\right)^N |\Phi_0\rangle + e^{-\frac{i}{\hbar}\hat{H}2N\tau}\left(c_1|\Phi_1\rangle + ...\right), \\ &= c_0(-1)^N e^{-\frac{i}{\hbar}(E_0+E_1)2N\tau} + |\Phi_0\rangle + e^{-\frac{i}{\hbar}\hat{H}2N\tau}\left(c_1|\Phi_1\rangle + ...\right). \end{aligned} \tag{502}$$

The second equality in Eq. (502) is obtained by substituting $\hat{U}^{2\pi}$ as defined by Eq. (500) and Φ_0 according to Eq. (492).

Equation (502) 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.

29.2 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

$$V(x) = -0.5x^2 + 1.0/(16.0 * 1.3544)x^4, \tag{503}$$

using the initial state

$$\Phi_0(x) = \pi^{-1/4}e^{-0.5(x-x0)^2}, \tag{504}$$

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. (500), applied in the time-window $t = 305$–500 a.u.

30 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:

$$I_0(\omega) = \frac{3}{2\pi\hbar}\int_{-\infty}^{\infty} dt e^{\frac{i}{\hbar}(\hbar\omega+E_0)t}\xi(t), \tag{505}$$

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. (505), can be performed by propagation of $|\Psi_0\rangle$ (*e.g.*, according to the SOFT method introduced in Sec. 9); 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. 28, 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:

$$I_\beta(\omega) = \sum_j \rho_j(\beta) I_j(\omega), \tag{506}$$

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:

$$P_0^{(k)}(\omega) = \lim_{t\to\infty} |c_k^{(1)}(t)|^2, \tag{507}$$

where $c_k^{(1)}(t)$ is defined by the Golden Rule expression of first order time-dependent perturbation theory,

$$c_k^{(1)}(t) = -\frac{i}{\hbar}\int_{-\infty}^{t} dt' \langle\Phi_k|e^{\frac{i}{\hbar}\hat{H}t'}\hat{H}_1(t')e^{-\frac{i}{\hbar}\hat{H}t'}|\Phi_0\rangle. \tag{508}$$

The derivation of Eq. (508), presented in the following section, assumes that the photoabsorption results from the interaction of the system with the monochromatic radiation field,

$$\overrightarrow{\epsilon(t)} = \lambda\overrightarrow{\varepsilon_0}(e^{i\omega t} + e^{-i\omega t}), \tag{509}$$

where $\lambda << 1$ is a small dimensionless parameter that defines the dipolar interaction,

$$\hat{H}_1(t) = -\lambda\overrightarrow{\varepsilon_0} \cdot \hat{\mu}(e^{i\omega t} + e^{-i\omega t}), \tag{510}$$

in the weak field limit.

Substituting the expression of the dipolar interaction, introduced by Eq. (510), into Eq. (508), we obtain:

$$c_k^{(1)}(t) = -\frac{i}{\hbar}\int_{-\infty}^{t} dt' \langle\Phi_k|\lambda\overrightarrow{\varepsilon_0} \cdot \hat{\mu}\left[e^{-\frac{i}{\hbar}(\hat{H}-E_k-\hbar\omega)t'} + e^{-\frac{i}{\hbar}(\hat{H}-E_k+\hbar\omega)t'}\right]|\Phi_0\rangle, \tag{511}$$

and substituting Eq. (511) into Eq. (507) we obtain:

$$P_0^{(k)}(\omega) = |\langle\Phi_k|\overrightarrow{\lambda\varepsilon_0} \cdot \hat{\mu}|\Phi_0\rangle|^2 \left[\delta(E_0 + \hbar\omega - E_k) + \delta(E_0 - \hbar\omega - E_k)\right]. \tag{512}$$

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:

$$\alpha_0(\omega) = \sum_k (E_k - E_0)|\langle\Phi_k|\overrightarrow{\lambda\varepsilon_0}\cdot\hat{\mu}|\Phi_0\rangle|^2\left[\delta(E_0+\hbar\omega-E_k)+\delta(E_0-\hbar\omega-E_k)\right]. \tag{513}$$

The absorption spectrum $\alpha_\beta(\omega)$, at finite temperature $T = 1/(\beta k_B)$, can be obtained from Eq. (513) as follows:

$$\begin{aligned}\alpha(\omega) = &\sum_j \rho_j \sum_k (E_k - E_j)\delta(E_j+\hbar\omega-E_k)|\langle\Phi_k|\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu}|\Phi_j\rangle|^2 \\ &+ \sum_j \rho_j \sum_k \sum_k (E_k - E_j)\delta(E_j-\hbar\omega-E_k)|\langle\Phi_k|\overrightarrow{\varepsilon_0}\cdot\hat{\mu}|\Phi_j\rangle|^2,\end{aligned} \tag{514}$$

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. (514) and noting that $\rho_k = \rho_j e^{-\beta(E_k-E_j)}$ we obtain:

$$\begin{aligned}\alpha(\omega) = &\sum_j \sum_k \rho_j (E_k - E_j)\delta(E_j+\hbar\omega-E_k)|\langle\Phi_k|\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu}|\Phi_j\rangle|^2 \\ &- \rho_j e^{-\beta(E_k-E_j)}(E_k - E_j)\delta(E_j+\hbar\omega-E_k)|\langle\Phi_j|\overrightarrow{\varepsilon_0}\cdot\hat{\mu}|\Phi_k\rangle|^2,\end{aligned} \tag{515}$$

which gives the absorption lineshape

$$I(\omega) = \frac{3\alpha(\omega)}{\hbar\omega(1-e^{-\beta\omega})} = 3\sum_j \sum_k \rho_j \delta(E_j+\hbar\omega-E_k)|\langle\Phi_k|\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu}|\Phi_j\rangle|^2. \tag{516}$$

At 0 K, the absorption lineshape is obtained from Eq. (516) as follows:

$$I_0(\omega) = 3\sum_k \delta(E_0+\hbar\omega-E_k)|\langle\Phi_k|\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu}|\Phi_0\rangle|^2, \tag{517}$$

that is equivalent to Eq. (505), since according to Eq. (505),

$$I_0(\omega) = \frac{3}{2\pi\hbar}\int_{-\infty}^{\infty} dt\langle\Phi_0|(\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu})e^{\frac{i}{\hbar}(\hbar\omega+E_0-\hat{H})t}(\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu})|\Phi_0\rangle. \tag{518}$$

Note that introducing the closure relation, $\hat{1} = \sum|\Phi_j\rangle\langle\Phi_j|$ into Eq. (518), we obtain:

$$\begin{aligned}I_0(\omega) &= 3\sum_k \langle\Phi_0|\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu}|\Phi_k\rangle\langle\Phi_k|\delta(\hbar\omega+E_0-\hat{H})\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu}|\Phi_0\rangle, \\ &= 3\sum_k \delta(\hbar\omega+E_0-E_k)\langle\Phi_0|\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu}|\Phi_k\rangle\langle\Phi_k|\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu}|\Phi_0\rangle,\end{aligned} \tag{519}$$

that is identical to Eq. (517).

Finally, we note that Eq. (519) gives the linear photoabsorption lineshape in terms of the dipole-dipole correlation function as follows:

$$\begin{aligned} I_0(\omega) &= \frac{3}{2\pi\hbar}\sum_k \int_{-\infty}^{\infty} dt e^{i\omega t}\langle\Phi_0|e^{\frac{i}{\hbar}\hat{H}t}(\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu})e^{-\frac{i}{\hbar}\hat{H}t}|\Phi_k\rangle\langle\Phi_k|(\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu})|\Phi_0\rangle, \\ &= \frac{3}{2\pi\hbar}\int_{-\infty}^{\infty} dt e^{i\omega t}\langle\Phi_0|(\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu}(t))(\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu}(0))|\Phi_0\rangle, \end{aligned} \tag{520}$$

as well as the finite temperature photoabsorption lineshape,

$$I_\beta(\omega) = \frac{3}{2\pi\hbar}\int_{-\infty}^{\infty} dt e^{i\omega t} Tr[\hat{\rho}\hat{\mu}(t)\hat{\mu}(0)], \tag{521}$$

where $\hat{\rho} = Z^{-1}exp(-\beta\hat{H})$ and $Z = Tr[\hat{\rho}]$.

31 Time Dependent Perturbation Theory

The goal of this section is to derive the Golden rule of time-dependent perturbation theory, introduced by Eq. (508) 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:

$$i\hbar\frac{\partial\psi}{\partial t} = [\hat{H} + \lambda\hat{\omega}(t)]\psi, \tag{522}$$

assuming that such solution can be written as a rapidly convergent expansion in powers of λ,

$$\psi_\lambda(x,t) = \sum_j\sum_{l=0}^{\infty} C_{jl}(t)\lambda^l\Phi_j(x)e^{-\frac{i}{\hbar}E_j t}. \tag{523}$$

Substituting Eq. (523) into Eq. (522) we obtain,

$$i\hbar\sum_{l=0}^{\infty}\left(\dot{C}_{kl}(t)\lambda^l + C_{kl}(t)\lambda^l(-\frac{i}{\hbar}E_k)\right)e^{-\frac{i}{\hbar}E_k t} = \sum_j\sum_{l=0}^{\infty} C_{jl}(t)\lambda^l\left(\langle\Phi_k|\Phi_j\rangle E_j + \lambda\langle\Phi_k|\hat{\omega}|\Phi_j\rangle\right)e^{-\frac{i}{\hbar}E_j t}. \tag{524}$$

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,

$$\dot{C}_{k_1}(t) = -\frac{i}{\hbar}\sum_j C_{j_0}(0) < \Phi_k|e^{\frac{i}{\hbar}E_k t}\hat{\omega}e^{-\frac{i}{\hbar}E_j t}|\Phi_j >= -\frac{i}{\hbar}\sum_j C_{j_0}(0) < \Phi_k|e^{\frac{i}{\hbar}\hat{H}t}\hat{\omega}e^{-\frac{i}{\hbar}\hat{H}t}|\Phi_j >, \tag{525}$$

Equation (525) 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. (525) we obtain,

$$C_{k_1}(t) = -\frac{i}{\hbar}\int_{-\infty}^{t} dt' \sum_j C_{j_0}(0) < \Phi_k|e^{\frac{i}{\hbar}\hat{H}t'}\hat{\omega}e^{-\frac{i}{\hbar}\hat{H}t'}|\Phi_j > .$$

which can also be written as follows:

$$C_{k_1}(t) = -\frac{i}{\hbar}\int_{-\infty}^{t} dt' < \Phi_k|e^{\frac{i}{\hbar}\hat{H}t'}\hat{\omega}e^{-\frac{i}{\hbar}\hat{H}t'}|\tilde{\psi}_0 > .$$

This expression gives the correction of the expansion coefficients to first order in λ.

Note: The substitution made in Eq. (525) can be justified as follows. The exponential function is defined in powers series as follows,

$$e^A = \sum_{n=0}^{\infty} \frac{A^n}{n!} = 1 + A + \frac{1}{2!}AA + ..., \qquad \mathbf{R4(169)}$$

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. (525).

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.

32 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,

$$\overrightarrow{\varepsilon(t)} = \lambda F_1(t - t_0)\overrightarrow{\varepsilon_{01}}e^{-i\omega_1 t} + \lambda F_2(t - t_0 - \Delta t)\overrightarrow{\varepsilon_{02}}e^{-i\omega_2 t} + c.c. \tag{526}$$

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:

$$P_0 = \sum_k P_0^{(k)} = \lim_{t_f \to \infty} \sum_k |c_k^{(2)}(t_f)|^2, \tag{527}$$

where $c_k^{(2)}(t_f)$ is defined by second order time-dependent perturbation theory,

$$c_k^{(2)}(t_f) = \hbar^{-2} \int_{-\infty}^{t_f} dt' \int_{-\infty}^{t'} dt'' \langle \Phi_k | e^{\frac{i}{\hbar}\hat{H}t''} \hat{H}_1(t'') e^{-\frac{i}{\hbar}\hat{H}(t''-t')} \hat{H}_1(t') e^{-\frac{i}{\hbar}\hat{H}t'} | \Phi_0 \rangle, \tag{528}$$

with

$$\hat{H}_1(t) = -(\lambda\overrightarrow{\varepsilon_{01}} \cdot \hat{\mu})F_1(t - t_0)e^{-i\omega_1 t} - (\lambda\overrightarrow{\varepsilon_{02}} \cdot \hat{\mu})F_2(t - t_0 - \Delta t)e^{-i\omega_2 t} + c.c. \tag{529}$$

Substituting Eq. (529) into Eq. (528) and then substituting Eq. (527) into Eq. (527) 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

$$\begin{aligned} P_0(+\omega_2, +\omega_1) = \hbar^{-4} \int_{-\infty}^{\infty} dt \int_{-\infty}^{t} dt' \int_{-\infty}^{\infty} dt'' \int_{-\infty}^{t''} dt''' e^{i\omega_1(t'''-t')} e^{i\omega_2(t''-t)} F_1(t' - t_0) \\ \times F_2(t - t_0 - \Delta t) F_1(t''' - t_0) F_2(t'' - t_0 - \Delta t) \langle \Phi_0 | e^{\frac{i}{\hbar}\hat{H}t'''} (\lambda\overrightarrow{\varepsilon_{01}} \cdot \hat{\mu}) e^{-\frac{i}{\hbar}\hat{H}t'''} \\ \times e^{\frac{i}{\hbar}\hat{H}t} (\lambda\overrightarrow{\varepsilon_{02}} \cdot \hat{\mu}) e^{-\frac{i}{\hbar}\hat{H}t} e^{\frac{i}{\hbar}\hat{H}t''} (\lambda\overrightarrow{\varepsilon_{02}} \cdot \hat{\mu}) e^{-\frac{i}{\hbar}\hat{H}t''} e^{\frac{i}{\hbar}\hat{H}t'} (\lambda\overrightarrow{\varepsilon_{01}} \cdot \hat{\mu}) e^{-\frac{i}{\hbar}\hat{H}t'} | \Phi_0 \rangle, \end{aligned} \tag{530}$$

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 osciallatory. Equation (530) 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:

$$\begin{aligned} I_2(X) + \hbar\omega_1 &\rightarrow I_2(B), \\ I_2(B) + \hbar\omega_2 &\rightarrow I_2(f). \end{aligned} \tag{531}$$

Considering that the electronic transition dipole moments are independent of nuclear coordinates, we obtain:

$$\begin{aligned} I(\omega_2, \Delta t) = \hbar^{-4}(\lambda\overrightarrow{\varepsilon_{01}} \cdot \overrightarrow{\mu_{BX}})^2(\lambda\overrightarrow{\varepsilon_{02}} \cdot \overrightarrow{\mu_{fB}})^2 \sum_j \rho_j \int_{-\infty}^{\infty} dt \int_{-\infty}^{\infty} dt'' e^{-i\omega_2(t''-t)} \\ \times \int_{-\infty}^{t} dt' \int_{-\infty}^{t''} dt''' e^{\frac{i}{\hbar}(E_j+\hbar\omega_1)(t'''-t')} F_1(t'-t_0) F_2(t-t_0-\Delta t) \\ \times F_1(t'''-t_0) F_2(t''-t_0-\Delta t) \xi_j(t-t''', t''-t, t'-t''), \end{aligned} \tag{532}$$

where

$$\xi_j(t-t''', t''-t, t'-t'') = \langle \Phi_j | e^{-\frac{i}{\hbar}\hat{H}_B(t-t''')} e^{\frac{i}{\hbar}\hat{H}_f(t''-t)} e^{\frac{i}{\hbar}\hat{H}_B(t'-t'')} | \Phi_j \rangle. \tag{533}$$

If desired, Eq. (532) could also be written as

$$I(\omega_2, \Delta t) = \int_{-\infty}^{\infty} dt e^{i\omega_2 t} C(t, \Delta t), \tag{534}$$

where $C(t, \Delta t)$ is readily identifiable from Eq. (532):

$$\begin{aligned} C(t, \Delta t) = \hbar^{-4}(\lambda\overrightarrow{\varepsilon_{01}} \cdot \overrightarrow{\mu_{BX}})^2(\lambda\overrightarrow{\varepsilon_{02}} \cdot \overrightarrow{\mu_{fB}})^2 \sum_j \rho_j \int_{-\infty}^{\infty} dt'' \int_{-\infty}^{t''} dt' \int_{-\infty}^{t+t''} dt''' e^{\frac{i}{\hbar}(E_j+\hbar\omega_1)(t'''-t')} \\ \times F_1(t'-t_0) F_2(t''+t-t_0-\Delta t) F_1(t'''-t_0) F_2(t+t''-t_0-\Delta t) \\ \times \xi_j(t''-t''', t, t'-t''-t). \end{aligned} \tag{535}$$

33 Pump-Probe Photoelectron Spectroscopy

Pump-probe photoelectron spectroscopy is essentially the same technique discussed Sec. 32 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. 32, 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

$$\begin{aligned} &I_2^-(X) + \hbar\omega_1 \to I_2^-(A'), \\ &I_2^-(A') + \hbar\omega_2 \to I_2(K,\nu) + e^-(\epsilon), \end{aligned} \tag{536}$$

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

$$|\Phi_0\rangle = |\psi_g\rangle|\chi_g\rangle, \tag{537}$$

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

$$|\Phi_f\rangle = |\psi_K\rangle|\chi_{E_K}\rangle, \tag{538}$$

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,

$$e^{-\frac{i}{\hbar}\hat{H}(t''-t')} = e^{-\frac{i}{\hbar}\hat{H}_{A'}(t''-t')}|\psi_{A'}\rangle\langle\psi_{A'}|, \tag{539}$$

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. (528), 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,

$$P_K = \left| \hbar^{-2} \int_{-\infty}^{\infty} dt' \int_{-\infty}^{t'} dt'' e^{\frac{i}{\hbar}(E_K+\epsilon)t''} \langle\Phi_f|\hat{H}_1(t'')e^{-\frac{i}{\hbar}\hat{H}_{A'}(t''-t')}\hat{H}_1(t')e^{-\frac{i}{\hbar}\hat{H}t'}|\Phi_0\rangle \right|^2 . \tag{540}$$

and the probability of a 2-photon transition to the electronic state K, leaving the photodetached electron with kinetic energy ϵ is

$$P_K(\epsilon) = \hbar^{-4} \int dE_K \left| \int_{-\infty}^{\infty} dt' \int_{-\infty}^{t'} dt'' e^{\frac{i}{\hbar}(E_K+\epsilon)t''} \langle\Phi_f|\hat{H}_1(t'')e^{-\frac{i}{\hbar}\hat{H}(t''-t')}\hat{H}_1(t')e^{-\frac{i}{\hbar}\hat{H}t'}|\Phi_0\rangle \right|^2 . \tag{541}$$

Therefore, the total probability of a 2-photon transition, leaving the photodetached electron with kinetic energy ϵ is $P(\epsilon) = \sum_K P_K(\epsilon)$:

$$P(\epsilon) = \hbar^{-4} \sum_K \int dE_K \left| \int_{-\infty}^{\infty} dt' \int_{-\infty}^{t'} dt'' e^{\frac{i}{\hbar}(E_K+\epsilon)t''} \langle\Phi_f|\hat{H}_1(t'')e^{-\frac{i}{\hbar}\hat{H}_{A'}(t''-t')}\hat{H}_1(t')e^{-\frac{i}{\hbar}\hat{H}t'}|\Phi_0\rangle \right|^2 . \tag{542}$$

Explicitly squaring the r.h.s. of Eq. (542) and using the relation,

$$\int dE_K |\chi_{E_K}\rangle\langle\chi_{E_K}| e^{-\frac{i}{\hbar}E_K(t''-t)} = e^{-\frac{i}{\hbar}H_K(t''-t)}, \tag{543}$$

we obtain the $(+\omega_1, +\omega_2)$ term, analogous to Eq. (530):

$$\begin{aligned} P(\epsilon) = \hbar^{-4} \sum_K \int_{-\infty}^{\infty} dt \int_{-\infty}^{t} dt' \int_{-\infty}^{\infty} dt'' \int_{-\infty}^{t''} dt''' e^{i\omega_1(t'''-t')} e^{i\omega_2(t''-t)} F_1(t'-t_0) F_2(t-t_0-\Delta t) \\ \times F_1(t'''-t_0) F_2(t''-t_0-\Delta t) e^{\frac{i}{\hbar}\epsilon(t''-t)} \langle\Phi_0| e^{\frac{i}{\hbar}\hat{H}_X t'''} (\lambda\overrightarrow{\varepsilon_{01}} \cdot \overrightarrow{\mu}_{XA'}) e^{-\frac{i}{\hbar}\hat{H}_{A'}t'''} \\ \times e^{\frac{i}{\hbar}\hat{H}_{A'}t} (\lambda\overrightarrow{\varepsilon_{02}} \cdot \overrightarrow{\mu}_{A'K}) e^{-\frac{i}{\hbar}\hat{H}_K t} e^{\frac{i}{\hbar}\hat{H}_K t''} (\lambda\overrightarrow{\varepsilon_{02}} \cdot \overrightarrow{\mu}_{KA'}) e^{-\frac{i}{\hbar}\hat{H}_{A'}t''} \\ \times e^{\frac{i}{\hbar}\hat{H}_{A'}t'} (\lambda\overrightarrow{\varepsilon_{01}} \cdot \overrightarrow{\mu}_{A'X}) e^{-\frac{i}{\hbar}\hat{H}_X t'} |\Phi_0\rangle, \end{aligned} \tag{544}$$

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. (532):

$$\begin{aligned} P_\beta(\epsilon, \Delta t) = \hbar^{-4} (\lambda\overrightarrow{\varepsilon_{01}} \cdot \overrightarrow{\mu_{A'X}})^2 (\lambda\overrightarrow{\varepsilon_{02}} \cdot \overrightarrow{\mu_{KA'}})^2 \sum_j \rho_j \int_{-\infty}^{\infty} dt \int_{-\infty}^{\infty} dt'' e^{i(\omega_2+\epsilon)(t''-t)} \\ \times \int_{-\infty}^{t} dt' \int_{-\infty}^{t''} dt''' e^{\frac{i}{\hbar}(E_j+\hbar\omega_1)(t'''-t')} F_1(t'-t_0) F_2(t-t_0-\Delta t) \\ \times F_1(t'''-t_0) F_2(t''-t_0-\Delta t) \xi_j(t-t''', t''-t, t'-t''), \end{aligned} \tag{545}$$

where

$$\xi_j(t-t''', t''-t, t'-t'') = \langle\Phi_j| e^{-\frac{i}{\hbar}\hat{H}_{A'}(t-t''')} e^{\frac{i}{\hbar}\hat{H}_K(t''-t)} e^{\frac{i}{\hbar}\hat{H}_{A'}(t'-t'')} |\Phi_j\rangle, \tag{546}$$

and $\rho_j = Z^{-1} exp(-\beta E_j)$, with $Z = \sum_j exp(-\beta E_j)$. If desired, Eq. (545) could also be written as

$$P_\beta(\epsilon, \Delta t) = \int_{-\infty}^{\infty} dt e^{i\epsilon t} C(t, \Delta t), \tag{547}$$

where $C(t, \Delta t)$ is readily identifiable from Eq. (545):

$$\begin{aligned} C(t, \Delta t) = \hbar^{-4} (\lambda\overrightarrow{\varepsilon_{01}} \cdot \overrightarrow{\mu_{XB}})^2 (\lambda\overrightarrow{\varepsilon_{02}} \cdot \overrightarrow{\mu_{BG}})^2 \sum_j \rho_j \int_{-\infty}^{\infty} dt'' \int_{-\infty}^{t''} dt' \int_{-\infty}^{t+t''} dt''' e^{\frac{i}{\hbar}(E_j+\hbar\omega_1)(t'''-t')} \\ \times F_1(t'-t_0) F_2(t''+t-t_0-\Delta t) F_1(t'''-t_0) F_2(t+t''-t_0-\Delta t) e^{\frac{i}{\hbar}\omega_2 t} \\ \times \xi_j(t''-t''', t, t'-t''-t). \end{aligned} \tag{548}$$

34 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 30. As an example, we consider the direct photoelectron detachment spectroscopy of I_2^-, studied among others by Neumark and co-workers [*J. Chem. Phys.* (1999) **110**:3736],

$$I_2^-(X) + \hbar\omega_1 \rightarrow I_2(K,\nu) + e^-(\epsilon). \tag{549}$$

As in Sec. 33, the initial state of I_2^- is

$$|\Phi_0\rangle = |\psi_g\rangle|\chi_g\rangle, \tag{550}$$

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

$$|\Phi_f\rangle = |\psi_K\rangle|\chi_{E_K}\rangle, \tag{551}$$

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. (520), the photoabsorption lineshape (at 0 K) for the one-photon photodetachment process is,

$$P_0(\epsilon) = \frac{3}{2\pi\hbar}\sum_K \int dE_K \int_{-\infty}^{\infty} dt e^{i\omega t}\langle\Phi_0|e^{\frac{i}{\hbar}\hat{H}t}(\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu})e^{-\frac{i}{\hbar}(E_K+\epsilon)t}|\Phi_k\rangle\langle\Phi_k|(\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu})|\Phi_0\rangle, \tag{552}$$

and using the relation

$$\int dE_K|\chi_{E_K}\rangle\langle\chi_{E_K}|e^{-\frac{i}{\hbar}E_K t} = e^{-\frac{i}{\hbar}H_K t}, \tag{553}$$

the finite temperature distribution is

$$P_\beta(\epsilon) = \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} dt e^{-\frac{i}{\hbar}\epsilon t}C(t), \tag{554}$$

with

$$C(t) = 3\sum_j \rho_j e^{\frac{i}{\hbar}(E_j+\hbar\omega t)}\sum_K \langle\Phi_j|(\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu})e^{-\frac{i}{\hbar}\hat{H}_K t}(\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu})|\Phi_j\rangle. \tag{555}$$

34.1 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 $V_0(g) = 0.00\ eV$; $V_0(e) = 0.94\ eV$; $D_e(g) = 18941\ cm^{-1}$; $D_e(e) = 4911\ cm^{-1}$; $\beta(g) = 1.517\ \mathring{A}^{-1}$; $\beta(e) = 1.535\ \mathring{A}^{-1}$; $R_{eq}(g) = 2.66\ \mathring{A}$ and $R_{eq}(e) = 3.105\ \mathring{A}$.

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].

Solution in Sec. 40.19 .

35 Exam 2

Exam 2 CHEM 572a
Advanced Quantum Mechanics

Exercise 1: DVR Method
(**10 points**) **1.1**: Explain the discrete variable representation method discussed in class and how to implement it numerically.

(**20 points**) **1.2**: Prove that the elements of the kinetic energy matrix can be expressed in the representation of equally spaced delta functions as follows:

$$T^{(\delta)}(j,j') = \frac{\hbar^2\pi^2}{2m(x_{max}-x_{min})^2}\frac{2}{N}\sum_{k=1}^{N-1} k^2 \text{Sin}\left(\frac{k\pi j}{N}\right)\text{Sin}\left(\frac{k\pi j'}{N}\right), \tag{556}$$

where the delta functions $\delta(x-x_j)$ are equally spaced as follows:

$$x_j = x_{min} + j\Delta, \text{ with } \Delta = (x_{max}-x_{min})/N, \tag{557}$$

with $j = 1$–(N-1).

Exercise 2: Tunneling (**10 points**) **2.1**: Explain how to compute the tunneling splitting of a proton in a symmetric double-well potential described by the following unperturbed Hamiltonian,

$$H_0(x,p) = \frac{p^2}{2} - \alpha(x^2 - \beta x^4), \tag{558}$$

with $\alpha = 1/2^2$ and $\beta = 1/2^5$.

(**20 points**) **2.2**: Prove that the underlying tunneling dynamics of a proton in a the double well potential described in **2.1** can be coherently controlled by a sequence of sufficiently frequent 2-π pulses when each pulse is described by the following operator:

$$\hat{U}^{2\pi} = 1 - 2|\Phi_0\rangle\langle\Phi_0|, \tag{559}$$

where $\Phi_0(x)$ is the initial state defined as follows:

$$\Phi_0(x) = \frac{1}{\sqrt{2}}\left(\chi_0(x) + \chi_1(x)\right), \tag{560}$$

with $\chi_0(x)$ and $\chi_1(x)$ the ground and first excited states of the double-well potential, respectively.

Exercise 3: Spectroscopy

(**20 points**) **3.1**: Prove that the linear photoabsorption lineshape $I_0(\omega)$ of a system at 0 K can be obtained as the Fourier transform of the survival amplitude $\xi(t) \equiv \langle \Psi_0 | \Psi_t \rangle$ as follows:

$$I_0(\omega) = \frac{3}{2\pi\hbar} \int_{-\infty}^{\infty} dt e^{\frac{i}{\hbar}(\hbar\omega + E_0)t} \xi(t), \tag{561}$$

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. Assume that the photoabsorption results from the interaction of the system with the monochromatic radiation field,

$$\overrightarrow{\epsilon(t)} = \lambda \overrightarrow{\varepsilon_0} (e^{i\omega t} + e^{-i\omega t}), \tag{562}$$

where $\lambda << 1$ is a small dimensionless parameter that defines the dipolar interaction,

$$\hat{H}_1(t) = -\lambda \overrightarrow{\varepsilon_0} \cdot \hat{\mu} (e^{i\omega t} + e^{-i\omega t}), \tag{563}$$

in the weak field limit.

(**20 points**) **3.2**: The total transition probability P_0 (at 0 K) due to a two-photon interaction of a system with an external radiation field can be obtained by first computing the transition probablity to a generic state $|\Phi_k\rangle$ and then summing the contributions from all possible final states $|\Phi_k\rangle$ as follows:

$$P_0 = \sum_k P_0^{(k)} = \lim_{t_f \to \infty} \sum_k |c_k^{(2)}(t_f)|^2. \tag{564}$$

Prove that, according to second order time-dependent perturbation theory, $c_k^{(2)}(t_f)$ is defined as follows:

$$c_k^{(2)}(t_f) = \hbar^{-2} \int_{-\infty}^{t_f} dt' \int_{-\infty}^{t'} dt'' \langle \Phi_k | e^{\frac{i}{\hbar}\hat{H}t''} \hat{H}_1(t'') e^{-\frac{i}{\hbar}\hat{H}(t''-t')} \hat{H}_1(t') e^{-\frac{i}{\hbar}\hat{H}t'} | \Phi_0 \rangle, \tag{565}$$

where $\hat{H}_1(t)$ is the dipolar radiation-matter interaction.

35.1 Answer Key

Exercise 1: DVR Method
(**10 points**) **1.1**: Explain the discrete variable representation method discussed in class and how to implement it numerically.

The DVR method solves the time-independent Schrödinger equation,

$$HC_j - C_j E_j = 0, \tag{566}$$

and obtains the eigenstates $\chi_j(x)$ in a grid-based representation: $\chi_j(x) = \sum_k C(k,j)\delta(x - x_k)$, as well as the corresponding eigenvalues E_j, by simple diagonalization of the Hamiltonian matrix H. This is accomplished 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 grid based representation is composed of delta functions $\delta(x - x_j)$, equally spaced at coordinates x_j as follows:

$$x_j = x_{min} + j\Delta, \text{ with } \Delta = (x_{max} - x_{min})/N, \tag{567}$$

with $j = 1$–(N-1). In such a representation, the Hamiltonian matrix elements are:

$$H(j,j') = V(x_j)\delta_{jj'} + T(j,j'), \tag{568}$$

where $T(j,j')$ is defined in item 1.2.
(**20 points**) **1.2**: Prove that the elements of the kinetic energy matrix can be expressed in the representation of equally spaced delta functions as follows:

$$T^{(\delta)}(j,j') = \frac{\hbar^2\pi^2}{2m(x_{max} - x_{min})^2}\frac{2}{N}\sum_{k=1}^{N-1} k^2 \text{Sin}\left(\frac{k\pi j}{N}\right)\text{Sin}\left(\frac{k\pi j'}{N}\right), \tag{569}$$

where the delta functions $\delta(x - x_j)$ are equally spaced as follows:

$$x_j = x_{min} + j\Delta, \text{ with } \Delta = (x_{max} - x_{min})/N, \tag{570}$$

with $j = 1$–(N-1).

We consider the Hamiltonian,

$$\hat{H} = \hat{T} + V(\hat{x}), \tag{571}$$

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:

$$\begin{aligned} V^{(\delta)}(j,k) = \langle j|V(\hat{x})|k\rangle &= \int dx \delta^*(x - x_j)V(\hat{x})\delta(x - x_k), \\ &= V(x_k)\delta_{j,k}. \end{aligned} \tag{572}$$

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:

$$T^{(\delta)} = \Gamma^{-1} T^{(\phi)} \Gamma, \tag{573}$$

where Γ is the transformation matrix defined by the linear combinations,

$$\phi_k(x) = \sum_j \Gamma(j,k)\delta(x - x_j)\Delta', \tag{574}$$

where

$$\Gamma(j,k) = \phi_k(x_j). \tag{575}$$

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:

$$\phi_k(x) = \sqrt{\frac{2}{x_{max} - x_{min}}} \text{Sin}\left(k \frac{\pi(x - x_{min})}{(x_{max} - x_{min})}\right), \tag{576}$$

with $\phi_k(x_{min}) = 0$ and $\phi_k(x_{max}) = 0$. Therefore,

$$\hat{T}\phi_k(x) = \frac{(\hbar\pi k)^2}{2m}\phi_k(x), \tag{577}$$

and $T^{(\phi)}$ is diagonal with matrix elements,

$$\hat{T}^{(\phi)}(j,k) = \langle \phi_j | \hat{T} | \phi_k \rangle = \frac{(\hbar k)^2}{2m} \frac{\pi^2}{(x_{max} - x_{min})^2} \delta_{jk}. \tag{578}$$

Therefore, substituting Eq. (578) and Eq. (575) into Eq. (573) we obtain,

$$\begin{aligned} T^{(\delta)}(i,i') &= \sum_{j,k=1}^{N-1} \Gamma^{-1}(i,j) T^{(\phi)}(j,k) \Gamma(k,i') = \sum_{j,k=1}^{N-1} \Gamma(j,i) T^{(\phi)}(j,k) \Gamma(k,i'), \\ &= \frac{\Delta\pi^2}{(x_{max} - x_{min})^2} \sum_{j,k=1}^{N-1} \phi_j(x_i) \frac{(\hbar k)^2}{2m} \delta_{jk} \phi_k(x_i') = \frac{\Delta\pi^2}{(x_{max} - x_{min})^2} \sum_{k=1}^{N-1} \phi_k(x_i) \frac{(\hbar k)^2}{2m} \phi_k(x_i'), \\ &= \frac{\Delta\hbar^2\pi^2}{2m(x_{max} - x_{min})^2} \frac{2}{(x_{max} - x_{min})} \sum_{k=1}^{N-1} k^2 \text{Sin}\left(k\pi \frac{(x_i - x_{min})}{(x_{max} - x_{min})}\right) \text{Sin}\left(k\pi \frac{(x_{i'} - x_{min})}{(x_{max} - x_{min})}\right). \end{aligned} \tag{579}$$

Finally, substituting Eq. (570) into Eq. (579) we obtain:

$$T^{(\delta)}(j,j') = \frac{\hbar^2\pi^2}{2m(x_{max} - x_{min})^2} \frac{2}{N} \sum_{k=1}^{N-1} k^2 \text{Sin}\left(\frac{k\pi j}{N}\right) \text{Sin}\left(\frac{k\pi j'}{N}\right). \tag{580}$$

Exercise 2: Tunneling

(**10 points**) **2.1**: Explain how to compute the tunneling splitting of a proton in a symmetric double-well potential described by the following unperturbed Hamiltonian,

$$H_0(x,p) = \frac{p^2}{2} - \alpha(x^2 - \beta x^4), \tag{581}$$

with $\alpha = 1/2^2$ and $\beta = 1/2^5$.

The tunneling splitting $\Delta = (E_1 - E_0)$ is the energy spacing between the ground and the first excited state. One way of computing this level spacing is by using the DVR method described in the previous problem to obtain E_1 and E_0 and then to compute the difference.

Another way of computing the tunneling splitting is by propagating a non-stationary state. The lowest frequency peak of the Fourier transform of the survival amplitude $\xi(t) \equiv \langle\Psi_0|\Psi_t\rangle$ is located at $\Omega = (E_1 - E_0)/\hbar$ and therefore indicates the value of Ω. To show this, we consider the propagation of the initial state,

$$\Phi_0(x) = c_0\chi_0(x) + c_1\chi_1(x) + ..., \tag{582}$$

where $\chi_0(x)$ and $\chi_1(x)$ are the ground and first excited states of the double-well. Since $\hat{H}|\chi_j\rangle = E_j|\chi_j\rangle$,

$$|\Phi_t\rangle = c_0\chi_0(x)e^{-\frac{i}{\hbar}E_0t} + c_1\chi_1(x)e^{-\frac{i}{\hbar}E_1t} + ..., \tag{583}$$

and

$$|\xi(t)|^2 = |\langle\Phi_0|\Phi_t\rangle|^2 = |c_0|^4 + |c_1|^4 + 2|c_0c_1|^2 cos[\Omega t] + ..., \tag{584}$$

where $\Omega = (E_1 - E_0)/\hbar$ is the tunneling frequency. Therefore, according to Eq. (584), the Fourier transform of $\xi(t)$ should have a prominent peak at Ω.

(**20 points**) **2.2**: Prove that the underlying tunneling dynamics of a proton in a the double well potential described in **2.1** can be coherently controlled by a sequence of sufficiently frequent 2-π pulses when each pulse is described by the following operator:

$$\hat{U}^{2\pi} = 1 - 2|\Phi_0\rangle\langle\Phi_0|, \tag{585}$$

where $\Phi_0(x)$ is the initial state defined as follows:

$$\Phi_0(x) = \frac{1}{\sqrt{2}}\left(\chi_0(x) + \chi_1(x)\right), \tag{586}$$

with $\chi_0(x)$ and $\chi_1(x)$ the ground and first excited states of the double-well potential, respectively.

The propagation of the system under the influence of N instantaneous 2-π pulses, applied at 2τ intervals, generates the time-evolved state,

$$\begin{aligned} |\Psi_{t+2N\tau}\rangle &= c_0\left(e^{-\frac{i}{\hbar}\hat{H}\tau}\hat{U}^{2\pi}e^{-\frac{i}{\hbar}\hat{H}\tau}\right)^N|\Phi_0\rangle + e^{-\frac{i}{\hbar}\hat{H}2N\tau}\left(c_1|\Phi_1\rangle + ...\right), \\ &= c_0(-1)^N e^{-\frac{i}{\hbar}(E_0+E_1)2N\tau} + |\Phi_0\rangle + e^{-\frac{i}{\hbar}\hat{H}2N\tau}\left(c_1|\Phi_1\rangle + ...\right). \end{aligned} \tag{587}$$

The second equality in Eq. (587) is obtained by substituting $\hat{U}^{2\pi}$ as defined by Eq. (585) and Φ_0 according to Eq. (586).

Equation (587) 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.

Exercise 3: Spectroscopy

(**20 points**) **3.1**: Prove that the linear photoabsorption lineshape $I_0(\omega)$ of a system at 0 K can be obtained as the Fourier transform of the survival amplitude $\xi(t) \equiv \langle \Psi_0 | \Psi_t \rangle$ as follows:

$$I_0(\omega) = \frac{3}{2\pi\hbar} \int_{-\infty}^{\infty} dt e^{\frac{i}{\hbar}(\hbar\omega + E_0)t} \xi(t), \tag{588}$$

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. Assume that the photoabsorption results from the interaction of the system with the monochromatic radiation field,

$$\overrightarrow{\epsilon(t)} = \lambda \overrightarrow{\varepsilon_0} (e^{i\omega t} + e^{-i\omega t}), \tag{589}$$

where $\lambda << 1$ is a small dimensionless parameter that defines the dipolar interaction,

$$\hat{H}_1(t) = -\lambda \overrightarrow{\varepsilon_0} \cdot \hat{\mu} (e^{i\omega t} + e^{-i\omega t}), \tag{590}$$

in the weak field limit.

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:

$$P_0^{(k)}(\omega) = \lim_{t \to \infty} |c_k^{(1)}(t)|^2, \tag{591}$$

where $c_k^{(1)}(t)$ is defined by the Golden Rule expression of first order time-dependent perturbation theory,

$$c_k^{(1)}(t) = -\frac{i}{\hbar} \int_{-\infty}^{t} dt' \langle \Phi_k | e^{\frac{i}{\hbar}\hat{H}t'} \hat{H}_1(t') e^{-\frac{i}{\hbar}\hat{H}t'} | \Phi_0 \rangle. \tag{592}$$

Substituting the expression of the dipolar interaction, introduced by Eq. (590), into Eq. (592), we obtain:

$$c_k^{(1)}(t) = -\frac{i}{\hbar} \int_{-\infty}^{t} dt' \langle \Phi_k | \lambda \overrightarrow{\varepsilon_0} \cdot \hat{\mu} \left[e^{-\frac{i}{\hbar}(\hat{H} - E_k - \hbar\omega)t'} + e^{-\frac{i}{\hbar}(\hat{H} - E_k + \hbar\omega)t'} \right] | \Phi_0 \rangle, \tag{593}$$

and substituting Eq. (593) into Eq. (591) we obtain:

$$P_0^{(k)}(\omega) = |\langle \Phi_k | \overrightarrow{\lambda \varepsilon_0} \cdot \hat{\mu} | \Phi_0 \rangle|^2 \left[\delta(E_0 + \hbar\omega - E_k) + \delta(E_0 - \hbar\omega - E_k) \right]. \tag{594}$$

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:

$$\alpha_0(\omega) = \sum_k (E_k - E_0) |\langle \Phi_k | \overrightarrow{\lambda \varepsilon_0} \cdot \hat{\mu} | \Phi_0 \rangle|^2 \left[\delta(E_0 + \hbar\omega - E_k) + \delta(E_0 - \hbar\omega - E_k) \right]. \tag{595}$$

The absoprtion spectrum $\alpha_\beta(\omega)$, at finite temperature $T = 1/(\beta k_B)$, can be obtained from Eq. (595) as follows:

$$\begin{aligned} \alpha(\omega) = & \sum_j \rho_j \sum_k (E_k - E_j) \delta(E_j + \hbar\omega - E_k) |\langle \Phi_k | \lambda \overrightarrow{\varepsilon_0} \cdot \hat{\mu} | \Phi_j \rangle|^2 \\ & + \sum_j \rho_j \sum_k \sum_k (E_k - E_j) \delta(E_j - \hbar\omega - E_k) |\langle \Phi_k | \overrightarrow{\varepsilon_0} \cdot \hat{\mu} | \Phi_j \rangle|^2, \end{aligned} \tag{596}$$

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. (596) and noting that $\rho_k = \rho_j e^{-\beta(E_k - E_j)}$ we obtain:

$$\begin{aligned} \alpha(\omega) = & \sum_j \sum_k \rho_j (E_k - E_j) \delta(E_j + \hbar\omega - E_k) |\langle \Phi_k | \lambda \overrightarrow{\varepsilon_0} \cdot \hat{\mu} | \Phi_j \rangle|^2 \\ & - \rho_j e^{-\beta(E_k - E_j)} (E_k - E_j) \delta(E_j + \hbar\omega - E_k) |\langle \Phi_j | \overrightarrow{\varepsilon_0} \cdot \hat{\mu} | \Phi_k \rangle|^2, \end{aligned} \tag{597}$$

which gives the absorption lineshape

$$I(\omega) = \frac{3\alpha(\omega)}{\hbar\omega(1 - e^{-\beta\omega})} = 3 \sum_j \sum_k \rho_j \delta(E_j + \hbar\omega - E_k) |\langle \Phi_k | \lambda \overrightarrow{\varepsilon_0} \cdot \hat{\mu} | \Phi_j \rangle|^2. \tag{598}$$

At 0 K, the absorption lineshape is obtained from Eq. (598) as follows:

$$I_0(\omega) = 3 \sum_k \delta(E_0 + \hbar\omega - E_k) |\langle \Phi_k | \lambda \overrightarrow{\varepsilon_0} \cdot \hat{\mu} | \Phi_0 \rangle|^2, \tag{599}$$

that is equivalent to Eq. (588), since according to Eq. (588),

$$I_0(\omega) = \frac{3}{2\pi\hbar} \int_{-\infty}^{\infty} dt \langle \Phi_0 | (\lambda \overrightarrow{\varepsilon_0} \cdot \hat{\mu}) e^{\frac{i}{\hbar}(\hbar\omega + E_0 - \hat{H})t} (\lambda \overrightarrow{\varepsilon_0} \cdot \hat{\mu}) | \Phi_0 \rangle. \tag{600}$$

(**20 points**) **3.2**: The total transition probability P_0 (at 0 K) due to a two-photon interaction of a system with an external radiation field can be obtained by first computing the transition probablity to a generic state $|\Phi_k\rangle$ and then summing the contributions from all possible final states $|\Phi_k\rangle$ as follows:

$$P_0 = \sum_k P_0^{(k)} = \lim_{t_f \to \infty} \sum_k |c_k^{(2)}(t_f)|^2. \tag{601}$$

Prove that, according to second order time-dependent perturbation theory, $c_k^{(2)}(t_f)$ is defined as follows:

$$c_k^{(2)}(t_f) = \hbar^{-2} \int_{-\infty}^{t_f} dt' \int_{-\infty}^{t'} dt'' \langle \Phi_k | e^{\frac{i}{\hbar}\hat{H}t''} \hat{H}_1(t'') e^{-\frac{i}{\hbar}\hat{H}(t''-t')} \hat{H}_1(t') e^{-\frac{i}{\hbar}\hat{H}t'} | \Phi_0 \rangle, \tag{602}$$

where $\hat{H}_1(t)$ is the dipolar radiation-matter interaction.

Given an arbitary state,

$$\tilde{\psi}(x,t) = \sum_j C_j \Phi_j(x) e^{-\frac{i}{\hbar}E_j t},$$

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:

$$i\hbar \frac{\partial \psi}{\partial t} = [\hat{H} + \lambda \hat{\omega}(t)]\psi, \tag{603}$$

assuming that such a solution can be written as a rapidly convergent expansion in powers of λ,

$$\psi_\lambda(x,t) = \sum_j \sum_{l=0}^{\infty} C_{jl}(t) \lambda^l \Phi_j(x) e^{-\frac{i}{\hbar}E_j t}. \tag{604}$$

Substituting Eq. (604) into Eq. (603) we obtain,

$$i\hbar \sum_{l=0}^{\infty} \left(\dot{C}_{kl}(t)\lambda^l + C_{kl}(t)\lambda^l (-\frac{i}{\hbar}E_k) \right) e^{-\frac{i}{\hbar}E_k t} = \sum_j \sum_{l=0}^{\infty} C_{jl}(t)\lambda^l \left(\langle \Phi_k | \Phi_j \rangle E_j + \lambda \langle \Phi_k | \hat{\omega} | \Phi_j \rangle \right) e^{-\frac{i}{\hbar}E_j t}. \tag{605}$$

Terms with λ^2: (Second-order time dependent perturbation theory)

$$i\hbar[\dot{C}_{k_2}(t) + C_{k_2}(t)(-\frac{i}{\hbar}E_k)]e^{-\frac{i}{\hbar}E_k t} = \sum_j [C_{j_2}(t)\delta_{kj}E_j + C_{j_1}(t) < \Phi_k | \hat{\omega} | \Phi_j >] e^{-\frac{i}{\hbar}E_j t},$$

$$\dot{C}_{k_2}(t) = -\frac{i}{\hbar} \sum_j < \Phi_k | e^{\frac{i}{\hbar}\hat{H}t} \hat{\omega} e^{-\frac{i}{\hbar}\hat{H}t} | \Phi_j > C_{j_1}(t),$$

$$C_{k_2}(t) = \left(-\frac{i}{\hbar} \right) \int_{-\infty}^{t} dt' \sum_j < \Phi_k | e^{\frac{i}{\hbar}\hat{H}t'} \hat{\omega} e^{-\frac{i}{\hbar}\hat{H}t'} | \Phi_j > C_{j_1}(t'),$$

$$C_{k_2}(t) = \left(-\frac{i}{\hbar} \right)^2 \sum_j \int_{-\infty}^{t} dt' \int_{-\infty}^{t'} dt'' < \Phi_k | e^{\frac{i}{\hbar}\hat{H}t'} \hat{\omega} e^{-\frac{i}{\hbar}\hat{H}t'} | \Phi_j >< \Phi_j | e^{\frac{i}{\hbar}\hat{H}t''} \hat{\omega} e^{-\frac{i}{\hbar}\hat{H}t''} | \tilde{\psi}_0 > .$$

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.

36 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]).

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 $(\xi_j - \xi_j^0)$:

$$V(\xi_1, \xi_2, ..., \xi_{3N-6}) = V(\xi_1^0, \xi_2^0, ..., \xi_{3N-6}^0) + \frac{1}{2} \sum_{j=1}^{3N-6} \mu_j \omega_j^2 \left(\xi_j - \xi_j^0\right)^2, \tag{606}$$

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:

$$V(\mathbf{r}, \theta, \phi, \mathbf{d}) = V_{str.}(\mathbf{r}) + V_{bend}(\theta) + V_{torsion}(\phi) + V_{non-bond}(\mathbf{d}), \tag{607}$$

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. (607) 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:

$$B^- + HA \rightarrow BH + AY^-. \tag{608}$$

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:

$$\frac{\partial V(r_1, r_2, ..., q_i, ...)}{\partial q_i} = 0, \tag{609}$$

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. (609), 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:

$$V(r_1, r_2, q_3, ..., q_{3N-6}) = V_0(r_1, r_2) + \sum_{j,k=3}^{3N-6} \frac{1}{2} \left(q_j - q_j^0\right) \left(\frac{\partial^2 V}{\partial q_j \partial q_k}\right)_{q_{j/k}=q_{j/k}^0} \left(q_k - q_k^0\right). \tag{610}$$

The potential has no linear term in $(q_k - q_k^0)$ because the first order derivatives are equal to zero (see Eq. (609)) 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

$$\begin{aligned} H(\mathbf{r}, \mathbf{P}_r, \mathbf{q}, \mathbf{P}_q) = & \sum_{j=1}^{2} \frac{\mathbf{P}_r^2}{2\mu_j} + \sum_{j=3}^{3N-6} \frac{\mathbf{P}_q^2}{2\mu_j} \\ & + V_0(\mathbf{r}) + \frac{1}{2} \sum_{j=3}^{3N-6} \sum_{k=3}^{3N-6} (q_j - q_j^0(\mathbf{r})) \frac{\partial^2 V(\mathbf{r}, \mathbf{q})}{\partial q_j \partial q_k} (q_k - q_k^0(\mathbf{r})). \end{aligned} \tag{611}$$

37 Exam 3

Exam 3 CHEM 572a
Advanced Quantum Mechanics

Exercise 1:
(**20 points**) **1.1**: Explain how to use the Hamilton-Jacobi equation to compute the trajectory of coordinates and momenta $q(t)$ and $p(t)$ for a system described by the Hamiltonian

$$H(q,p) = p^2/(2m) + V(q)$$

.
(**20 points**) **1.2**: Derive the equations of motion of the "hidden" variables $q(t)$ and $p(t)$ as described by Bohmian dynamics.
(**15 points**) **1.3**: Explain why the trajectories of hidden variables in the Bohmian formulation of quantum mechanics do not violate the uncertainty principle.
(**15 points**) **1.4**: "Do you think the moon exists only when we look at it?" Explain your answer within the context of the orthodox interpretation of quantum mechanics and the Bohmian interpreation of quantum mechanics.
Exercise 2:
(**15 points**) **2.1**: Define the Wigner-transform of $\Psi_t(x)$ and derive its equation of motion. Explain how to compute the expectation value of $\hat{H} = \hat{p}^2/(2m) + V(\hat{x})$ by using the Wigner transform instead of the wave function.
(**15 points**) **2.2**: Explain how to compute the photoelectron detachment spectrum of I_2^-.

37.1 Answer Key

Exercise 1:
(**20 points**) **1.1**:
Solving the Hamilton-Jacobi equation

$$\frac{\partial S(q,P,t)}{\partial t} + H(q, \frac{\partial S(q,P,t)}{\partial q}) = 0, \tag{612}$$

we can find the characteristic function $S(q,P,t)$. To obtain $p(t)$ we compute the partial derivative of $S(q,P,t)$ with respect to $q(t)$,

$$p(t) = \frac{\partial S(q,P,t)}{\partial q}, \tag{613}$$

and to obtain $q(t)$ we first compute Q as the partial derivative of $S(q,P,t)$ with respect to P,

$$Q = \frac{\partial S(q,P,t)}{\partial P}, \tag{614}$$

and then we solve for $q(t)$ as a function of Q and P. Both Q and P are constant in time since

$$\widetilde{H}(Q,P) = \frac{\partial S(q,P,t)}{\partial t} + H(q, \frac{\partial S(q,P,t)}{\partial q}) = 0, \tag{615}$$

and

$$\begin{aligned} \dot{Q} &= \frac{\partial \widetilde{H}(Q,P)}{\partial P}, \\ \dot{P} &= -\frac{\partial \widetilde{H}(Q,P)}{\partial Q}. \end{aligned} \tag{616}$$

The resulting equations of motion for $p(t)$ and $q(t)$ are Hamilton's equations:

$$\begin{aligned} \dot{q}(t) &= \frac{\partial H(q,p)}{\partial p}, \\ \dot{p}(t) &= -\frac{\partial H(q,p)}{\partial q}. \end{aligned} \tag{617}$$

(**20 points**) **1.2**:

To obtain the equations of motion of the "hidden" variables $p(t)$ and $q(t)$ as described by Bohmian dynamics, we first solve the time-dependent Schrödinger equation,

$$i\hbar \frac{\partial \Psi(q)}{\partial t} = \hat{H}\Psi(q), \tag{618}$$

by substituting $\Psi_t(q)$, in terms of the real amplitude $A_t(q)$ and phase $S_t(q)$ functions,

$$\Psi_t(q) = A_t(q) e^{\frac{i}{\hbar} S_t(q)}, \tag{619}$$

The left hand side of Eq. (618) gives

$$i\hbar \frac{\partial \Psi_t(q)}{\partial t} = \left[i\hbar \frac{\partial A_t(q)}{\partial t} - A_t(q) \frac{\partial S_t(q)}{\partial t} \right] e^{\frac{i}{\hbar} S_t(q)}, \tag{620}$$

and the right hand side of Eq. (618) gives

$$\hat{H}\Psi_t(q) = \left[-\frac{\hbar^2}{2m} \frac{\partial^2 A}{\partial q^2} - \frac{i\hbar}{m} \frac{\partial A}{\partial q} \frac{\partial S}{\partial q} + \frac{1}{2m} A_t(q) \left(\frac{\partial S}{\partial q} \right)^2 + A(q)V(q) \right] e^{\frac{i}{\hbar} S_t(q)}. \tag{621}$$

Considering that the real parts of the r.h.s' of Eqs. (620) and (621) must be equal, we obtain:

$$\frac{\partial S_t(q)}{\partial t} + \frac{1}{2m} \left(\frac{\partial S}{\partial q} \right)^2 + V(q) + V_Q(q,t) = 0, \tag{622}$$

where

$$V_Q(q,t) = -\frac{\hbar^2}{2m} \frac{\partial^2 A_t}{\partial q^2} \frac{1}{A_t(q)}. \tag{623}$$

As shown in (1.2), Eq. (622) is the Hamilton-Jacobi equation for the time-dependent Hamiltonian

$$H(q,p) = \frac{p^2}{2m} + V(q) + V_Q(q,t), \tag{624}$$

with $p = \partial S/\partial q$ and $V_Q(q,t)$ the time-dependent external field potential defined by Eq. (623). The equations of motion of the "hidden" variables $q(t)$ and $p(t)$ are Hamilton's equations (see Eq. 617) with the Hamiltonian defined according to Eq. (624).

(**15 points**) **1.3**:

The orthodox formulation of quantum mechanics centers around the uncertainty principle and assumes that the properties of a system can only be determined by measurements. These involve interactions between the measuring device and the system that collapse quantum states into specific eigenstates of the operator that corresponds to the property being measured. The probabilities of actual experimental results are determined by the wavefunction. This is an auxilary quantity that provides the most complete possible specification of the quantum state of the system before the measurement was performed. The quantum state determines an inherent uncertainty on the precision with which we can conceive properties of the system such as postion and momentum as simultaneously existing quantities. In contrast, the Bohmian formulation assumes that the properties of the system are intrisic to the system and independent of the measurement process. The uncertainty with which we can determine those properties, however, is given by the irreducible disturbance between the measuring device and the system. According to this formulation, the uncertainty on the properties of the system is regarded not as an inherent limitation on the precision with which we can correctly conceive the simultaneous definition of properties, such as position and momenta, but rather as a practical limitation on the precision with which these quantities can simultaneously be measured. Therefore, Bohmian trajectories do not violate the uncertainty principle because the variables $q(t)$ and $p(t)$ are practically "hidden"(not observable) while the uncertainty principle refers to observables –*i.e.*, expectation values of these variables as determined by the averages over the ensemble of Bohmian trajectories. The expectation values satisfy the uncertainty principle since there will always be an irreducible disturbance associated with the measurement, or the initial preparation of the system, with devices that interact with the observed system by means of indivisible quanta. Only if the precise effects of those disturbances could be corrected for, one could determine the "hidden" variables and have simultaneous measurements of momentum and position with unlimited precision.

(**15 points**) **1.4**:

Of course, we all think the moon exists even if we do not observe it. However, as discussed in (1.3), the orthodox interpretation of quantum mechanics assumes that the wavefunction provides the most complete possible specification of the state of the system and only determines the probability of actual experimental results. Within this interpretation, the properties of the moon demonstrating its existance (e.g., coordinates, momentum, etc.) can only be conceived in a probabilistic sense within the context of a process of measurement (i.e., when we use photons to observe it). In contrast to that interpretation, the Bohmian formulation shows that there is an alternative possible interpretation of quantum mechanics where the properties of systems exist regardless of any process of measurement (even when we do not observe them) and the probabilistic aspect of the theory is due to the unavoidable disturbances associated with the process of measurement.

Exercise 2:
(**15 points**) **2.1**:

Given a wavefunction $\Psi_t(x)$, the Wigner transform $\rho_t^W(p,q)$ is defined as follows:

$$\rho_t^W(p,q) \equiv \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} ds\, e^{\frac{i}{\hbar}ps}\Psi_t^*(q+s/2)\Psi_t(q-s/2). \tag{625}$$

The expectation value of $\hat{H} = \hat{p}^2/(2m)+V(\hat{x})$ can be computed by using the Wigner transform, instead of the wave function, as follows:

$$\langle\Psi_t|\hat{H}|\Psi_t\rangle = \int dq \int dp H(q,p)\rho_t^W(p,q), \tag{626}$$

since

$$\begin{aligned}
\int dq \int dp H(q,p)\rho_t^W(p,q) &= \frac{1}{2\pi\hbar}\int dq\int dp\frac{p^2}{2m}\int ds e^{\hat{i}\hbar ps}\Psi_t^*(q+s/2)\Psi_t(q-s/2) \\
&\quad + \frac{1}{2\pi\hbar}\int dp\int ds e^{\hat{i}\hbar ps}\int dq V(q)\Psi_t^*(q+s/2)\Psi_t(q-s/2), \\
&= \int dp\frac{p^2}{2m}\frac{1}{2\pi\hbar}\int dx'\int dx'' e^{\hat{i}\hbar p(x'-x'')}\Psi_t^*(x')\Psi_t(x'') \\
&\quad + \int ds\delta(s)\int dq V(q)\Psi_t^*(q+s/2)\Psi_t(q-s/2), \\
&= \int dp\frac{p^2}{2m}\widetilde{\Psi}_t^*(p)\widetilde{\Psi}_t(p) + \int dq V(q)\Psi_t^*(q)\Psi_t(q), \\
&= \int dp\langle\Psi|\frac{p^2}{2m}|p\rangle\langle p|\Psi_t\rangle + \langle\Psi_t|V(q)|\Psi_t\rangle. \\
&= \langle\Psi|\frac{p^2}{2m}|\Psi_t\rangle + \langle\Psi_t|V(q)|\Psi_t\rangle.
\end{aligned} \tag{627}$$

(**15 points**) **2.2**:

The photoabsorption lineshape (at 0 K) is,

$$\begin{aligned}
I_0(\omega) &= \frac{3}{2\pi\hbar}\int_{-\infty}^{\infty} dt e^{i\omega t}\langle\Phi_0|(\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu}(t))(\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu}(0))|\Phi_0\rangle, \\
&= \frac{3}{2\pi\hbar}\sum_f\int_{-\infty}^{\infty} dt e^{i\omega t}\langle\Phi_0|e^{\frac{i}{\hbar}\hat{H}t}(\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu})e^{-\frac{i}{\hbar}\hat{H}t}|\Phi_f\rangle\langle\Phi_f|(\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu})|\Phi_0\rangle,
\end{aligned} \tag{628}$$

where the final states are of the form

$$|\Phi_f\rangle = |\psi_K\rangle|\chi_{E_K}\rangle, \tag{629}$$

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 $E_f = E_K(\nu) + \epsilon$. Therefore, the photoabsorption

lineshape (at 0 K) for the one-photon photodetachment process is,

$$P_0(\epsilon) = \frac{3}{2\pi\hbar}\sum_K \int dE_K \int_{-\infty}^{\infty} dt e^{i\omega t}\langle\Phi_0|e^{\frac{i}{\hbar}\hat{H}t}(\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu})e^{-\frac{i}{\hbar}(E_K+\epsilon)t}|\Phi_k\rangle\langle\Phi_k|(\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu})|\Phi_0\rangle, \quad (630)$$

and using the relation

$$\int dE_K|\chi_{E_K}\rangle\langle\chi_{E_K}|e^{-\frac{i}{\hbar}E_K t} = e^{-\frac{i}{\hbar}H_K t}, \quad (631)$$

the finite temperature distribution is

$$P_\beta(\epsilon) = \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} dt e^{-\frac{i}{\hbar}\epsilon t}C(t), \quad (632)$$

with

$$C(t) = 3\sum_j \rho_j e^{\frac{i}{\hbar}(E_j+\hbar\omega t)}\sum_K \langle\Phi_j|(\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu})e^{-\frac{i}{\hbar}\hat{H}_K t}(\lambda\overrightarrow{\varepsilon_0}\cdot\hat{\mu})|\Phi_j\rangle. \quad (633)$$

Therefore, according to Eqs. (633) and (632), the calculation of the photoelectron detachment spectrum requires the propagation of the system in the neutral states K of I_2 for time t and the overlap with the initial state of I_2^- for all populated initial states.

38 Semiclassical Dynamics

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

$$G(x) = i\hbar\frac{\partial\psi}{\partial t} + \frac{\hbar^2}{2m}\frac{\partial^2\psi}{\partial x^2} - V(x,t)\psi = 0. \tag{634}$$

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

$$\psi(x,t) = e^{iS(t)/\hbar}\varphi(x,t), \tag{635}$$

with

$$\varphi(x,t) = \pi^{-1/4}\hbar^{-1/4}Q^{-1/2}e^{-\gamma(x-q)^2/(2\hbar)+\frac{i}{\hbar}p(x-q)}, \tag{636}$$

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

A Taylor expansion gives,

$$G(x) = G(q) + G'(q)(x-q) + \frac{1}{2}G''(q)(x-q)^2 + ... \tag{637}$$

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

$$i\hbar\frac{\partial\psi}{\partial t} = \left(-\dot{S} - \frac{i\hbar}{2}\frac{\dot{Q}}{Q} - i\left(\dot{P}Q^{-1} - PQ^{-2}\dot{Q}\right)(x-q)^2/2 + iPQ^{-1}(x-q)\dot{q} - \dot{p}(x-q) + p\dot{q}\right)\psi, \tag{638}$$

and

$$\frac{\hbar^2}{2m}\frac{\partial^2\psi}{\partial x^2} = \frac{\hbar^2}{2m}\left([-PQ^{-1}(x-q)/\hbar + \frac{i}{\hbar}p]^2 - PQ^{-1}/\hbar\right)\psi, \tag{639}$$

we obtain

$$\begin{aligned} G(x) = \Bigg(&-\dot{S} - \frac{i\hbar}{2}\frac{\dot{Q}}{Q} - i\left(\dot{P}Q^{-1} - PQ^{-2}\dot{Q}\right)(x-q)^2/2 + iPQ^{-1}(x-q)\dot{q} - \dot{p}(x-q) + p\dot{q} \\ &+ \frac{\hbar^2}{2m}\left[-PQ^{-1}(x-q)/\hbar + \frac{i}{\hbar}p\right]^2 - \frac{\hbar}{2m}PQ^{-1} - V(x)\Bigg)\psi. \end{aligned} \tag{640}$$

with,

$$G'(x) = G(x)\frac{\psi'}{\psi} + \left(-i\left(\dot{P}Q^{-1} - PQ^{-2}\dot{Q}\right)(x-q) + iPQ^{-1}\dot{q} - \dot{p}\right.$$
$$\left. -\frac{\hbar}{m}\left[-PQ^{-1}(x-q)/\hbar + \frac{i}{\hbar}p\right]PQ^{-1} - V'(x)\right)\psi. \tag{641}$$

and

$$\begin{aligned} G''(x) = {} & G(x)\frac{\psi''}{\psi} + G'(x)\frac{\psi'}{\psi} - G(x)\frac{\psi'^2}{\psi^2} \\ & + \left[G'(x)\frac{1}{\psi} - G(x)\frac{\psi'}{\psi^2}\right]\psi' \\ & + \left(-i\left(\dot{P}Q^{-1} - PQ^{-2}\dot{Q}\right)\right. \\ & \left. + \frac{1}{m}\left[PQ^{-1}\right]^2 - V''(x)\right)\psi. \end{aligned} \tag{642}$$

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

$$G'(q) = \left(iPQ^{-1}\dot{q} - \dot{p} - i\frac{p}{m}PQ^{-1} - V'(q)\right)\psi(q) = 0. \tag{643}$$

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,

$$\begin{aligned} \dot{q} &= \frac{p}{m}, \\ \dot{p} &= -V'(q). \end{aligned} \tag{644}$$

In addition,

$$G(q) = \left(-\dot{S} - \frac{i\hbar}{2}\frac{\dot{Q}}{Q} - PQ^{-1}\frac{\hbar}{2m} - V(q) - \frac{p^2}{2m} + p\dot{q}\right)\psi(q) = 0, \tag{645}$$

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

$$\begin{aligned} \dot{S} &= p\dot{q} - \left(V(q) + \frac{p^2}{2m}\right), \\ \dot{Q} &= i\frac{P}{m}. \end{aligned} \tag{646}$$

Finally,

$$G''(q) = \left(-i\left(\dot{P}Q^{-1} - PQ^{-2}\dot{Q}\right) + \frac{1}{m}\left[PQ^{-1}\right]^2 - V''(q)\right)\psi(q) = 0, \tag{647}$$

which is verified when

$$\dot{P} = iV''(q)Q. \tag{648}$$

39 Semiclassical Dynamics in the Gaussian-Hermite Basis

Consider the time-dependent Schrödinger equation,

$$i\hbar\frac{\partial\psi}{\partial t} = \frac{\hat{p}^2}{2m}\psi + V(x,t)\psi, \tag{649}$$

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. (649) can be written, as follows:

$$\psi_0(x,t) = e^{iS(t)/\hbar}\varphi_0(x,t), \tag{650}$$

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

$$\varphi_\nu(x,t) = H_\nu(\hbar^{-1/2}|Q|^{-1}(x-q))A(Q,\nu)e^{-\gamma(x-q)^2/(2\hbar)+\frac{i}{\hbar}p(x-q)}, \tag{651}$$

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,

$$A(Q,\nu) = 2^{-\nu/2}(\nu!)^{-1/2}\pi^{-1/4}\hbar^{-1/4}Q^{-(\nu+1)/2}\bar{Q}^{\nu/2}, \tag{652}$$

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

$$\varphi_0(x,t) = \pi^{-1/4}\hbar^{-1/4}Q^{-1/2}e^{-\gamma(x-q)^2/(2\hbar)+\frac{i}{\hbar}p(x-q)}. \tag{653}$$

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

$$\begin{aligned}\dot{S} &= \frac{p^2}{2m} - V(q,t),\\ \dot{q} &= \frac{p}{m},\\ \dot{p} &= -\frac{\partial V}{\partial q}.\end{aligned} \tag{654}$$

Finally, the equations of motion for Q and P are obtained by substituting $\psi_0(x,t)$ into Eq. (649):

$$\begin{aligned}\dot{Q} &= i\frac{P}{m},\\ \dot{P} &= i\frac{\partial^2 V}{\partial q^2}Q,\end{aligned} \tag{655}$$

or,

$$\begin{aligned}\ddot{Q} &= -\frac{1}{m}\frac{\partial^2 V}{\partial q^2}Q\\ &= -\frac{k(t)}{m}Q,\\ &= -\omega(t)^2 Q,\end{aligned} \tag{656}$$

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:

$$\begin{aligned} Q(t) &= \frac{\partial q(t)}{\partial q(0)} Q(0) + i \frac{\partial q(t)}{\partial p(0)} P(0), \\ P(t) &= \frac{\partial p(t)}{\partial p(0)} P(0) - i \frac{\partial p(t)}{\partial q(0)} Q(0), \end{aligned} \tag{657}$$

since Q and P, defined according to Eqs. (657), satisfy Eqs. (655). According to Eqs. (657), Q and P satisfy the following relation:

$$\bar{Q}P + \bar{P}Q = 2, \tag{658}$$

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, $\gamma(t) = |Q(t)|^{-2} + i\mathrm{Im}[P(t)/Q(t)]$. Note that, according Eqs. (655), $Q(t)$ 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)]:

$$\begin{aligned} Ł_+ &= (2\hbar)^{-1/2}(\bar{P}(x-q) - i\bar{Q}(\hat{p}-p)), \\ Ł_- &= (2\hbar)^{-1/2}(P(x-q) + iQ(\hat{p}-p)), \end{aligned} \tag{659}$$

and noting that

$$\begin{aligned} [Ł_-, Ł_+] &= \mathbf{1}, \\ \frac{1}{2}(Ł_- Ł_+ + Ł_+ Ł_-)\psi_\nu &= \left(\nu + \frac{1}{2}\right)\psi_\nu, \end{aligned} \tag{660}$$

we see that

$$Ł_+ \psi_\nu = \sqrt{\nu+1}\psi_{\nu+1}. \tag{661}$$

Therefore,

$$\psi_\nu(x,t) = Ł_+^\nu \psi_0(x;t) = \sqrt{\nu!}e^{iS(t)/\hbar}\varphi_\nu(x,t), \tag{662}$$

with $\nu = 0, 1, 2, \ldots$ are also solutions of Eq. (649).

39.1 Multidimentional Semiclassical Dynamics

The 2-dimensional generalization of Eq. (653) is:

$$\varphi_0(\mathbf{x},t) = \pi^{-n/4}\hbar^{-n/4}[\det(\mathbf{Q})]^{-1/2}e^{-(\mathbf{x}-\mathbf{q})\cdot\mathbf{P}\mathbf{Q}^{-1}\cdot(\mathbf{x}-\mathbf{q})/(2\hbar)+\frac{i}{\hbar}\mathbf{p}\cdot(\mathbf{x}-\mathbf{q})}, \tag{663}$$

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

$$\begin{aligned} \dot{\mathbf{Q}} &= i\frac{\mathbf{P}}{m}, \\ \dot{\mathbf{P}} &= i\mathbf{k}\mathbf{Q}, \end{aligned} \tag{664}$$

or,

$$\ddot{\mathbf{Q}} = -\frac{1}{m}\mathbf{k}\mathbf{Q}, \tag{665}$$

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. (663), we use the log-derivative substitution:

$$\begin{aligned} \mathbf{R} &= \frac{\partial}{\partial t}\log[\mathbf{Q}], \\ &= \dot{\mathbf{Q}}\mathbf{Q}^{-1}, \end{aligned} \tag{666}$$

giving,

$$\dot{\mathbf{Q}} = \mathbf{R}\mathbf{Q}. \tag{667}$$

Integrating Eq. (667), we obtain:

$$\begin{aligned} \mathbf{Q} &= \exp\int_0^t dt'\mathbf{R}(t'), \\ &= \prod_{k=1}^{n} e^{\Delta t \mathbf{R}_k}. \end{aligned} \tag{668}$$

Therefore,

$$\begin{aligned} \det(\mathbf{Q}) &= \prod_{k=1}^{n} \det\left[e^{\Delta t \mathbf{R}_k}\right], \\ &= \prod_{k=1}^{n} e^{\Delta t \ \det[\mathbf{R}_k]}, \\ &= e^{\Delta t \sum_{k=1}^{n} \mathrm{Tr}[\mathbf{R}_k]}, \\ &= e^{\Delta t \int_0^t dt' \mathrm{Tr}[\mathbf{R}(t')]}. \end{aligned} \tag{669}$$

The equation of motion for $\mathbf{R}$ is obtained from Eq. (666), as follows:

$$\dot{\mathbf{R}} = \ddot{\mathbf{Q}}\mathbf{Q}^{-1} - \left(\dot{\mathbf{Q}}\mathbf{Q}^{-1}\right)^2. \tag{670}$$

Substituting Eqs. (666) and (656) into Eq. (670), we obtain:

$$\dot{\mathbf{R}} = -\omega(t)^2 - \mathbf{R}^2. \tag{671}$$

39.2 Log Derivative Propagation

For constant ω, the propagation of $\mathbf{R} = \dot{\mathbf{Q}}\mathbf{Q}^{-1}$ can be based on Eq. (665):

$$\ddot{\mathbf{Q}} = -\frac{\mathbf{k}}{m}\mathbf{Q}, \tag{672}$$

with

$$\begin{aligned}
\mathbf{Q}(t) &= \mathbf{Q}(0)\cos\left(\sqrt{\frac{\mathbf{k}}{m}}t\right) + \frac{\dot{\mathbf{Q}}(0)}{\sqrt{\frac{\mathbf{k}}{m}}}\sin\left(\sqrt{\frac{\mathbf{k}}{m}}t\right), \\
\dot{\mathbf{Q}}(t) &= \dot{\mathbf{Q}}(0)\cos\left(\sqrt{\frac{\mathbf{k}}{m}}t\right) - \mathbf{Q}(0)\sqrt{\frac{\mathbf{k}}{m}}\sin\left(\sqrt{\frac{\mathbf{k}}{m}}t\right).
\end{aligned} \tag{673}$$

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

$$\mathbf{R}(t) = \left(\mathbf{R}(0)\cos[\omega t] - \omega\sin[\omega t]\right)\left(\cos[\omega t] + \mathbf{R}(0)\sin[\omega t]/\omega\right)^{-1}. \tag{674}$$

When ω changes slowly with time,

$$\mathbf{R}(t+\delta) = \left(\mathbf{R}(t)\cos[\omega(t)\delta] - \omega(t)\sin[\omega(t)\delta]\right)\left(\cos[\omega(t)\delta] + \mathbf{R}(t)\sin[\omega(t)\delta]/\omega(t)\right)^{-1}. \tag{675}$$

When the instantaneous normal modes (*i.e.*, eigenvectors of $\mathbf{K}$) are approximately constant, we can solve Eq. (672) by transforming it into a problem of n 1-dimensional equations, as follows [[*J. Chem. Phys.* (1999) **110**:9922-9936]]:

$$\begin{aligned}
\dot{\tilde{\mathbf{Q}}}(j,j) &= i\frac{\tilde{\mathbf{P}}(j,j)}{m}, \\
\dot{\tilde{\mathbf{P}}}(j,j) &= i\tilde{k}(j)\tilde{\mathbf{Q}}(j,j),
\end{aligned} \tag{676}$$

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

$$\mathbf{L}^{\dagger} \cdot \mathbf{K}(t) \cdot \mathbf{L} = m\tilde{\omega}(t)^2. \tag{677}$$

where $\tilde{\omega}$ is a diagonal matrix. The new variables $\tilde{\mathbf{Q}}$ and $\tilde{\mathbf{P}}$, introduced by Eq. (676), are defined according to the analogous transformations,

$$\begin{aligned}
\tilde{\mathbf{Q}}(t) &= \mathbf{L}^{\dagger} \cdot \mathbf{Q}(t) \cdot \mathbf{L}, \\
\tilde{\mathbf{P}}(t) &= \mathbf{L}^{\dagger} \cdot \mathbf{P}(t) \cdot \mathbf{L}.
\end{aligned} \tag{678}$$

Assuming that $\mathbf{L}$ are approximately constant, we obtain Eq. (676) by computing the time-derivative of Eqs. (678) and substituting $\dot{\mathbf{Q}}$ and $\dot{\mathbf{P}}$ according to Eqs. (664). Note that $\det(\mathbf{Q}) = \det(\tilde{\mathbf{Q}})$ since the $\det(\mathbf{L}) = \det(\mathbf{L}^{\dagger}) = 1$.

An approximate solution of Eqs. (676) could also be obtained by solving them according to the WKB approximation, as follows. Combining the Eqs. (676), we obtain:

$$\ddot{\tilde{\mathbf{Q}}}(j,j) = -\tilde{\omega}(j)^2 \tilde{\mathbf{Q}}(j,j), \tag{679}$$

The possible WKB solutions,

$$\tilde{\mathbf{Q}}_a(j,j;t) = \tilde{\mathbf{Q}}_a(j,j;0)\text{exp}\left(\pm i \int_0^t \tilde{\omega}(j;t')dt'\right), \tag{680}$$

satisfy Eq. (679), as follows:

$$\ddot{\tilde{\mathbf{Q}}}_a(j,j;t) = -\tilde{\omega}(j;t)^2 \tilde{\mathbf{Q}}_a(j,j;t) + \Delta, \tag{681}$$

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:

$$\begin{aligned}
\tilde{\mathbf{Q}}_a(j,j;t) &= \tilde{\mathbf{Q}}_a(j,j;0)\text{cos}\left(\int_0^t \tilde{\omega}(j;t')dt'\right) + \frac{\dot{\tilde{\mathbf{Q}}}(0)}{\tilde{\omega}(j;t)}\text{sin}\left(\int_0^t \tilde{\omega}(j;t')dt'\right), \\
\dot{\tilde{\mathbf{Q}}}_a(j,j;t) &= \dot{\tilde{\mathbf{Q}}}_a(j,j;0)\text{cos}\left(\int_0^t \tilde{\omega}(j;t')dt'\right) - \tilde{\mathbf{Q}}_a(j,j;0)\tilde{\omega}(j;t')\text{sin}\left(\int_0^t \tilde{\omega}(j;t')dt'\right).
\end{aligned} \tag{682}$$

39.3 Normalization of Multidimensional Gaussians

The goal of this section is show that the multidimensional Gaussian, introduced by Eq. (663), is normalized as follows:

$$\begin{aligned}
I_2 &= \int dx_1 dx_2 \, e^{-c_{11}x_1^2 - c_{12}x_1x_2 - c_{21}x_2x_1 - c_{22}x_2^2}, \\
&= \int dx_1 dx_2 \, e^{-(x_1 x_2)\begin{pmatrix} c_{11}c_{12} \\ c_{21}c_{22} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}}, \\
&= \int d\mathbf{x} \, e^{-\mathbf{x}^T \cdot \mathbf{c} \cdot \mathbf{x}}, \\
&= \sqrt{\frac{\pi^2}{det(\mathbf{c})}}.
\end{aligned} \tag{683}$$

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

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \Gamma_{11}\Gamma_{12} \\ \Gamma_{21}\Gamma_{22} \end{pmatrix}\begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix}, \tag{684}$$

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

$$\begin{pmatrix} c_{11}c_{12} \\ c_{21}c_{22} \end{pmatrix} \begin{pmatrix} \Gamma_{11}\Gamma_{12} \\ \Gamma_{21}\Gamma_{22} \end{pmatrix} = \begin{pmatrix} \Gamma_{11}\Gamma_{12} \\ \Gamma_{21}\Gamma_{22} \end{pmatrix} \begin{pmatrix} \lambda_1 0 \\ 0\lambda_2 \end{pmatrix}. \tag{685}$$

Substituting Eq. (684) into Eq. (683), we obtain:

$$\begin{aligned} I_2 &= \int d\xi_1 d\xi_2 \left| \frac{\partial(x_1, x_2)}{\partial(\xi_1, \xi_2)} \right| e^{-(\xi_1 \xi_2) \begin{pmatrix} \lambda_1 0 \\ 0\lambda_2 \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix}}, \\ &= \int d\xi_1 d\xi_2 \, e^{-\lambda_1 \xi_1^2 - \lambda_2 \xi_2^2}, \\ &= \sqrt{\frac{\pi^2}{\lambda_1 \lambda_s}} = \sqrt{\frac{\pi^2}{det(\lambda)}}. \end{aligned} \tag{686}$$

Furthermore, according to Eq. (685),

$$\begin{aligned} det(\lambda) &= det(\Gamma)^{-1} det(\mathbf{c}) det(\Gamma) \\ &= det(\mathbf{c}), \end{aligned} \tag{687}$$

Therefore, substituting Eq. (687) into Eq. (686), we obtain Eq. (683).
Note: The Jacobian of the orthogonal transformation, introduced by Eq. (684), is equal to 1 since

$$\begin{aligned} \left| \frac{\partial(x_1, x_2)}{\partial(\xi_1, \xi_2)} \right| &= \begin{vmatrix} \frac{\partial x_1}{\partial \xi_1} \frac{\partial x_1}{\partial \xi_2} \\ \frac{\partial x_2}{\partial \xi_1} \frac{\partial x_2}{\partial \xi_2} \end{vmatrix} \\ &= det(\Gamma) \\ &= 1. \end{aligned} \tag{688}$$

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

$$\begin{aligned} 1 &= det(\Gamma^{-1}\Gamma), \\ &= det(\Gamma^{-1}) det(\Gamma). \end{aligned} \tag{689}$$

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

$$det(\Gamma) = \frac{1}{det(\Gamma)}, \tag{690}$$

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

40 Solutions to Computational Assignments

40.1 Problem 1

Computational Problem 1: Write a computer program to represent the wave-packet, introduced by Eq. (4) 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

```
f77 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
```

Download from (http://ursula.chem.yale.edu/~batista/classes/summer/P1/Problem1.f),

```
      PROGRAM Problem_1
      call Initialize()
      CALL SAVEWF(0)
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
```

```
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

40.2 Problem 2

Computational Problem 2: Write a computer program to represent the initial state, introduced by Eq. (4), 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

```
f77 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
```

Download from (http://ursula.chem.yale.edu/~batista/classes/summer/P2/Problem2.f),

```
      PROGRAM Problem2
      call Initialize()
      CALL SAVEFT()
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
```

```
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutines 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
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

40.3 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. (10) for the initial wave-packet, introduced by Eq. (4), 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

```
f77 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 $\langle\Psi_t|\hat{x}|\Psi_t\rangle$.

Download from (http://ursula.chem.yale.edu/~batista/classes/summer/P3/Problem3.f),

```
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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)
```

```
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

40.4 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. (4), 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. (10) 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

```
f77 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.

Download from (http://ursula.chem.yale.edu/~batista/classes/summer/P4/Problem4.f),

```
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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)
```

```
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Pp(pe)
```

```
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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutines 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)
         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
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

40.5 Problem 5

Computational Problem 5: Expand the exponential operators in both sides of Eq. (28) 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:

$$e^{-i\hat{H}\tau} = 1 - i\hat{H}\tau - \frac{1}{2}\hat{H}^2\tau^2 + O(\tau^3), \tag{691}$$

where $\hat{H} = \hat{p}^2/(2m) + \hat{V}$. Therefore,

$$e^{-i\hat{H}\tau} = 1 - i\hat{H}\tau - \frac{1}{2}\frac{\hat{p}^4}{4m^2}\tau^2 - \frac{1}{2}\hat{V}^2\tau^2 - \frac{1}{2}\frac{\hat{p}^2}{2m}\hat{V}\tau^2 - \frac{1}{2}\hat{V}\frac{\hat{p}^2}{2m}\tau^2 + O(\tau^3), \tag{692}$$

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. (692).

Expanding the right-hand-side (r.h.s.) of Eq. (18) gives,

$$\begin{aligned} e^{-iV(\hat{x})\tau/2}e^{-i\hat{p}^2\tau/(2m)}e^{-iV(\hat{x})\tau/2} = &\left(1 - i\hat{V}\tau/2 - \frac{1}{2}\hat{V}^2\tau^2/4 + O(\tau^3)\right)\left(1 - i\frac{\hat{p}^2}{2m}\tau - \frac{1}{2}\frac{\hat{p}^4}{4m^2}\tau^2 + O(\tau^3)\right) \\ &\times \left(1 - i\hat{V}\tau/2 - \frac{1}{2}\hat{V}^2\tau^2/4 + O(\tau^3)\right), \end{aligned} \tag{693}$$

$$\begin{aligned} e^{-iV(\hat{x})\tau/2}e^{-i\hat{p}^2\tau/(2m)}e^{-iV(\hat{x})\tau/2} = &\left(1 - i\hat{V}\tau/2 - \frac{1}{2}\hat{V}^2\tau^2/4 - i\frac{\hat{p}^2}{2m}\tau - \hat{V}\frac{\hat{p}^2}{2m}\tau^2/2 - \frac{1}{2}\frac{\hat{p}^4}{4m^2}\tau^2 + O(\tau^3)\right) \\ &\times \left(1 - i\hat{V}\tau/2 - \frac{1}{2}\hat{V}^2\tau^2/4 + O(\tau^3)\right), \end{aligned} \tag{694}$$

$$\begin{aligned} e^{-iV(\hat{x})\tau/2}e^{-i\hat{p}^2\tau/(2m)}e^{-iV(\hat{x})\tau/2} = &1 - i\hat{V}\tau/2 - \frac{1}{2}\hat{V}^2\tau^2/4 - i\frac{\hat{p}^2}{2m}\tau - \hat{V}\frac{\hat{p}^2}{2m}\tau^2/2 - \frac{1}{2}\frac{\hat{p}^4}{4m^2}\tau^2 \\ &- i\hat{V}\tau/2 - \hat{V}^2\tau^2/4 - \frac{\hat{p}^2}{2m}\hat{V}\tau^2/2 - \frac{1}{2}\hat{V}^2\tau^2/4 + O(\tau^3), \end{aligned} \tag{695}$$

$$\begin{aligned} e^{-iV(\hat{x})\tau/2}e^{-i\hat{p}^2\tau/(2m)}e^{-iV(\hat{x})\tau/2} = &1 - i\hat{V}\tau - i\frac{\hat{p}^2}{2m}\tau - \frac{1}{2}\hat{V}^2\tau^2 - \hat{V}\frac{\hat{p}^2}{2m}\tau^2/2 - \frac{1}{2}\frac{\hat{p}^4}{4m^2}\tau^2 \\ &- \frac{\hat{p}^2}{2m}\hat{V}\tau^2/2 + O(\tau^3). \end{aligned} \tag{696}$$

Note that the r.h.s. of Eq. (696) is identical to the r.h.s. of E. (692), completing the proof that the Trotter expansion, introduced by Eq. (18), is accurate to second order in τ.

40.6 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. (10). Compare the resulting propagated state with the analytic solution obtained by substituting Eq. (26) into Eq. (25).

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

```
f77 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
```

Download from (http://ursula.chem.yale.edu/~batista/classes/summer/P6/Problem6.f),

```
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
```

```
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       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
```

```
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
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)
```

```
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

40.7 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. (26) into Eq. (25). 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:

$$\begin{aligned} p_{j+1} &= p_j + (F(x_j) + F(x_{j+1}))\tau/2 \\ x_{j+1} &= x_j + p_j\tau/m + F(x_j)\tau^2/(2m). \end{aligned} \tag{697}$$

In order to visualize the output of this program, cut the source code attached below, compile it by typing

```
f77 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/classes/summer/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
```

Download from (http://ursula.chem.yale.edu/~batista/classes/summer/P7/Problem7.f)

```
      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
```

```
      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
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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
```

```
      END DO
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutine for FFT from Numerical Recipes
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
      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
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

40.8 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

```
f77 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/classes/summer/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
```

Download from (http://ursula.chem.yale.edu/~batista/classes/summer/P8/Problem8.f)

```
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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=-5.0
      xmax=25.0
      dt=0.2
      rmass=1.0
      xk=-.5
      pk=0.0
      nstep=100
      ndump=1
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
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=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
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
```

```
      CALL VA(VPOT1,x)
      CRV(1,1)=VPOT1
c
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
      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)
```

```
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutine for FFT from Numerical Recipes
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

40.9 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 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.

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

```
f77 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/classes/summer/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
```

Download from (http://ursula.chem.yale.edu/~batista/classes/summer/P9/Problem9.f)

```
      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
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
      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
```

```
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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,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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
```

```
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutine for FFT from Numerical Recipes
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

40.10 Problem 10

Problem 10: (a) Derive Eq. (50) by considering that,

$$e^{-i\mathbf{V}_c 2\tau} = \mathbf{D}^\dagger \begin{pmatrix} e^{iV_c(\mathbf{x})2\tau} & 0 \\ 0 & e^{-iV_c(\mathbf{x})2\tau} \end{pmatrix} \mathbf{D}, \tag{698}$$

with

$$\mathbf{D} = \mathbf{D}^\dagger \equiv \begin{pmatrix} -1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}, \tag{699}$$

since

$$e^{-i\mathbf{V}_c 2\tau} = \mathbf{1} + (-i\mathbf{V}_c 2\tau) + \frac{1}{2!}(-i\mathbf{V}_c 2\tau)^2 + ..., \tag{700}$$

and

$$\mathbf{V}_c \equiv \begin{pmatrix} 0 & V_c(\mathbf{x}) \\ V_c(\mathbf{x}) & 0 \end{pmatrix} = \mathbf{D}^\dagger \begin{pmatrix} -V_c(\mathbf{x}) & 0 \\ 0 & V_c(\mathbf{x}) \end{pmatrix} \mathbf{D}, \tag{701}$$

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

In order to derive Eq. (28) we need to prove the following equation:

$$e^{-iV_0\tau} e^{-iV_c 2\tau} e^{-iV_0\tau} = \begin{pmatrix} e^{-iV_1(\mathbf{x})2\tau}\cos(2V_c(\mathbf{x})\tau) & -i\,\sin(2V_c(\mathbf{x})\tau)\, e^{-i(\hat{V}_1(\mathbf{x})+\hat{V}_2(\mathbf{x}))\tau} \\ -i\,\sin(2V_c(\mathbf{x})\tau)\, e^{-i(V_1(\mathbf{x})+\hat{V}_2(\mathbf{x}))\tau} & \cos(2V_c(\mathbf{x})\tau)\, e^{-iV_2(\mathbf{x})2\tau} \end{pmatrix}. \tag{702}$$

where

$$e^{-iV_0\tau} = e^{-i\begin{pmatrix} V_1(\mathbf{x}) & 0 \\ 0 & V_2(\mathbf{x}) \end{pmatrix}\tau}. \tag{703}$$

Expanding the exponential on the r.h.s. of Eq. (703) gives

$$e^{-i\tau\begin{pmatrix} V_1(\mathbf{x}) & 0 \\ 0 & V_2(\mathbf{x}) \end{pmatrix}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} -i\tau V_1(\mathbf{x}) & 0 \\ 0 & -i\tau V_2(\mathbf{x}) \end{pmatrix} + \begin{pmatrix} \frac{1}{2!}V_1(\mathbf{x})^2(-i\tau)^2 & 0 \\ 0 & \frac{1}{2!}V_2(\mathbf{x})^2(-i\tau)^2 \end{pmatrix} + ... \tag{704}$$

Therefore,

$$e^{-i\tau\begin{pmatrix} V_1(\mathbf{x}) & 0 \\ 0 & V_2(\mathbf{x}) \end{pmatrix}} = \begin{pmatrix} e^{-iV_1(\mathbf{x})\tau} & 0 \\ 0 & e^{-iV_2(\mathbf{x})\tau} \end{pmatrix}. \tag{705}$$

In addition, according to Eq. (30),

$$e^{-i\mathbf{V}_c 2\tau} = \mathbf{D}^\dagger \begin{pmatrix} e^{iV_c(\mathbf{x})2\tau} & 0 \\ 0 & e^{-iV_c(\mathbf{x})2\tau} \end{pmatrix} \mathbf{D}, \tag{706}$$

with

$$\mathbf{D} = \mathbf{D}^\dagger \equiv \begin{pmatrix} -1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}, \tag{707}$$

Therefore,

$$e^{-iV_0\tau}e^{-iV_c 2\tau}e^{-iV_0\tau} = \begin{pmatrix} e^{-iV_1(\mathbf{x})\tau} & 0 \\ 0 & e^{-iV_2(\mathbf{x})\tau} \end{pmatrix} \mathbf{D}^\dagger \begin{pmatrix} e^{iV_c(\mathbf{x})2\tau} & 0 \\ 0 & e^{-iV_c(\mathbf{x})2\tau} \end{pmatrix} \mathbf{D} \begin{pmatrix} e^{-iV_1(\mathbf{x})\tau} & 0 \\ 0 & e^{-iV_2(\mathbf{x})\tau} \end{pmatrix}. \tag{708}$$

The multiplication of the five matrices on the r.h.s. of Eq. (708) gives the matrix on the r.h.s. of Eq.(702).

40.11 Problem 11

Problem 11: Derive Eq. (49) by writing the matrix V in the basis of adiabatic eigenstates

$$\begin{aligned}\phi_1(x) &= L_{11}(x)|1\rangle + L_{21}(x)|2\rangle,\\ \phi_2(x) &= L_{12}(x)|1\rangle + L_{22}(x)|2\rangle,\end{aligned} \tag{709}$$

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

$$e^{-i\mathbf{V}2\tau} = \mathbf{1} + (-i\mathbf{V}2\tau) + \frac{1}{2!}(-i\mathbf{V}2\tau)^2 + ..., \tag{710}$$

show that in the adiabatic representation

$$e^{-i\mathbf{V}2\tau} = \begin{pmatrix} e^{-iE_1(x)2\tau} & 0 \\ 0 & e^{-iE_2(x)2\tau} \end{pmatrix}. \tag{711}$$

Finally, show that the diagonal matrix introduced by Eq. (711) can be rotated to the representation of diabatic states $|1\rangle$, $|2\rangle$ according to the similarity transformation

$$\mathbf{L}^{-1} \begin{pmatrix} e^{-iE_1(x)\tau} & 0 \\ 0 & e^{-iE_2(x)\tau} \end{pmatrix} \mathbf{L}. \tag{712}$$

According to the definition of the eigenstates of the potential energy matrix, given by Eq. (34),

$$\begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} = \begin{pmatrix} E_1 & 0 \\ 0 & E_1 \end{pmatrix} \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix}, \tag{713}$$

and

$$\begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \begin{pmatrix} L_{12} \\ L_{22} \end{pmatrix} = \begin{pmatrix} E_2 & 0 \\ 0 & E_2 \end{pmatrix} \begin{pmatrix} L_{12} \\ L_{22} \end{pmatrix}. \tag{714}$$

Therefore,

$$\begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \begin{pmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix}, \tag{715}$$

and

$$\begin{pmatrix} L_{11} & L_{21} \\ L_{12} & L_{22} \end{pmatrix} \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \begin{pmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{pmatrix} = \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix}, \tag{716}$$

or

$$\begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix}, \begin{pmatrix} L_{11} & L_{21} \\ L_{12} & L_{22} \end{pmatrix}. \tag{717}$$

Therefore, defining

$$\mathbf{L} = \begin{pmatrix} L_{11} & L_{21} \\ L_{12} & L_{22} \end{pmatrix}, \tag{718}$$

gives

$$\begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} = \mathbf{L}^{-1} \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix} \mathbf{L}. \tag{719}$$

Exponentiating both sides of Eq. (719), gives

$$e^{-i\tau \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix}} = e^{-i\tau \mathbf{L}^{-1} \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix} \mathbf{L}}. \tag{720}$$

Expanding the r.h.s. of Eq. (720) gives,

$$e^{-i\tau \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \mathbf{L}^{-1} \begin{pmatrix} -i\tau E_1 & 0 \\ 0 & -i\tau E_2 \end{pmatrix} \mathbf{L} + \mathbf{L}^{-1} \begin{pmatrix} \frac{1}{2!} E_1^2 (-i\tau)^2 & 0 \\ 0 & \frac{1}{2!} E_2^2 (-i\tau)^2 \end{pmatrix} \mathbf{L} + ..., \tag{721}$$

since $\mathbf{L}^{-1}\mathbf{L} = 1$. Therefore,

$$e^{-i\tau \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix}} = \mathbf{L}^{-1} \begin{pmatrix} e^{-iE_1(\mathbf{x})\tau} & 0 \\ 0 & e^{-iE_2(\mathbf{x})\tau} \end{pmatrix} \mathbf{L}. \tag{722}$$

40.12 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. (49). 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. (4). Use $x_0 = -2.2$, $p_0 = 0$, $m = 1$, $\omega = 1$ and two coupled potential energy surfaces described by the potential energy matrix

$$V = \begin{pmatrix} V_1(x) & \delta \\ \delta & V_2(x) \end{pmatrix}, \tag{723}$$

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. (723), 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

```
f77 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/classes/summer/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
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
```

Download from (http://ursula.chem.yale.edu/~batista/classes/summer/P12/P12_c/Problem12.f)

```
      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.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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
```

```
      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=100
      ndump=1
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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)
c
         chi0(kk,1)=chi(kk,1)
         chi0(kk,2)=chi(kk,2)
      end do
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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=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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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,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)
```

```
     1           +vprop(kk,i,2)*chin2(in)
            END DO
          END DO
          END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutines from Numerical Recipes to compute FFT
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
c     Subroutines to compute eigenvalues and eigenvectors
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
            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
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      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
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      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
```

40.13 Problem 12p

Problem 12p:

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

```
f77 Problem12p.f -o Problem12p
```

run it by typing

```
./Problem12p
```

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/classes/summer/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
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
```

Download from (http://ursula.chem.yale.edu/~batista/classes/summer/P12/P12_c/Problem12p.f)

```
      PROGRAM Problem12p
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
      INTEGER istep,nstep,iseed
      REAL dt,rn
      COMPLEX vprop,tprop,energy
      PARAMETER(npts=9,nptx=2**npts,NN=2,ntraj=200)
      DIMENSION vprop(nptx,NN,NN),tprop(nptx)
      COMMON /ENER/ energy(NN)
c
      iseed=912371
      call srand(iseed)
      CALL ReadParam(nstep,ndump,dt)

      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.GE.1) CALL PROPAGATE(vprop,tprop,dt)
            IF(mod((istep-1),ndump).EQ.0) THEN
               CALL ACCUM(istep,ndump,dt)
            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
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=100
      ndump=1
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE Initialize(kt)

      IMPLICIT NONE
      INTEGER NN,nptx,npts,kk,counter,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),counter(200)
      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
            counter(kk)=0
         END DO
      END IF
c
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE SAVEWF(je2,ndump,dt)
c
c     Dump Time Evolved Wave packet
c
      IMPLICIT NONE
      INTEGER je2,nptx,npts,kk,NN,ncount,ndump,jj,counter
      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),counter(200)
      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/counter(jj)+real(energy(1))
      end do
      write(1,33)

      do kk=1,nptx
         x=xmin+kk*dx
         r2=abs(c1(kk,jj,2))
         write(1,33) x,r2/counter(jj)+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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE ACCUM(je2,ndump,dt)
```

```
c
c     Accumulate Time Evolved Wave packet
c
      IMPLICIT NONE
      INTEGER je2,nptx,npts,kk,NN,ncount,ndump,jj,counter,ns
      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),counter(200)
      common /xy/  xmin,xmax
      COMMON / wfunc/ chi(nptx,NN)
      COMMON /ENER/ energy(NN)
      COMMON / OCCUP/ ns
c
      jj=je2/ndump
      counter(jj)=counter(jj)+1
c
c     Accumulate Wave-packet components
c
      do kk=1,nptx
         c1(kk,jj,ns)=c1(kk,jj,ns)+chi(kk,ns)
      end do
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PROPAGATE(vprop,tprop,dt)
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,NF,ns,ns_n,ns_o
      COMPLEX chi,vprop,chin,tprop,eye,rc
      REAL cs,si,dt,rn
      PARAMETER(npts=9,nptx=2**npts,NN=2)
      DIMENSION chin(nptx)
      DIMENSION tprop(nptx),vprop(nptx,NN,NN)
      COMMON / wfunc/ chi(nptx,NN)
      COMMON / OCCUP/ ns
c
      eye=(0.0,1.0)
c
c     Stochastic Jump
c
      NF=0
      cs=cos(0.3*dt)
      si=sin(0.3*dt)
      rc=cs+si
      rn=rand()*rc
      IF(rn.LE.cs)  NF=1        ! flag for adiabatic dynamics
      ns_n=ns     ! new surface index
      ns_o=ns     ! old surface index
      IF(NF.EQ.0) THEN
         rc=-eye*rc
         ns_o = ns
         IF(ns_o.EQ.1) THEN
            ns_n = 2
         ELSE
            ns_n = 1
         END IF
         ns=ns_n
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutines from Numerical Recipes to compute FFT
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
c     Subroutines to compute eigenvalues and eigenvectors
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
            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
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      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
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      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
```

40.14 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. (10) 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
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)=8.
      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
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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       FUNCTION V(r)
       implicit real *8(a-h,o-z)
       V=0.5*1.*(r-4.)**2
       RETURN
       END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       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
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)
      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
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
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 dvrho1.f, compile it by typing

```
f77 dvrho1.f -o dvrho1
```

and run it by typing

```
./dvrho1
```

Then, cut the script attached below, save it with the name scr_ho1 in the same directory where you run your code, and visualize the 4 eigenstates by typing

```
gnuplot<scr_ho1
```

where the file named scr_ho1 has the following lines:

Download from (http://ursula.chem.yale.edu/∼batista/classes/v572/scr_ho1)

```
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
```

40.15 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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       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
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutines to compute eigenvalues and eigenvectors from NR
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
cccccccccccccccccccccccccccccccccccccccccccccccccc
      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
cccccccccccccccccccccccccccccccccccccccccccccccccc
```

To visualize the output of the program listed above, save it in a file named dvrmo1.f, compile it by typing

```
f77 dvrmo1.f -o dvrmo1
```

and run it by typing

```
./dvrmo1
```

Then, cut the script attached below, save it with the name scr_ho1 in the same directory where you run your code, and visualize the 4 eigenstates by typing

```
gnuplot<scr_ho1
```

where the file named scr_ho1 has the following lines:

Download from (http://ursula.chem.yale.edu/∼batista/classes/v572/scr_ho1)

```
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
```

40.16 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. (10) 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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      double precision function rhosc(r)
```

```
      implicit real *8(a-h,o-z)
      rhosc=0.5*1.*(r-5.)**2
      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)
      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
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      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
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

To visualize the output of the program listed above, save it in a file named dvrho2.f, compile it by typing

```
f77 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.
```

40.17 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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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/3.0d0-0.5d0/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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
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)
      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
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
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
```

To visualize the output of the program listed above, save it in a file named dvrmo2.f, compile it by typing

```
f77 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.
```

40.18 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

$$V(x) = -0.5x^2 + 1.0/(16.0 * 1.3544)x^4, \tag{724}$$

for the initial state

$$\Phi_0(x) = \pi^{-1/4} e^{-0.5(x-x0)^2}, \tag{725}$$

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. (500) 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

```
f77 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
```

.

```
      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
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
```

```
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
```

```
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
```

```
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
```

```
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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)
```

```
               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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
```

```
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE EXCITEDA(V,x)
c
c     Potential Energy Surface: Harmonic Oscillator
c
      implicit none
      REAL V,x
      v=0.5*x*x
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutines from Numerical Recipes
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
```

```
            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
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      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
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      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
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
```

40.19 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 $V_0(g) = 0.00\ eV$; $V_0(e) = 0.94\ eV$; $D_e(g) = 18941\ cm^{-1}$; $D_e(e) = 4911\ cm^{-1}$; $\beta(g) = 1.517\ \mathring{A}^{-1}$; $\beta(e) = 1.535\ \mathring{A}^{-1}$; $R_{eq}(g) = 2.66\ \mathring{A}$ and $R_{eq}(e) = 3.105\ \mathring{A}$.

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

40.20 Computational Problem: WTP

1. Write a program to write the DVR Liouvillian, introduced by Eq. (**??**), for the Morse potential.
2. Propagate the Wigner transform of a state initialized as $|0\rangle$, according to Eq. (**??**), 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<scr

40.21 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. (419), with $H_{11} = -H_{22} = H_{12} = H_{21} = 1.0$, initialized in one of the 2 states according to Eq. (429) and evolving by Velocity Verlet according to Eq. (425). 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
```