Contents

1 Syllabus 5

2 The Fundamental Postulates of Quantum Mechanics 6

3 Continuous Representations 9

4 Vector Space 9
   4.1 Exercise 1 ................................................................. 12
   4.2 Exercise 2 ................................................................. 12

5 Digital Grid-Based Representations 12
   5.1 Computational Problem 1 ............................................... 13
   5.2 Computational Problem 2 ............................................... 14
   5.3 Computational Problem 3 ............................................... 14
   5.4 Computational Problem 4 ............................................... 15

6 Heisenberg Representation 15

7 Fourier Grid Hamiltonian 18
   7.1 Computational Problem FGH .......................................... 18

8 Variational Theorem 19

9 SOFT Method 19
   9.1 Computational Problem 5 ............................................... 20
   9.2 Computational Problem 6 ............................................... 21
   9.3 Computational Problem 7 ............................................... 21
   9.4 Computational Problem 8 ............................................... 22
   9.5 Computational Problem 9 ............................................... 22

10 SOFT Propagation on Multiple Surfaces 22
   10.1 Problem 10 ............................................................... 23
   10.2 Problem 11 ............................................................... 24
   10.3 Computational Problem 12 ........................................... 24

11 Path Integrals: Thermal Correlation Functions 24
   11.1 Exercise 13 .............................................................. 27

12 SOFT Computations of Thermal Correlation Functions 28
   12.1 Computational Problem 14 ........................................... 29

13 MP/SOFT Method 29
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Exam 1</td>
<td>32</td>
</tr>
<tr>
<td>14.1</td>
<td>Answer Key</td>
<td>33</td>
</tr>
<tr>
<td>15</td>
<td>Discrete Variable Representation</td>
<td>37</td>
</tr>
<tr>
<td>15.1</td>
<td>Computational Problem 15</td>
<td>40</td>
</tr>
<tr>
<td>16</td>
<td>Tunneling Dynamics</td>
<td>40</td>
</tr>
<tr>
<td>16.1</td>
<td>Coherent Control of Tunneling Dynamics</td>
<td>41</td>
</tr>
<tr>
<td>16.2</td>
<td>Computational Problem 16</td>
<td>42</td>
</tr>
<tr>
<td>17</td>
<td>Linear Photoabsorption Lineshape: A Time Dependent Picture</td>
<td>42</td>
</tr>
<tr>
<td>18</td>
<td>Time Dependent Perturbation Theory</td>
<td>45</td>
</tr>
<tr>
<td>19</td>
<td>Nonlinear (Pump-Probe) Spectroscopy</td>
<td>48</td>
</tr>
<tr>
<td>20</td>
<td>Pump-Probe Photoelectron Spectroscopy</td>
<td>49</td>
</tr>
<tr>
<td>21</td>
<td>Direct Photoelectron-Detachment Spectroscopy</td>
<td>52</td>
</tr>
<tr>
<td>21.1</td>
<td>Computational Problem 17</td>
<td>52</td>
</tr>
<tr>
<td>22</td>
<td>Exam 2</td>
<td>54</td>
</tr>
<tr>
<td>22.1</td>
<td>Answer Key</td>
<td>56</td>
</tr>
<tr>
<td>23</td>
<td>The Reaction Surface Hamiltonian Method</td>
<td>62</td>
</tr>
<tr>
<td>24</td>
<td>Wigner Transform Formulation of Quantum Dynamics</td>
<td>63</td>
</tr>
<tr>
<td>24.1</td>
<td>Linearized Wigner Transform:</td>
<td>66</td>
</tr>
<tr>
<td>24.2</td>
<td>Motion of Auxiliary Variables:</td>
<td>66</td>
</tr>
<tr>
<td>25</td>
<td>Bohmian Quantum Dynamics</td>
<td>67</td>
</tr>
<tr>
<td>25.1</td>
<td>Hamilton Jacobi Equation</td>
<td>68</td>
</tr>
<tr>
<td>25.2</td>
<td>Quantum Dynamics: Motion of Hidden Variables</td>
<td>68</td>
</tr>
<tr>
<td>25.3</td>
<td>Discussion of Bohmian Trajectories</td>
<td>70</td>
</tr>
<tr>
<td>25.4</td>
<td>EPR Paradox</td>
<td>70</td>
</tr>
<tr>
<td>26</td>
<td>Exam 3</td>
<td>71</td>
</tr>
<tr>
<td>26.1</td>
<td>Answer Key</td>
<td>71</td>
</tr>
<tr>
<td>27</td>
<td>Semiclassical Dynamics</td>
<td>76</td>
</tr>
<tr>
<td>28</td>
<td>Semiclassical Dynamics in the Gaussian-Hermite Basis</td>
<td>78</td>
</tr>
<tr>
<td>28.1</td>
<td>Multidimentional Semiclassical Dynamics</td>
<td>80</td>
</tr>
<tr>
<td>28.2</td>
<td>Log Derivative Propagation</td>
<td>81</td>
</tr>
</tbody>
</table>
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:


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”.


Useful search engines for mathematical and physical concepts can be found at http://scienceworld.wolfram.com/physics/ and http://mathworld.wolfram.com/

The lecture notes are posted online at: (http://ursula.chem.yale.edu/~batista/classes/v572/v572.pdf)

Grading: There will be no final exam for this class.

The final grading evaluation is the same for both undergraduate and graduate students: homework (50%), and three mid-terms (50%).

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.

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 \( t \).

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:

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

and

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

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

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

**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,

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

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

7
The eigenfunctions of a linear and hermitian operator form a complete basis set. Therefore, any function \( \psi(x) \) that is continuous, single valued, and square integrable can be expanded as a linear combination of eigenfunctions \( \phi_n(x) \) of a linear and hermitian operator \( \hat{A} \) as follows,

\[
\psi(x) = \sum_j C_j \phi_j(x),
\]

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

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

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

**Hilbert-Space**

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

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

\[
\begin{pmatrix}
\phi_2(x) \\
\psi(x) \\
\phi_1(x)
\end{pmatrix}
\]

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 \( \alpha \) 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 \( \phi_j \) 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.

4 Vector Space

Vector-Ket Space \( \varepsilon \): 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 \( \varepsilon \) is,
Note that the expansion coefficients $C_j$ depend only on the kets $|\psi_j\rangle$ 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 $\chi$ is a mathematical operation that transforms a function $\psi(x)$ into a number. This concept is extended to the vector-ket space $\varepsilon$, as an operation that transforms a vector-ket into a number as follows,

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

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 $\varepsilon^*$:** For every ket $|\psi\rangle$ we define a linear functional $<\psi|$, called bra-vector, as follows:

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

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

$$<\psi|(a|\phi\rangle + b|f\rangle) = a <\psi|(|\phi\rangle) + b <\psi|(|f\rangle).$$
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 $\varepsilon^*$ 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 $\varepsilon^*$ according to the following diagram:

![Diagram of Dual-Space $\varepsilon^*$](image)

where $C_j^*$ is the projection of $<\psi|$ along $<\phi_j|$. 

**Projection Operator and Closure Relation**

Given a ket $|\psi>$ in a certain basis set $|\phi_j>$,

$$|\psi> = \sum_j C_j |\phi_j>,$$ (1)

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

$$C_j = <\phi_j|\psi>.$$ (2)

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

$$|\psi> = \sum_j |\phi_j><\phi_j|\psi>.$$ (3)

From Eq.(3), it is obvious that

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

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

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

where

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

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

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

This is the so-called Continuity Equation.

**Note:** Remember that given a vector field $j$, e.g., $j(x,y,z) = j_1(x,y,z)\hat{i} + j_2(x,y,z)\hat{j} + j_3(x,y,z)\hat{k}$, 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:

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

### 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.
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,

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

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

\[ \Psi_0(x) = \int dx' c(x') \delta(x-x'), \]

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 \( x = (x_{\text{min}}, x_{\text{max}}) \), by discretizing Eq. (5) as follows,

\[ \Psi_0(x) = \Delta \sum_{j=1}^{n} c_j \delta(x-x_j), \]

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

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., \( x_j, p_j, \gamma_j \), 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 \( x_j = x_{\text{min}} + (j-1)\Delta \) with finite resolution \( \Delta = (x_{\text{max}} - x_{\text{min}})/(n-1) \) 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:

\[ \Psi_0(p) = \langle p | \Psi_0 \rangle. \]

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

\[ \langle p | \Psi_0 \rangle = \int dx \langle p | x \rangle \langle x | \Psi_0 \rangle = (2\pi)^{-1/2} \int dx e^{-ipx} \langle x | \Psi_0 \rangle. \]
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:
\[
\int dx \exp(-a_2x^2 + a_1x + a_0) = \sqrt{\pi/a_2} \exp(a_0 + a_1^2/(4a_2)).
\]

Next, we consider the grid-based representation of operators (e.g., \(\hat{x}\), \(\hat{p}\), \(V(\hat{x})\), and \(\hat{T} = \frac{\hat{p}^2}{2m}\)) 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 \(x_j = x_{\text{min}} + (j - 1)\Delta\), with finite resolution \(\Delta = (x_{\text{max}} - x_{\text{min}})/(n - 1)\)). 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,

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

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

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

However, for a more general operator (e.g., \( \hat{T} = \frac{\hat{p}^2}{2m} \)) 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),

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

\[
\langle x | p \rangle = (2\pi)^{-1/2} e^{ip\hat{x}}.
\] (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., \( \hat{T} = \frac{\hat{p}^2}{2m} \)) 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 \( T = \langle \Psi_0 | \frac{\hat{p}^2}{2m} | \Psi_0 \rangle \) by using the Fourier transform procedure, where \( \Psi_0 \) 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 \( \hat{H} = \frac{\hat{p}^2}{2m} + V(\hat{x}) \), 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. \( \hat{H} \psi_l = E_l \psi_l \),

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

\[
|\psi_l \rangle = \sum_j C_i^{(j)} |\phi_j \rangle, \tag{16}
\]

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

\[
\sum_j \hat{H} |\phi_j \rangle C_i^{(j)} = \sum_j E_i C_i^{(j)} |\phi_j \rangle.
\]

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

\[
\sum_j \langle \phi_k | \hat{H} |\phi_j \rangle C_i^{(j)} = \sum_j E_i \langle \phi_k | \phi_j \rangle C_i^{(j)}, \tag{18}
\]

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,

\[
\begin{align*}
(k = 1) & \rightarrow \quad H_{11} C_1^{(1)} + H_{12} C_1^{(2)} + H_{13} C_1^{(3)} + ... + H_{1n} C_1^{(n)} = E_1 C_1^{(1)} + 0 C_1^{(2)} + ... + 0 C_1^{(n)}, \\
(k = 2) & \rightarrow \quad H_{22} C_1^{(1)} + H_{22} C_1^{(2)} + H_{23} C_1^{(3)} + ... + H_{2n} C_1^{(n)} = 0 C_1^{(1)} + E_1 C_1^{(2)} + ... + 0 C_1^{(n)}, \\
(k = n) & \rightarrow \quad H_{nn} C_1^{(1)} + H_{n2} C_1^{(2)} + H_{n3} C_1^{(3)} + ... + H_{nn} C_1^{(n)} = 0 C_1^{(1)} + 0 C_1^{(2)} + ... + E_1 C_1^{(n)},
\end{align*}
\]

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

\[
\begin{bmatrix}
H_{11} & H_{12} & ... & H_{1n} \\
H_{21} & H_{22} & ... & H_{2n} \\
... & ... & ... & ...
\end{bmatrix}
\begin{bmatrix}
C_1^{(1)} \\
C_1^{(2)} \\
C_1^{(n)}
\end{bmatrix} =
\begin{bmatrix}
E_1 & 0 & 0 & 0 \\
0 & E_1 & 0 & 0 \\
0 & 0 & ... & E_1
\end{bmatrix}
\begin{bmatrix}
C_1^{(1)} \\
C_1^{(2)} \\
C_1^{(n)}
\end{bmatrix}. \tag{20}
\]

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 \( C_i^{(j)} = \langle \phi_j | \psi_l \rangle \), 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,

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

can be written in the matrix representation as follows,

\[ \langle \psi_l | H | \psi_l \rangle = C_l^\dagger H C_l = \begin{bmatrix} H_{11} & H_{12} & \cdots & H_{1n} \\ H_{21} & H_{22} & \cdots & H_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ H_{n1} & H_{n2} & \cdots & H_{nn} \end{bmatrix} \begin{bmatrix} C_l^{(1)} \\ C_l^{(2)} \\ \vdots \\ C_l^{(n)} \end{bmatrix}. \]

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 \(C_l^{(j)}\), 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 \(\det[H - \hat{1}E] = 0\), where \(\hat{1}\) is the unity matrix.

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

\[ \begin{bmatrix} H_{11} & H_{12} & \cdots & H_{1n} \\ H_{21} & H_{22} & \cdots & H_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ H_{n1} & H_{n2} & \cdots & H_{nn} \end{bmatrix} \begin{bmatrix} C_l^{(1)} \\ C_l^{(2)} \\ \vdots \\ C_l^{(n)} \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}. \]

(21)

7 Fourier Grid Hamiltonian

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

\[ H(j, j') = V(x_j) \langle x_j | x_{j'} \rangle + \frac{\Delta x \Delta p}{2\pi \hbar} \sum_{k=1}^{n_p} e^{i \frac{\hbar^2}{2m} (x_{j'} - x_j) p_k} \frac{p_k^2}{2m}, \]

\[ = V(x_j) \delta_{jj'} + \frac{\Delta x \Delta p}{2\pi \hbar} \sum_{k=1}^{n_p} e^{i \frac{\hbar^2}{2m} (x_{j'} - x_j) p_k} \frac{p_k^2}{2m}, \]
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

\[
x_j = (j - n_x/2)\Delta_x,
\]

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

\[
\langle x_l|\hat{T}|x_j \rangle = \langle x_l|\frac{p^2}{2m}|x_j \rangle,
\]

\[
= \int dp \int dp' \langle x_l|p' \rangle \frac{p'^2}{2m} \langle p'|p \rangle \langle p|x_j \rangle,
\]

\[
= \int dp \int dp' \langle x_l|p' \rangle \frac{p'^2}{2m} \langle p'|p \rangle \langle p|x_j \rangle
\]

\[
= \int dp \langle x_l|p \rangle \frac{p^2}{2m} \langle p|x_j \rangle = \frac{\Delta x}{2\pi\hbar} \int dpe^{i(x_l-x_j)p} \frac{p^2}{2m},
\]

\[
= \frac{\Delta x \Delta p}{2\pi\hbar} \sum_{k=1}^{n_p} e^{i(x_l-x_j)p_k} \frac{p_k^2}{2m},
\]

since the identity operator is \( I = \sum_j |x_j \rangle \langle x_j | \), 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,
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 $\hat{H}$.

$$< \psi | \hat{H} | \psi > = \sum_j \sum_k C_k^* C_j < \phi_k | \hat{H} | \phi_j >,$$

$$= \sum_j \sum_k C_k^* C_j E_j \delta_{kj},$$

$$= \sum_j C_j^* C_j E_j \geq E_0 \sum_j C_j^* C_j,$$

where, $\sum_j C_j^* C_j = 1$.

**Variational Approach**: Starting with an initial trial wave function $\psi$ 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. The link [http://ursula.chem.yale.edu/~batista/classes/vvv/VT570.tar](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](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:

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

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

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

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 \( \tau \) 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,

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

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 \( \tau \).

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

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

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

\[
\Psi_{t_{k+1}}(x) = e^{-iV(\hat{\tau})\tau/2} \frac{1}{\sqrt{2\pi}} \int dp \int dx' e^{-i\hat{p}^2\tau/(2m)} \frac{1}{\sqrt{2\pi}} \int \langle x'|p\rangle e^{-iV(\hat{\tau})\tau/2}\Psi_{t_k}(x'). \tag{30}
\]

According to Eq. (30), then, the computational task necessary to propagate \( \Psi_t(x) \) for a time-increment \( \tau \) 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 \( x_j = x_{\text{min}} + (j - 1)\Delta \), with finite resolution \( \Delta = (x_{\text{max}} - x_{\text{min}})/(n - 1) \).

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 \( \tilde{\bar{\Psi}}_{t_k}(p_j) \), and represent the kinetic energy part of the Trotter expansion \( e^{-i\hat{p}^2\tau/(2m)} \) as an array of numbers \( e^{-ip^2\tau/(2m)} \) associated with a grid of equally spaced momenta \( p_j = j/(x_{\text{max}} - x_{\text{min}}) \).

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

\[\tilde{\bar{\Psi}}_{t_k}(p_j) = e^{-ip^2\tau/(2m)}\tilde{\bar{\Psi}}_{t_k}(p_j).\]
5. Inverse Fourier transform $\tilde{\Psi}_{tk}(p_j)$ to obtain $\tilde{\Psi}_{tk}(x_j)$ 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 $\tilde{\Psi}_{tk}(x_j)$ by simple multiplication of array elements,

$$\Psi_{tk+1}(x_j) = e^{-iV(x_j)\tau/2}\tilde{\Psi}_{tk}(x_j).$$

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:

$$p_{j+1} = p_j + (F(x_j) + F(x_{j+1}))\tau/2,$$

$$x_{j+1} = x_j + p_j\tau/m + F(x_j)\tau^2/(2m).$$

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 $a = \sqrt{k/(2De)}$, 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 $\text{abs}(x) < 0.5$, and $V(x) = 0$, otherwise. Hint: In order to avoid artificial recurrences you might need to add an absorbing imaginary potential $V_a(x) = i(\text{abs}(x) - 10)^4$, if $\text{abs}(x) > 10$, and $V_a(x) = 0$, otherwise.
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,

\[ \hat{H} = \frac{p^2}{2m} + \hat{V}, \]

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

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

\[ |\Psi(x; t)\rangle = \varphi_1(x; t)|1\rangle + \varphi_2(x; t)|2\rangle, \]

given the initial conditions \( \varphi_1(x; 0) \) and \( \varphi_2(x; 0) \), where \( \varphi_1(x; t) \) and \( \varphi_2(x; t) \) 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, \( |\Psi(x; t)\rangle \) involves two wave-packet components and \( \hat{H} \) is a \( 2 \times 2 \) matrix in the basis of \(|1\rangle\) and \(|2\rangle\).

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

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

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 \( \varphi_1(x; t) \) and \( \varphi_2(x; t) \) for time \( \tau \), as follows,

\[ \left( \begin{array}{c} \varphi'_1(x; t + \tau) \\ \varphi'_2(x; t + \tau) \end{array} \right) = \left( \begin{array}{cc} e^{-i\frac{p^2}{2m}\tau} & 0 \\ 0 & e^{-i\frac{p^2}{2m}\tau} \end{array} \right) \left( \begin{array}{c} \varphi_1(x; t) \\ \varphi_2(x; t) \end{array} \right). \]

- Step [II]. Mix the two wave-packet components \( \varphi'_1(x; t + \tau) \) and \( \varphi'_2(x; t + \tau) \),

\[ \left( \begin{array}{c} \varphi''_1(x; t + \tau) \\ \varphi''_2(x; t + \tau) \end{array} \right) = M \left( \begin{array}{c} \varphi'_1(x; t + \tau) \\ \varphi'_2(x; t + \tau) \end{array} \right), \]

with

\[ M \equiv L^{-1} \left( \begin{array}{cc} e^{-iE_1(x)\tau} & 0 \\ 0 & e^{-iE_2(x)\tau} \end{array} \right) L, \]

where \( E_1(x) \) and \( E_2(x) \) are the eigenvalues of the potential energy matrix \( V = V_0 + V_c \) and \( L \) 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 $2 \times 2$ Hermitian matrix $V$, for which the matrix $M$ can be found analytically,

$$M \equiv \begin{pmatrix} e^{-i\hat{V}_1 \cdot \hat{x}^2 \tau} \cos(2V_c \hat{x} \tau) & -i \sin(2V_c \hat{x} \tau) e^{-i(\hat{V}_1 \cdot \hat{x} + \hat{V}_2 \cdot \hat{x}) \tau} \\ -i \sin(2V_c \hat{x} \tau) e^{-i(\hat{V}_1 \cdot \hat{x} + \hat{V}_2 \cdot \hat{x}) \tau} & \cos(2V_c \hat{x} \tau) e^{-i\hat{V}_2 \cdot \hat{x}^2 \tau} \end{pmatrix}. \quad (38)$$

- Step [III]. Propagate $\psi''_1(x; t+\tau)$ and $\psi''_2(x; t+\tau)$ for time $\tau$, according to the free-particle propagator, by applying the kinetic energy part of the Trotter expansion:

$$\begin{pmatrix} \psi_1(x; t+2\tau) \\ \psi_2(x; t+2\tau) \end{pmatrix} = \begin{pmatrix} e^{-i\hat{p}^2/m\tau} & 0 \\ 0 & e^{-i\hat{p}^2/m\tau} \end{pmatrix} \begin{pmatrix} \psi'_2(x; t+\tau) \\ \psi'_2(x; t+\tau) \end{pmatrix}. \quad (39)$$

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. (38) by considering that,

$$e^{-iV_c \cdot \hat{x}^2 \tau} = D^\dagger \begin{pmatrix} e^{iV_c \cdot \hat{x}^2 \tau} & 0 \\ 0 & e^{-iV_c \cdot \hat{x}^2 \tau} \end{pmatrix} D, \quad (40)$$

with

$$D = D^\dagger \equiv \begin{pmatrix} -1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}, \quad (41)$$

since

$$e^{-iV_c \cdot \hat{x}^2 \tau} = 1 + (-iV_c \cdot \hat{x}^2 \tau) + \frac{1}{2!}(-iV_c \cdot \hat{x}^2 \tau)^2 + ..., \quad (42)$$

and

$$V_c \equiv \begin{pmatrix} 0 & V_c(x) \\ V_c(x)^\dagger & 0 \end{pmatrix} = D^\dagger \begin{pmatrix} -V_c(x) & 0 \\ 0 & V_c(x) \end{pmatrix} D, \quad (43)$$

with $DD^\dagger = 1$.

### 10.2 Problem 11

Derive Eq. (37) by writing the matrix $V$ in the basis of adiabatic eigenstates

$$\phi_1(x) = L_{11}(x)|1\rangle + L_{21}(x)|2\rangle, \quad \phi_2(x) = L_{12}(x)|1\rangle + L_{22}(x)|2\rangle, \quad (44)$$

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

$$e^{-iV_c \cdot \hat{x}^2 \tau} = 1 + (-iV_c \cdot \hat{x}^2 \tau) + \frac{1}{2!}(-iV_c \cdot \hat{x}^2 \tau)^2 + ..., \quad (45)$$

23
show that in the adiabatic representation
\[ e^{-iVt} = \begin{pmatrix} e^{-iE_1(x)t} & 0 \\ 0 & e^{-iE_2(x)t} \end{pmatrix}. \] (46)

Finally, show that the diagonal matrix introduced by Eq. (46) can be rotated to the representation of diabatic states |1⟩, |2⟩ according to the similarity transformation
\[ L^{-1} \begin{pmatrix} e^{-iE_1(x)t} & 0 \\ 0 & e^{-iE_2(x)t} \end{pmatrix} L. \] (47)

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. (37). Propagate \[ |\Psi(x; t)\rangle = \varphi_1(x; t)|1⟩ + \varphi_2(x; t)|2⟩, \] where \( \varphi_1(x; 0) = \varphi_1(x; 0) = \Psi_0(x) \) 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
\[ V = \begin{pmatrix} V_1(x) & \delta \\ \delta & V_2(x) \end{pmatrix}, \] (48)

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 \( \Psi(x; t) \) according to the potential energy matrix introduced by Eq. (48), with \( \delta = 0 \) and compare your results with those obtained in item (a).

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:
\[ C(t) = \langle A(0)B(t) \rangle = \frac{1}{\tau} \int_0^\tau dt' A(q(t'), p(t')) B(q(t' + t), p(t' + t)), \] (49)

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,
\[ C(t) = Tr[\hat{\rho} \hat{A} \hat{B}(t)], \] (50)

\[1\] Note that calculations of \( C(t) \) provide a description of any equilibrium property, \( \langle A \rangle \), when \( \hat{B} = 1 \), or dynamical ensemble average \( \langle B(t) \rangle \), when \( \hat{A} = 1 \), respectively.
where $\hat{\rho} = Z^{-1} \exp(-\beta \hat{H})$ 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

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

(51)

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,

$$\hat{H} = \frac{\hat{p}^2}{2m} + \hat{V},$$

(52)

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,

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

(53)

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

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

(54)

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,

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

(55)

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

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

(56)

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

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

(57)

to second order accuracy.

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

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

(58)
where

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

(59)

since

\[ -i\hbar \frac{\partial}{\partial x} \langle x|p \rangle = p \langle x|p \rangle. \]  

(60)

Furthermore,

\[ \langle x|e^{-\epsilon V/2}|x' \rangle = e^{-\epsilon V(x)/2}\delta(x - x'). \]  

(61)

Therefore,

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

(62)

which gives,

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

(63)

or,

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

(64)

Matrix elements of the Boltzmann operator at finite-temperature can be obtained by substituting Eq. (64) into Eq. (56):

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

(65)

where \( \omega = 1/(\hbar\sqrt{\epsilon}) \). Note that the r.h.s of Eq. (65) corresponds to the partition function of a chain of \( n \) harmonic oscillators with coordinates \( x_j \) 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 multidimensional integral, introduced by Eq. (65), can be computed by importance sampling Monte Carlo by sampling sets of coordinates \( x_1, \ldots, x_{n-1} \) 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

\[ \hat{H} = \frac{p^2}{2m} + \frac{1}{2} m\omega^2 x^2, \]  

(66)

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:

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

(67)
which in the free particle limit ($\omega \to 0$) becomes
\[
\langle x|e^{-\beta \hat{H}}|x'\rangle = \sqrt{\frac{m}{2\pi\beta\hbar^2}} \exp\left(-\frac{m}{2\beta\hbar^2} \left[(x - x')^2\right]\right), \tag{68}
\]
since $\sinh(\beta\hbar\omega) \to \beta\hbar\omega$ and $\cosh(\beta\hbar\omega) \to 1$.

Matrix elements of the time-evolution operator $e^{-i\hbar\hat{H}t}$ 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. (64) and we obtain the short-time propagator as follows:
\[
\langle x|e^{-i\frac{\hbar}{\beta} \hat{H}\tau}|x'\rangle = \sqrt{\frac{m}{2\pi\hbar i\tau}} e^{i\frac{\hbar}{2\beta} \left[\frac{\epsilon}{\tau} (x - x')^2\right]} e^{i\frac{1}{2} \frac{\hbar}{\beta} \left(V(x) + V(x')\right)\tau}. \tag{69}
\]
Then, we concatenate the short-time propagators introduced by Eq. (69) and we obtain the finite-time propagator,
\[
\langle x_0|e^{-\frac{i\hbar}{\beta} \hat{H}t}|x_n\rangle = \int dx_1...\int dx_{n-1} \frac{m}{2\pi\hbar i\tau}^{n/2} e^{i\frac{\hbar}{2\beta} \left[\frac{\epsilon}{\tau} (x_{j+1} - x_j)^2\right]} e^{i\frac{1}{2} \frac{\hbar}{\beta} \left(V(x_j) + V(x_{j-1})\right)\tau}, \tag{70}
\]
which in the limit when $\tau \to 0$ and $n \to \infty$ with $t = n\tau$ becomes,
\[
\langle x_0|e^{-\frac{i\hbar}{\beta} \hat{H}t}|x_n\rangle = \int \mathcal{D}[x(t)] e^{i\frac{\hbar}{\beta} S_c(t)}, \tag{71}
\]
where $S_c(t)$ is the classical action associated with the arbitrary trajectory $x(t)$,
\[
S_c(t') \equiv \int_0^{t'} dt \left[\frac{1}{2} m \left(\frac{\partial}{\partial t} x(t)\right)^2 - V(x(t))\right], \tag{72}
\]
and $\mathcal{D}[x(t)]$ is defined as follows,
\[
\int \mathcal{D}[x(t)] f(x(t)) \equiv \int dx_1...\int dx_{n-1} \left(m/2\pi\hbar i\tau\right)^{n/2} f(x(t)), \tag{73}
\]
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.

### 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. (53):

$$C(t) = Z^{-1} \int dx \int dx' \int dx'' \langle x | e^{-\frac{it}{2}H_0} | x'' \rangle A(x') \langle x' | e^{iH_1 t} \hat{B} e^{-iH_1 t} | x'' \rangle \langle x'' | e^{-\frac{it}{2}H_0} | x \rangle. \quad (74)$$

The computational task necessary to obtain $C(t)$, according to Eq. (74), requires obtaining the matrix elements $A(x') \langle x' | e^{-\frac{it}{2}H_0} | x'' \rangle$ and $\langle x'' | e^{-\frac{it}{2}H_0} | x \rangle$ 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 $\beta \rightarrow -it$ from imaginary to real time.

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

$$\left\{ \frac{\partial}{\partial \beta} - \frac{1}{2m} \nabla_x^2 + V_0(x) \right\} \rho(x, x'; \beta) = 0, \quad (75)$$

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

$$\rho(x, x'; \epsilon) = \left( \frac{m}{2\pi \epsilon} \right)^{1/2} e^{-\frac{1}{2m} [V_0(x) + V_0(x')] - \frac{m}{2\epsilon} (x-x')^2}, \quad (76)$$

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

Equation (75) is formally integrated as follows,

$$\rho(x, x'; \beta) = \int dx'' \rho(x, x''; \beta - \epsilon) \rho(x'', x'; \epsilon), \quad (77)$$

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

$$\hat{1} = \int dx_j |x_j \rangle \langle x_j|, \quad (78)$$

yielding,

$$\langle x | e^{-(\beta-\epsilon)\hat{H}_0} | x'' \rangle = \int dx_{s-1} \cdots \int dx_2 \langle x | e^{-i\hat{H}_0 \tau} | x_{s-1} \rangle \cdots \langle x_1 | e^{-i\hat{H}_0 \tau} | x'' \rangle, \quad (79)$$

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

Each finite-time propagator, introduced by Eq. (79), is approximated for sufficiently small imaginary-time slices $\tau$ by the Trotter expansion to second-order accuracy,

$$e^{-i \hat{H}_0 \tau} \approx e^{-i V_0(x) \tau/2} e^{-i \frac{\hbar^2}{m} \tau} e^{-i V_0(x) \tau/2}. \quad (80)$$
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. (67) at $T_i$ and $T_f$, respectively.

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

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

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

13 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 $\hat{\rho}(x_j, x'_k; \epsilon)$ is substituted by coherent-state expansions generated according to the Matching-Pursuit algorithm.

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

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

\[
\hat{\rho}(x, x'; \epsilon) \approx \sum_{j=1}^{\infty} c_j \phi_j(x) [\phi'_j(x')]^*, \tag{81}
\]

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

\[
\phi_j(x) \equiv \prod_{k=1}^{N} A_{\phi_j}(k) e^{-\gamma_{\phi_j}(k)(x(k)-x_{\phi_j}(k))^2/2} e^{i \mu_{\phi_j}(k)(x(k)-x_{\phi_j}(k))}, \tag{82}
\]
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
\[
A_{\phi_j}(k) \equiv \frac{\langle a_{\phi_j}(k) \rangle}{\pi} \frac{1}{2a_{\phi_j}(k)} \frac{1}{2b_{\phi_j}(k) + f_{\phi_j}(k)} \frac{1}{2a_{\phi_j}(k)}.
\]
The expansion coefficients, introduced by Eq. (81), are defined as follows:
\[
c_j \equiv \begin{cases} I_j, & \text{when } j = 1, \\ I_j - \sum_{k=1}^{j-1} c_k \langle \phi_j | \phi_k \rangle \langle \phi_k' | \phi_j' \rangle, & \text{for } j = 2 - n, \end{cases}
\]
where the overlap integral \( I_j \) is defined as follows,
\[
I_j \equiv \int dx' dx \, \phi_j(x) \tilde{p}(x, x'; \epsilon)[\phi_j'(x')]^*.
\]

- 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:
\[
\rho(x, x'; \epsilon + i\tau) = \sum_{j=1}^{n} c_j e^{-i\omega(x)\tau/2} \phi_j(x) [\phi_j'(x')]^*.
\]

Note that the MP/SOFT approach reduces the computational task necessary for the imaginary or real time propagation of the Boltzmann operator matrix elements \( \rho(x, x'; \beta) \) to the problem of recursively generating the coherent-state expansions introduced by Eq. (81).

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 \( \phi_j(x) \) and \( \phi_j'(x) \), to locally maximize the overlap integral \( I_j \), introduced in Eq. (84). 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 \( \phi_1 \) and \( \phi_1' \) in the expansion introduced by Eq. (81) and the first expansion coefficient \( c_1 \), as follows: \( \rho_1(x, x'; \epsilon) = c_1 \phi_1(x) [\phi_1'(x')]^* + \epsilon_1(x, x') \), where \( c_1 \equiv I_1 \), as defined according to Eq. (84). Note that due to the definition of \( c_1 \), the residue \( \epsilon_1(x, x') \) does not overlap with the product state \( \phi_1(x) [\phi_1'(x')]^* \). Therefore, the norm of the remaining residue \( \epsilon_1(x, x') \) is smaller than the norm of the initial target state \( \tilde{p}(x, x'; \epsilon) \) —i.e., \( \| \epsilon_1 \| < \| \tilde{p} \| \).
• Step [1.2]. Goto [1.1], replacing \( \tilde{\rho}(x, x'; \epsilon) \) by \( \varepsilon_1(x, x') \) — i.e., sub-decompose the residue by its projection along the direction of its locally optimum match as follows: 
\[
\varepsilon_1(x, x') = c_2 \phi_2(x) [\phi_2'(x')]^* + \varepsilon_2(x, x'),
\]
where 
\[
c_2 \equiv \int dx'dx \, \phi_2(x) \varepsilon_1(x, x') [\phi_2'(x')]^*.
\] (87)

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

Step [1.2] is repeated each time on the resulting residue. After \( n \) successive projections, the norm of the residue \( \varepsilon_n \) is smaller than a desired precision \( \epsilon \) — i.e., 
\[
\| \varepsilon_n \| = (1 - \sum_{j=1}^{n} |c_j|^2)^{1/2} < \epsilon,
\]
and the resulting expansion is given by Eq. (81). Note that norm conservation of \( \tilde{\rho} \) 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 
\[
\langle \phi_j | e^{-iV_j(\hat{x})\tau/2} | \tilde{\phi}_k \rangle \text{ and } \langle \phi_j | e^{-iV_j(\hat{x})\tau/2} | \phi_k \rangle,
\]
where \( |\phi_k\rangle \) and \( |\tilde{\phi}_k\rangle \) are localized Gaussians introduced by Eqs. (82) and (86), 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 \( V_j(\hat{x}) \) 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.
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 \( e^{\beta \hat{H}} \) 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 \( \Phi_k \) and \( \Phi_n \) 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 \( E_t = \langle \Psi_t | \hat{H} | \Psi_t \rangle \).
14.1 Answer Key

Exercise 1:
(10 points) Exercise 1.1: See lecture notes (pages 3–5).

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 $t$.

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.

Postulate 3: The only possible experimental results of a measurement of an observable are the eigenvalues of the operator that corresponds to such observable.

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,

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

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: 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 $\int dx\phi_n^*\phi_m = 0$. 

33
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
\[
\exp(-\beta \hat{H}) = 1 - \beta \hat{H} + \frac{1}{2} \beta^2 \hat{H}^2 + \ldots
\]
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:
\[
-i \hbar \int_{-\infty}^{\infty} dx \phi_j^* \frac{\partial \phi_k(x)}{\partial x} = -i \hbar \left( \phi_j^*(x) \phi_k(x) \bigg|_{-\infty}^{\infty} - \int_{-\infty}^{\infty} dx \phi_k(x) \frac{\partial}{\partial x} \phi_j^*(x) \right),
\]
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 \( \Phi_j \) and \( \Phi_n \) 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,
\[
\int dx \left( \frac{\partial \Phi_n^*}{\partial t} \hat{H} \Phi_n + \Phi_j \frac{\partial \hat{H}}{\partial t} \Phi_n + \Phi_j^* \hat{H} \frac{\partial \Phi_n}{\partial t} \right) = 0.
\]
Considering that \( \hat{H} \) is hermitian, we obtain:
\[
E_n \langle \Phi_j | \Phi_n \rangle + \langle \Phi_j | \frac{\partial \hat{H}}{\partial t} | \Phi_n \rangle + E_j \langle \Phi_j | \hat{H} \Phi_n \rangle = 0.
\]
Finally, note that \( \langle \Phi_j | \Phi_n \rangle = -\langle \Phi_j | \Phi_n \rangle \) since \( \frac{\partial}{\partial t} \langle \Phi_k | \Phi_n \rangle = 0 \). Thus,
\[
(E_n - E_j) \langle \hat{\Phi}_j | \Phi_n \rangle + \langle \Phi_j | \frac{\partial \hat{H}}{\partial t} | \Phi_n \rangle = 0.
\]

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 \( \tau \) involves the following steps:
1. Represent \( \Psi_{tk}(x') \) and \( e^{-i(V(x') + \epsilon(tk))\tau/2} \) as arrays of numbers \( \Psi_{tk}(x_j) \) and \( e^{-i(V(x_j) + \epsilon(tk))\tau/2} \) associated with a grid of equally spaced coordinates \( x_j = x_{\text{min}} + (j - 1)\Delta \), with finite resolution \( \Delta = (x_{\text{max}} - x_{\text{min}})/(n - 1) \).

2. Apply the potential energy part of the Trotter expansion \( e^{-i(V(x') + \epsilon(tk))\tau/2} \) to \( \Psi_{tk}(x') \) by simple multiplication of array elements:

\[
\tilde{\Psi}_{tk}(x_j) = e^{-i(V(x_j) + \epsilon(tk))\tau/2} \Psi_{tk}(x_j).
\]

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

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

\[
\tilde{\Psi}_{tk}(p_j) = e^{-ip^2\tau/(2m)} \tilde{\Psi}_{tk}(p).
\]

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

6. Apply the potential energy part of the Trotter expansion \( e^{-i(V(x') + \epsilon(tk))\tau/2} \) to \( \tilde{\Psi}_{tk}(x') \) by simple multiplication of array elements,

\[
\Psi_{tk+1}(x_j) = e^{-iV(x_j)\tau/2} \tilde{\Psi}_{tk}(x_j).
\]

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 \( \Psi_t \) 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:

\[
\tilde{\Psi}_t(x) = \int_{-\infty}^{\infty} dp dx' \langle x'|p \rangle \frac{\hat{p}^2}{2m} \langle p|x' \rangle \langle x'|\Psi_t \rangle + V(x)\Psi_t(x),
\]

\[
= \int_{-\infty}^{\infty} dp dx' \langle x'|p \rangle \frac{\hat{p}^2}{2m} \langle p|x' \rangle \langle x'|\Psi_t \rangle + V(x)\Psi_t(x),
\]
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 repeatedly applying the following transformation:

\[
\begin{align*}
p_{j+1} &= p_j + \left( F(x_j) + F(x_{j+1}) \right) \tau / 2 \\
x_{j+1} &= x_j + p_j \tau / m + F(x_j) \tau^2 / (2m).
\end{align*}
\]

(88)

After applying the transformation \( N \) times, with \( N = t/\tau \), we evaluate the energy of the system as follows: \( E_t = \frac{p_N^2}{2m} + V(x_N) \).
15 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,

\[ HC_j + C_jE_j = 0. \] (89)

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:

\[ x_j = x_{\min} + j \Delta, \quad \text{with} \quad \Delta = (x_{\max} - x_{\min})/N, \] (90)

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:

\[ H(j, j') = \delta_{jj'} \left( V(x_j) + \frac{\hbar^2}{2m \Delta^2} (1-j-j') \left( \frac{\pi^2}{3} + (1-\delta_{jj'}) \frac{2}{(j-j')^2} \right) \right), \] (91)

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 \( j = 1, 2, \ldots \). Furthermore, we show that for the particular case of a radial coordinate, defined in the interval \( x = (0, \infty) \), the Hamiltonian matrix elements are:

\[ H(j, j') = V(x_j) \delta_{jj'} + \frac{\hbar^2}{2m \Delta^2} (-1)^{j-j'} \left( \frac{\pi^2}{3} - \frac{1}{2j^2} \right) + (1-\delta_{jj'}) \left( \frac{2}{(j-j')^2} - \frac{2}{(j+j')^2} \right) \] (92)

To derive Eq. (91) and Eq. (92), we consider the Hamiltonian,

\[ \hat{H} = \hat{T} + V(\hat{x}), \] (93)

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

\[ V^{(\delta)}(j, k) = \left\langle j | V(\hat{x}) | k \right\rangle = \int dx \delta^*(x - x_j) V(\hat{x}) \delta(x - x_k), \] (94)

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.
and then rotating $T^{(\phi)}$ to the representation of delta functions by using the following similarity transformation:

$$T^{(\phi)} = \Gamma^{-1}T^{(\phi)}\Gamma,$$

where $\Gamma$ is the transformation matrix defined by the linear combinations,

$$\phi_k(x) = \sum_j \Gamma(j, k) \delta(x - x_j) \Delta',$$

where

$$\Gamma(j, k) = \phi_k(x_j).$$

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:

$$\phi_k(x) = \sqrt{\frac{2}{x_{\text{max}} - x_{\text{min}}}} \sin\left(k \frac{\pi (x - x_{\text{min}})}{x_{\text{max}} - x_{\text{min}}}\right),$$

with $\phi_k(x_{\text{min}}) = 0$ and $\phi_k(x_{\text{max}}) = 0$. Therefore,

$$\hat{T}\phi_k(x) = \frac{(\hbar \pi k)^2}{2m} \phi_k(x),$$

and $T^{(\phi)}$ is diagonal with matrix elements,

$$\hat{T}^{(\phi)}(j, k) = \langle \phi_j | \hat{T} | \phi_k \rangle = \frac{(\hbar k)^2}{2m} \frac{\pi^2}{(x_{\text{max}} - x_{\text{min}})^2} \delta_{jk}.$$ (100)

Therefore, substituting Eq. (100) and Eq. (97) into Eq. (95) we obtain,

$$T^{(\phi)}(i, i') = \sum_{j, k=1}^{N-1} \Gamma^{-1}(i, j)T^{(\phi)}(j, k)\Gamma(k, i') = \sum_{j, k=1}^{N-1} \Gamma(j, i)T^{(\phi)}(j, k)\Gamma(k, i'),$$

$$= \frac{\Delta \pi^2}{(x_{\text{max}} - x_{\text{min}})^2} \sum_{j, k=1}^{N-1} \phi_j(x_i) \frac{(\hbar k)^2}{2m} \delta_{jk} \phi_k(x_{i'}') = \frac{\Delta \pi^2}{(x_{\text{max}} - x_{\text{min}})^2} \sum_{k=1}^{N-1} \phi_k(x_i) \frac{(\hbar k)^2}{2m} \phi_k(x_{i'}'),$$

$$= \frac{\Delta \hbar^2 \pi^2}{2m(x_{\text{max}} - x_{\text{min}})^2} \sum_{k=1}^{N-1} k^2 \sin\left(k \pi \frac{(x_i - x_{\text{min}})}{x_{\text{max}} - x_{\text{min}}}\right) \sin\left(k \pi \frac{(x_{i'}' - x_{\text{min}})}{x_{\text{max}} - x_{\text{min}}}\right).$$ (101)

Finally, substituting Eq. (90) into Eq. (101) we obtain:

$$T^{(\delta)}(j, j') = \frac{\hbar^2 \pi^2}{2m(x_{\text{max}} - x_{\text{min}})^2} \sum_{k=1}^{N-1} k^2 \sin\left(k \pi \frac{j}{N}\right) \sin\left(k \pi \frac{j'}{N}\right).$$ (102)
To calculate the finite series introduced by Eq. (102) we first note that,

\[
2 \sin \left( \frac{k \pi j}{N} \right) \sin \left( \frac{k \pi j'}{N} \right) = \cos \left( \frac{k \pi (j - j')}{N} \right) - \cos \left( \frac{k \pi (j + j')}{N} \right),
\]

so that Eq. (102) can be written as follows:

\[
T^{(\delta)}(j, j') = \frac{\hbar^2 \pi^2}{2m(x_{\text{max}} - x_{\text{min}})^2} \frac{2}{N} \left[ \text{Re} \sum_{k=1}^{N-1} k^2 \exp \left( i \frac{k \pi (j - j')}{N} \right) - \text{Re} \sum_{k=1}^{N-1} k^2 \exp \left( i \frac{k \pi (j + j')}{N} \right) \right].
\]

(104)

Then, we consider the geometric series \( S_N = \sum_{k=0}^{N-1} x^k \) and we note that \( S_N - xS_N = 1 - x^N \), therefore \( S_N = (1 - x^N)/(1 - x) \). Also, we note that

\[
x \frac{\partial}{\partial x} \sum_{k=0}^{N-1} x^k = \sum_{k=0}^{N-1} k x^k,
\]

\[
x^2 \frac{\partial^2}{\partial x^2} \sum_{k=0}^{N-1} x^k = \sum_{k=0}^{N-1} k^2 x^k - \sum_{k=0}^{N-1} k x^k,
\]

Therefore,

\[
\sum_{k=1}^{N-1} k^2 x^k = x^2 \frac{\partial^2}{\partial x^2} \left( \frac{1 - x^N}{1 - x} \right) + x \frac{\partial}{\partial x} \left( \frac{1 - x^N}{1 - x} \right).
\]

(106)

We evaluate the sums over \( k \) in Eq. (104) analytically to obtain:

\[
T^{(\delta)}(j, j') = \frac{\hbar^2 (-1)^{j-j'}}{2m(x_{\text{max}} - x_{\text{min}})^2} \frac{\pi^2}{2} \left[ \frac{1}{\sin^2[\pi (j - j')/(2N)]} - \frac{1}{\sin^2[\pi (j + j')/(2N)]} \right],
\]

(107)

for \( j \neq j' \) and

\[
T^{(\delta)}(j, j) = \frac{\hbar^2}{2m(x_{\text{max}} - x_{\text{min}})^2} \frac{\pi^2}{2} \left[ \frac{(2N^2 + 1)}{3} - \frac{1}{\sin^2[\pi j/N]} \right].
\]

(108)

Equation (91) is obtained from Eq. (107) and Eq. (108), by taking the limit \( x_{\text{min}} \to -\infty, \) \( x_{\text{max}} \to \infty, \) at finite \( \Delta \). This requires \( N \to \infty \). Furthermore, since \( \Delta(j + j') = x_j + x_{j'} - 2x_{\text{min}} \) and \( \Delta(j - j') = x_j - x_{j'} \), this limit implies \( (j + j') \to \infty \) while \( (j - j') \) remains finite.

Equation (92) is obtained from Eq. (107) and Eq. (108), by making \( x_{\text{min}} = 0 \), and taking the limit \( x_{\text{max}} \to \infty, \) at finite \( \Delta \). 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.
15.1 Computational Problem 15

(Due November 16, 2008)

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 \( E(\nu) = \left(1/2 + \nu\right)\hbar \omega \), \( \nu = 0–10 \). 

15.2 Change the potential of the code written in 15.1 to that of a Morse oscillator \( V(\hat{x}) = D_e(1 - \exp(-a(\hat{x} - x_e)))^2 \), with \( x_e = 0 \), \( D_e = 8 \), 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 \( V(x, y) = \frac{1}{2}m\omega^2(x^2 + y^2) \) 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 \( V(\hat{x}, \hat{y}) = D_e(1 - \exp(-a(\hat{x} - x_e)))^2 + D_e(1 - \exp(-a(\hat{y} - x_e)))^2 \), with \( x_e = 0 \), \( D_e = 8 \), and \( a = \sqrt{k/(2D_e)} \), where \( k = m\omega^2 \), and recompute the eigenvalues and eigenfunctions.

16 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,

\[
H_0(x, p) = \frac{p^2}{2} - \alpha(x^2 - \beta x^4),
\]

with \( \alpha = 1/2^2 \) and \( \beta = 1/2^5 \). In the absence of an external perturbation, the initial non-stationary state

\[
\Phi_0(x) = \pi^{-1/4}e^{-(x-x_0)^2/2},
\]

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. (110)),

\[
\Phi_0(x) = \frac{1}{\sqrt{2}}(\chi_0(x) + \chi_1(x)),
\]

with \( \alpha = 1/2^2 \) and \( \beta = 1/2^5 \). In the absence of an external perturbation, the initial non-stationary state
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 \( \hat{H}|\chi_j\rangle = E_j|\chi_j\rangle \),

\[
|\Phi_t\rangle = \frac{1}{\sqrt{2}} \left( \chi_0(x)e^{-iE_0t} + \chi_1(x)e^{-iE_1t} \right),
\]

and

\[
|\xi(t)|^2 = |\langle \Phi_0|\Phi_t \rangle|^2 = \frac{1}{2} + \frac{1}{2} \cos(\Omega t),
\]

with the tunneling frequency \( \Omega = (E_1 - E_0)/\hbar \). Note that \( \Omega \) 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. (113) and (113), is that tunneling is the result of interference between the two components of the coherent superposition defined by Eq. (113) since changing the relative phases of the two terms would affect the underlying tunneling dynamics. For example, introducing a phase of \( \theta \) in the first term of Eq. (112) we obtain,

\[
|\Phi_t\rangle = \frac{1}{\sqrt{2}} \left( \chi_0(x)e^{-iE_0t}e^{i\theta} + \chi_1(x)e^{-iE_1t} \right),
\]

and

\[
|\xi(t)| = |\langle \Phi_0|\Phi_t \rangle|^2 = \frac{1}{2} + \frac{1}{2} \cos(\Omega t + \theta).
\]

This equation indicates that the probability of having the system overlapping with the initial state on the left of the barrier at time \( t \) is a function of \( \theta \). Therefore, manipulating \( \theta \) with an external field could be an effective method for coherently controlling the underlying tunneling dynamics.

### 16.1 Coherent Control of Tunneling Dynamics

As an example of coherent control of tunneling dynamics we consider the perturbational influence of instantaneous 2-\( \theta \) pulses described by the following operator:

\[
\hat{U}^{2\theta} = \cos \left( \frac{\Gamma \tau}{2} \right) \left( |\Phi_0\rangle\langle \Phi_0 | + |\Phi_a\rangle\langle \Phi_a | \right) - isin \left( \frac{\Gamma \tau}{2} \right) \left( |\Phi_0\rangle\langle \Phi_a | + |\Phi_a\rangle\langle \Phi_0 | \right),
\]

with \( \tau = 2\theta/\Gamma \). In particular, when \( \theta = \pi \),

\[
\hat{U}^{2\pi} = - (|\Phi_0\rangle\langle \Phi_0 | + |\Phi_a\rangle\langle \Phi_a |),
\]

a pulse that induces a \( \pi \) 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-\( \pi \) pulses.

Applying the 2 \( \pi \) pulse, described by Eq. (117), to a coherent state \( |\Psi_{t_0}\rangle = c_0(t_0)|\Phi_0\rangle + c_1(t_0)|\Phi_1\rangle + ... \), that has negligible population in the auxiliary state \( |\Phi_a\rangle \) (i.e., \( c_a(t_0) = 0 \), we obtain

\[
\hat{U}^{2\pi} |\Psi_t\rangle = -c_0(t_0)|\Phi_0\rangle + c_1(t_0)|\Phi_1\rangle + ...
\]
The pulse can also be represented as

\[ \hat{U}^{2\pi} = 1 - 2|\Phi_0 \rangle \langle \Phi_0|, \]  

(119)

since

\[ \hat{U}^{2\pi} |\Psi_{t_0} \rangle = (1 - 2|\Phi_0 \rangle \langle \Phi_0|) |\Psi_t \rangle = -c_0(t_0) |\Phi_0 \rangle + c_1(t_0) |\Phi_1 \rangle + ... \]  

(120)

The propagation of the system under the influence of \( N \) instantaneous 2-\( \pi \) pulses, applied at 2\( \tau \) intervals, generates the time-evolved state,

\[ |\Psi_{t+2N\tau} \rangle = c_0 \left( e^{-\frac{i}{\hbar} \hat{H} \tau} \hat{U}^{2\pi} e^{-\frac{i}{\hbar} \hat{H} \tau} \right)^N |\Phi_0 \rangle + e^{-\frac{i}{\hbar} \hat{H} 2N\tau} (c_1 |\Phi_1 \rangle + ...) \]  

(121)

The second equality in Eq. (121) is obtained by substituting \( \hat{U}^{2\pi} \) as defined by Eq. (119) and \( \Phi_0 \) according to Eq. (111).

Equation (121) shows that the square of the expansion coefficient associated with state \( \Phi_0 \) remains constant, for as long as the train of 2-\( \pi \) 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.

16.2 Computational Problem 16

(Due November 25, 2008) Modify the program for wave-packet propagation developed in Problem 12 and simulate the propagation of a wave packet in the symmetric double well

\[ V(x) = -0.5x^2 + 1.0/(16.0 * 1.3544)x^4, \]  

(122)

using the initial state

\[ \Phi_0(x) = \pi^{-1/4} e^{-0.5(x-x_0)^2}, \]  

(123)

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-\( \pi \) pulses, as described by Eq. (119), applied in the time-window \( t = 305-500 \) a.u.

17 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:

\[ I_0(\omega) = \frac{3}{2\pi\hbar} \int_{-\infty}^{\infty} dt e^{i(\hbar\omega+E_0)t} \xi(t), \]  

(124)
where \( |\Psi_0\rangle = \lambda \vec{\varepsilon}_0 \cdot \hat{\mu} |\Phi_0\rangle \), 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. (124), 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 overlapping 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. 15, 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:

\[
I_\beta(\omega) = \sum_j \rho_j(\beta) I_j(\omega),
\]

(125)

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 probability to state \( |\Phi_k\rangle \) as follows:

\[
P_0^{(k)}(\omega) = \lim_{t \to \infty} |c_k^{(1)}(t)|^2,
\]

(126)

where \( c_k^{(1)}(t) \) is defined by the Golden Rule expression of first order time-dependent perturbation theory,

\[
c_k^{(1)}(t) = -\frac{i}{\hbar} \int_{-\infty}^{t} dt' \langle \Phi_k | e^{\frac{i}{\hbar} \hat{H} t'} \hat{H}_1(t') e^{-\frac{i}{\hbar} \hat{H} t'} |\Phi_0\rangle.
\]

(127)

The derivation of Eq. (127), presented in the following section, assumes that the photoabsorption results from the interaction of the system with the monochromatic radiation field,

\[
\varepsilon(t) = \lambda \vec{\varepsilon}_0 (e^{i\omega t} + e^{-i\omega t}),
\]

(128)

where \( \lambda \ll 1 \) is a small dimensionless parameter that defines the dipolar interaction,

\[
\hat{H}_1(t) = -\lambda \vec{\varepsilon}_0 \cdot \hat{\mu} (e^{i\omega t} + e^{-i\omega t}),
\]

(129)

in the weak field limit.

Substituting the expression of the dipolar interaction, introduced by Eq. (129), into Eq. (127), we obtain:

\[
c_k^{(1)}(t) = -\frac{i}{\hbar} \int_{-\infty}^{t} dt' \langle \Phi_k | \lambda \vec{\varepsilon}_0 \cdot \hat{\mu} \left[ e^{\frac{i}{\hbar}(\hat{H} - E_k - \omega) t'} + e^{\frac{i}{\hbar}(\hat{H} - E_k + \omega) t'} \right] \rangle |\Phi_0\rangle,
\]

(130)

and substituting Eq. (130) into Eq. (126) we obtain:

\[
P_0^{(k)}(\omega) = |\langle \Phi_k | \lambda \vec{\varepsilon}_0 \cdot \hat{\mu} |\Phi_0\rangle|^2 \left[ \delta(E_0 + \hbar \omega - E_k) + \delta(E_0 - \hbar \omega - E_k) \right].
\]

(131)
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:

$$\alpha_0(\omega) = \sum_k (E_k - E_0)|\langle \Phi_k | \lambda \varepsilon_0 | \Phi_0 \rangle|^2 \left[ \delta(E_0 + \hbar \omega - E_k) + \delta(E_0 - \hbar \omega - E_k) \right].$$  \hspace{1cm} (132)

The absorption spectrum $\alpha_\beta(\omega)$, at finite temperature $T = 1/(\beta k_B)$, can be obtained from Eq. (132) as follows:

$$\alpha(\omega) = \sum_j \rho_j \sum_k (E_k - E_j) \delta(E_j + \hbar \omega - E_k) |\langle \Phi_k | \lambda \varepsilon_0 | \Phi_j \rangle|^2$$

$$+ \sum_j \rho_j \sum_k (E_k - E_j) \delta(E_j - \hbar \omega - E_k) |\langle \Phi_k | \lambda \varepsilon_0 | \Phi_j \rangle|^2,$$  \hspace{1cm} (133)

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. (133) and noting that $\rho_k = \rho_j e^{-\beta(E_k - E_j)}$ we obtain:

$$\alpha(\omega) = \sum_j \sum_k \rho_j (E_k - E_j) \delta(E_j + \hbar \omega - E_k) |\langle \Phi_k | \lambda \varepsilon_0 | \Phi_j \rangle|^2$$

$$- \rho_j e^{-\beta(E_k - E_j)} (E_k - E_j) \delta(E_j + \hbar \omega - E_k) |\langle \Phi_j | \lambda \varepsilon_0 | \Phi_k \rangle|^2,$$  \hspace{1cm} (134)

which gives the absorption lineshape

$$I(\omega) = \frac{3\alpha(\omega)}{\hbar \omega (1 - e^{-\beta \omega})} = 3 \sum_j \sum_k \rho_j \delta(E_j + \hbar \omega - E_k) |\langle \Phi_k | \lambda \varepsilon_0 | \Phi_j \rangle|^2.$$  \hspace{1cm} (135)

At 0 K, the absorption lineshape is obtained from Eq. (135) as follows:

$$I_0(\omega) = 3 \sum_k \delta(E_0 + \hbar \omega - E_k) |\langle \Phi_k | \lambda \varepsilon_0 | \Phi_0 \rangle|^2,$$  \hspace{1cm} (136)

that is equivalent to Eq. (124), since according to Eq. (124),

$$I_0(\omega) = \frac{3}{2\pi \hbar} \int_{-\infty}^{\infty} dt \langle \Phi_0 | (\lambda \varepsilon_0 | \mu) e^{\frac{i}{\hbar} (\hbar \omega + E_0 - \hat{H}) t} (\lambda \varepsilon_0 | \mu) |\Phi_0 \rangle.$$  \hspace{1cm} (137)

Note that introducing the closure relation, $\hat{1} = \sum |\Phi_j\rangle \langle \Phi_j|$ into Eq. (137), we obtain:

$$I_0(\omega) = 3 \sum_k \langle \Phi_0 | \lambda \varepsilon_0 | \mu \rangle \langle \Phi_k | \delta(\hbar \omega + E_0 - \hat{H}) \lambda \varepsilon_0 | \mu \rangle \langle \Phi_k | \lambda \varepsilon_0 | \mu \rangle \langle \Phi_0 \rangle,$$

$$= 3 \sum_k \delta(\hbar \omega + E_0 - E_k) \langle \Phi_0 | \lambda \varepsilon_0 | \mu \rangle \langle \Phi_k | \lambda \varepsilon_0 | \mu \rangle \langle \Phi_0 \rangle,$$  \hspace{1cm} (138)
that is identical to Eq. (136).

Finally, we note that Eq. (138) gives the linear photoabsorption lineshape in terms of the dipole-dipole correlation function as follows:

\[
I_0(\omega) = \frac{3}{2\pi\hbar} \sum_k \int_{-\infty}^{\infty} dt e^{i\omega t} \langle \Phi_0 | e^{i\frac{\hat{H}t}{\hbar}} (\lambda \epsilon_0 \cdot \hat{\mu}) e^{-i\frac{\hat{H}t}{\hbar}} | \Phi_k \rangle \langle \Phi_k | (\lambda \epsilon_0 \cdot \hat{\mu}) | \Phi_0 \rangle,
\]

\[(139)\]

as well as the finite temperature photoabsorption lineshape,

\[
I_\beta(\omega) = \frac{3}{2\pi\hbar} \int_{-\infty}^{\infty} dt e^{i\omega t} \text{Tr} \left[ \hat{\rho} \hat{\omega}(t) \hat{\rho}(0) \right],
\]

\[(140)\]

where \( \hat{\rho} = Z^{-1} \exp(-\beta \hat{H}) \) and \( Z = \text{Tr}[\hat{\rho}] \).

18 Time Dependent Perturbation Theory

The goal of this section is to derive the Golden rule of time-dependent perturbation theory, introduced by Eq. (127) 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 arbitrary state, \( \tilde{\psi}(x, t) = \sum_j C_j \Phi_j(x) e^{-\frac{i}{\hbar} E_j t} \),

\[(141)\]

for the initially unperturbed system described by the Hamiltonian \( \hat{H} \), for which \( \hat{H} \Phi_j = E_j \Phi_j \) and \( i\hbar \frac{\partial \hat{\psi}}{\partial t} = \hat{H} \hat{\psi} \), let us obtain the solution of the time dependent Schrödinger equation:

\[
i\hbar \frac{\partial \hat{\psi}}{\partial t} = [\hat{H} + \lambda \hat{\omega}(t)] \hat{\psi},
\]

\[(141)\]

assuming that such solution can be written as a rapidly convergent expansion in powers of \( \lambda \),

\[
\psi_\lambda(x, t) = \sum_j \sum_{l=0}^{\infty} C_{jl}(t) \lambda^l \Phi_j(x) e^{-\frac{i}{\hbar} E_j t}.
\]

\[(142)\]

Substituting Eq. (142) into Eq. (141) we obtain,

\[
i\hbar \sum_{l=0}^{\infty} \left( \hat{C}_{kl}(t) \lambda^l + C_{kl}(t) \lambda^l (-\frac{i}{\hbar} E_k) \right) e^{-\frac{i}{\hbar} E_k t} = \sum_j \sum_{l=0}^{\infty} C_{jl}(t) \lambda^l (\langle \Phi_k | \phi_j \rangle E_j + \lambda \langle \Phi_k | \hat{\omega} | \Phi_j \rangle) e^{-\frac{i}{\hbar} E_j t}.
\]

\[(143)\]
Terms with $\lambda^0$: (Zero-order time dependent perturbation theory)

$$+i\hbar[\hat{C}_{k_0}(t)e^{-\frac{i}{\hbar}E_k t} + C_{k_0}(t)(-\frac{i}{\hbar} E_k e^{-\frac{i}{\hbar}E_k t})] = \sum_j C_{j_0}(t)\delta_{kj}E_j e^{-\frac{i}{\hbar}E_j t} = C_{k_0}(t)E_k e^{-\frac{i}{\hbar}E_k t}. $$

Since,

$$\hat{C}_{k_0}(t) = 0, \quad \Rightarrow \quad C_{k_0}(t) = C_{k_0}(0). $$

Therefore, the unperturbed wave function is correct to zeroth order in $\lambda$.

Terms with $\lambda^0$: (Zero-order time dependent perturbation theory)

$$+i\hbar[\hat{C}'_{k_0}(t)e^{-\frac{i}{\hbar}E_k t} + C_{k_0}(t)(-\frac{i}{\hbar} E_k e^{-\frac{i}{\hbar}E_k t})] = \sum_j C_{j_0}(t)\delta_{kj}E_j e^{-\frac{i}{\hbar}E_j t} = C_{k_0}(t)E_k e^{-\frac{i}{\hbar}E_k t}. $$

Since,

$$\hat{C}'_{k_0}(t) = 0, \quad \Rightarrow \quad C_{k_0}(t) = C_{k_0}(0). $$

Therefore, the unperturbed wave function is correct to zeroth order in $\lambda$.

Terms with $\lambda$: (First-order time dependent perturbation theory)

$$i\hbar[\hat{C}'_{k_1}(t)e^{-\frac{i}{\hbar}E_{k_1} t} + C_{k_1}(t)(-\frac{i}{\hbar} E_k e^{-\frac{i}{\hbar}E_k t})] = \sum_j C_{j_1}(t)\delta_{kj}E_j e^{-\frac{i}{\hbar}E_j t} + C_{j_0}(t) < \Phi_k|\hat{\omega}|\Phi_j > e^{-\frac{i}{\hbar}E_j t},$$

$$\hat{C}'_{k_1}(t) = -\frac{i}{\hbar} \sum_j \left( C_{j_0}(0) < \Phi_k|\hat{\omega}|\Phi_j > e^{-\frac{i}{\hbar}(E_j-E_k) t} \right).$$

Therefore,

$$\hat{C}'_{k_1}(t) = -\frac{i}{\hbar} \sum_j C_{j_0}(0) < \Phi_k|e^{\frac{i}{\hbar}\hat{H}t}\hat{\omega}e^{-\frac{i}{\hbar}\hat{H}t}|\Phi_j > = -\frac{i}{\hbar} \sum_j C_{j_0}(0) < \Phi_k|e^{\frac{i}{\hbar}\hat{H}t}\hat{\omega}e^{-\frac{i}{\hbar}\hat{H}t}|\Phi_j >, \quad (144)$$

Equation (144) 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. (144) 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:

$$C_{k_1}(t) = -\frac{i}{\hbar} \int_{-\infty}^{t} dt' < \Phi_k|e^{\frac{i}{\hbar}\hat{H}t'}\hat{\omega}e^{-\frac{i}{\hbar}\hat{H}t'}|\hat{\psi}_0 > .$$

This expression gives the correction of the expansion coefficients to first order in $\lambda$. 

46
Note: The substitution made in Eq. (144) can be justified as follows. The exponential function is defined in powers series as follows,

\[ e^A = \sum_{n=0}^{\infty} \frac{A^n}{n!} = 1 + A + \frac{1}{2!} AA + \ldots, \]

In particular, when \( A = -i\hat{H}t/\hbar \),

\[ e^{-\frac{i}{\hbar}\hat{H}t} = 1 + (-\frac{i}{\hbar}\hat{H}t) + \frac{1}{2!}(-\frac{i}{\hbar}t)^2\hat{H}\hat{H} + \ldots. \]

Furthermore, since

\[ \hat{H}\Phi_j > = E_j\Phi_j >, \]

and,

\[ \hat{H}\hat{H}\Phi_j > = E_j^2\Phi_j >, \]

we obtain,

\[ e^{-\frac{i}{\hbar}\hat{H}t}\Phi_j > = [1 + (-\frac{i}{\hbar}E_j t) + \frac{1}{2!}(-\frac{i}{\hbar}t)^2E_j^2 + \ldots]\Phi_j > = e^{-\frac{i}{\hbar}E_j t}\Phi_j >, \]

which is the substitution implemented in Eq. (144).

Terms with \( \lambda^2 \): (Second-order time dependent perturbation theory)

\[ i\hbar[\dot{C}_k(t) + C_k(t)(-\frac{i}{\hbar}E_k)]e^{-\frac{i}{\hbar}E_k t} = \sum_j [C_{j_2}(t)\delta_{kj}E_j + C_{j_1}(t) < \Phi_k|\hat{\omega}|\Phi_j >]e^{-\frac{i}{\hbar}E_j t}, \]

\[ \dot{C}_k(t) = -\frac{i}{\hbar}\sum_j < \Phi_k|e^{\frac{i}{\hbar}\hat{H}t}\hat{\omega}e^{-\frac{i}{\hbar}\hat{H}t}|\Phi_j > C_{j_1}(t), \]

\[ C_k(t) = \left( -\frac{i}{\hbar} \right) \int_{-\infty}^{t} dt' \sum_j < \Phi_k|e^{\frac{i}{\hbar}\hat{H}t'}\hat{\omega}e^{-\frac{i}{\hbar}\hat{H}t'}|\Phi_j > C_{j_1}(t'), \]

\[ C_k(t) = \left( -\frac{i}{\hbar} \right)^2 \sum_j \int_{-\infty}^{t} dt' \int_{-\infty}^{t'} dt'' < \Phi_k|e^{\frac{i}{\hbar}\hat{H}(t'-t'')}\hat{\omega}e^{-\frac{i}{\hbar}\hat{H}(t'-t'')}|\Phi_j > < \Phi_j|e^{\frac{i}{\hbar}\hat{H}(t'-t'')}\hat{\omega}e^{-\frac{i}{\hbar}\hat{H}(t'-t'')}|\tilde{\psi}_0 >. \]

Since \( 1 = \sum_j |\Phi_j > < \Phi_j | \),

\[ C_k(t) = \left( -\frac{i}{\hbar} \right)^2 \int_{-\infty}^{t} dt' \int_{-\infty}^{t'} dt'' < \Phi_k|e^{\frac{i}{\hbar}\hat{H}(t'-t'')}\hat{\omega}e^{-\frac{i}{\hbar}\hat{H}(t'-t'')}|\tilde{\psi}_0 >. \]

This expression gives the correction of the expansion coefficients to second order in \( \lambda \).
19 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,

\[
\overrightarrow{\varepsilon}(t) = \lambda F_1(t - t_0)\varepsilon_0 e^{-i\omega_1 t} + \lambda F_2(t - t_0 - \Delta t)\varepsilon_0 e^{-i\omega_2 t} + c.c. \quad (145)
\]

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 \( \Delta t \) between the pump and probe pulses allows this technique to probe the excited state dynamics at various times \( \Delta t \) after photoexcitation 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 probability to state \( |\Phi_k\rangle \) and then summing over all possible final states as follows:

\[
P_0 = \sum_k P_0^{(k)} = \lim_{t_f \to \infty} \sum_k |c_k^{(2)}(t_f)|^2, \quad (146)
\]

where \( c_k^{(2)}(t_f) \) is defined by second order time-dependent perturbation theory,

\[
c_k^{(2)}(t_f) = \hbar^{-2} \int_{-\infty}^{t_f} dt' \int_{-\infty}^{t'} dt'' \langle \Phi_k | e^{\frac{i}{\hbar} \hat{H} t''} \hat{H}_1(t'') e^{-\frac{i}{\hbar} \hat{H} t'} | \Phi_0 \rangle, \quad (147)
\]

with

\[
\hat{H}_1(t) = -(\lambda \varepsilon_0 \cdot \hat{\mu}) F_1(t - t_0) e^{-i\omega_1 t} - (\lambda \varepsilon_0 \cdot \hat{\mu}) F_2(t - t_0 - \Delta t) e^{-i\omega_2 t} + c.c. \quad (148)
\]

Substituting Eq. (148) into Eq. (147) and then substituting Eq. (146) into Eq. (146) we obtain a sum of 16 terms, associated with all possible pairs of interactions \((\pm \omega_j, \pm \omega_k)\) with \( j = 1, 2 \) and \( k = 1, 2 \). In particular, the term \((+\omega_1, +\omega_2)\) is

\[
P_0(+\omega_1, +\omega_2) = \hbar^{-4} \int_{-\infty}^{\infty} dt \int_{-\infty}^{t} dt' \int_{-\infty}^{t'} dt'' \int_{-\infty}^{t''} dt''' e^{i\omega_1 (t'' - t')} e^{i\omega_2 (t'' - t)} F_1(t' - t_0) \times F_2(t - t_0 - \Delta t) F_1(t'' - t_0) F_2(t'' - t_0 - \Delta t) \langle \Phi_0 | e^{\frac{i}{\hbar} \hat{H} t'} (\lambda \varepsilon_0 \cdot \hat{\mu}) e^{-\frac{i}{\hbar} \hat{H} t'} \rangle \times e^{\frac{i}{\hbar} \hat{H}_1} (\lambda \varepsilon_0 \cdot \hat{\mu}) e^{-\frac{i}{\hbar} \hat{H}_1} (\lambda \varepsilon_0 \cdot \hat{\mu}) e^{-\frac{i}{\hbar} \hat{H}_1} | \Phi_0 \rangle, \quad (149)
\]

and corresponds to the contribution due to absorption at \( \omega_1 \), often promoting the system from the ground state to an intermediate excited state, followed by absorption at \( \omega_2 \) to promote the system to a final state of even higher energy. At low temperature, this term often dominates the total transition probability since the integrands of off-resonant terms are more highly oscillatory. Equation (149) 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 \( \omega_1 \) 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:

\[
I_2(X) + \hbar \omega_1 \rightarrow I_2(B), \\
I_2(B) + \hbar \omega_2 \rightarrow I_2(f).
\]

Considering that the electronic transition dipole moments are independent of nuclear coordinates, we obtain:

\[
I(\omega_2, \Delta t) = \hbar^{-4} (\lambda \vec{\mu}_{01} \cdot \vec{\mu}_{Bx})^2 (\lambda \vec{\mu}_{02} \cdot \vec{\mu}_{fB})^2 \sum_j \rho_j \int_{-\infty}^{\infty} dt \int_{-\infty}^{\infty} dt'' e^{-i\omega_2(t''-t)} \times \int_{-\infty}^{t''} dt' \int_{-\infty}^{\infty} dt''' \int_{-\infty}^{\infty} dt'''' e^{i(E_j + h\omega_1)(t''''-t')F_1(t'-t_0)F_2(t-t_0-\Delta t)} \times F_1(t''-t_0)F_2(t''-t_0-\Delta t)\xi_j(t-t'', t'-t),
\]

where

\[
\xi_j(t-t'', t'-t) = \langle \Phi_j | e^{-\frac{1}{\hbar} \hat{H}_B(t-t'')} e^{\frac{1}{\hbar} \hat{H}_J(t'-t)} e^{\frac{1}{\hbar} \hat{H}_B(t''-t'')} | \Phi_j \rangle.
\]

If desired, Eq. (151) could also be written as

\[
I(\omega_2, \Delta t) = \int_{-\infty}^{\infty} dt e^{i\omega_2 t} C(t, \Delta t),
\]

where \( C(t, \Delta t) \) is readily identifiable from Eq. (151):

\[
C(t, \Delta t) = \hbar^{-4} (\lambda \vec{\mu}_{01} \cdot \vec{\mu}_{Bx})^2 (\lambda \vec{\mu}_{02} \cdot \vec{\mu}_{fB})^2 \sum_j \rho_j \int_{-\infty}^{\infty} dt'' \int_{-\infty}^{\infty} dt' \int_{-\infty}^{\infty} dt''' \int_{-\infty}^{\infty} dt'''' e^{i(E_j + h\omega_1)(t''''-t')} \times F_1(t'-t_0)F_2(t''+t-t_0-\Delta t)F_1(t''-t_0)F_2(t+t''-t_0-\Delta t) \times \xi_j(t''-t'', t, t'-t''-t).
\]

\section{Pump-Probe Photoelectron Spectroscopy}

Pump-probe photoelectron spectroscopy is essentially the same technique discussed Sec. [9] 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 \( \epsilon \) of the photodetached electrons as a function of the time delay \( \Delta t \) 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. [9] 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)].

49
As an example, we consider the photoelectron spectroscopy of $I_2^-$ due to the photoelectron detachment process

$$I_2^-(X) + h\omega_1 \rightarrow I_2^-(A'),$$

$$I_2^-(A') + h\omega_2 \rightarrow I_2(K, \nu) + e^-(\epsilon), \quad (155)$$

where $K$ and $\nu$ indicate the electronic and vibrational states of $I_2$, and $\epsilon$ the kinetic energy of the photodetached electron. The initial state of $I_2^-$ is

$$|\Phi_0\rangle = |\psi_g\rangle|\chi_g\rangle, \quad (156)$$

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

$$|\Phi_f\rangle = |\psi_K\rangle|\chi_{E_K}\rangle, \quad (157)$$

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,

$$e^{-\frac{i}{\hbar}H(t'')} = e^{-\frac{i}{\hbar}\hat{H}_{A'}(t''-t')}|\psi_{A'}\rangle\langle\psi_{A'}|, \quad (158)$$

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. (147), 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,

$$P_K = \left|\frac{\hbar}{2} \int_{-\infty}^{\infty} dt' \int_{-\infty}^{t''} dt'' e^{\frac{i}{\hbar}(E_{K'}+\epsilon)t''} \langle\Phi_f|\hat{H}_1(t'')e^{-\frac{i}{\hbar}\hat{H}_{A'}(t''-t')}\hat{H}_1(t')e^{-\frac{i}{\hbar}\hat{H}_{1}'t'}|\Phi_0\rangle\right|^2. \quad (159)$$

and the probability of a 2-photon transition to the electronic state $K$, leaving the photodetached electron with kinetic energy $\epsilon$ is

$$P_K(\epsilon) = \hbar^{-4} \int dE_K \left|\frac{\hbar}{2} \int_{-\infty}^{\infty} dt' \int_{-\infty}^{t''} dt'' e^{\frac{i}{\hbar}(E_{K'}+\epsilon)t''} \langle\Phi_f|\hat{H}_1(t'')e^{-\frac{i}{\hbar}\hat{H}_{A'}(t''-t')}\hat{H}_1(t')e^{-\frac{i}{\hbar}\hat{H}_{1}'t'}|\Phi_0\rangle\right|^2. \quad (160)$$

Therefore, the total probability of a 2-photon transition, leaving the photodetached electron with kinetic energy $\epsilon$ is $P(\epsilon) = \sum_K P_K(\epsilon)$:

$$P(\epsilon) = \hbar^{-4} \sum_K \int dE_K \left|\frac{\hbar}{2} \int_{-\infty}^{\infty} dt' \int_{-\infty}^{t''} dt'' e^{\frac{i}{\hbar}(E_{K'}+\epsilon)t''} \langle\Phi_f|\hat{H}_1(t'')e^{-\frac{i}{\hbar}\hat{H}_{A'}(t''-t')}\hat{H}_1(t')e^{-\frac{i}{\hbar}\hat{H}_{1}'t'}|\Phi_0\rangle\right|^2. \quad (161)$$
Explicitly squaring the r.h.s. of Eq. (161) and using the relation,
\[ \int dE_K |\chi_{E_K}\rangle \langle \chi_{E_K}| e^{-\frac{i}{\hbar}E_K(t''-t)} = e^{-\frac{i}{\hbar}H_K(t''-t)}, \]
(162)
we obtain the \((+\omega_1, +\omega_2)\) term, analogous to Eq. (149):
\[
P(\epsilon) = \hbar^{-4} \sum_{K} \int_{-\infty}^{\infty} dt \int_{-\infty}^{t} dt' \int_{-\infty}^{\infty} dt'' \int_{-\infty}^{t''} dt''' e^{i\omega_1(t''-t')} e^{i\omega_2(t''-t)} F_1(t' - t_0) F_2(t - t_0 - \Delta t) \]
\[
\times F_1(t''' - t_0) F_2(t'' - t_0 - \Delta t) e^{i\epsilon(t''-t)} \langle \Phi_0| e^{\frac{i}{\hbar}H_X t'''} (\lambda_{\epsilon_{01}} \cdot \hat{\mu}_{X'K}) e^{-\frac{i}{\hbar}H_{A'} t''} \]
\[
\times e^{\frac{i}{\hbar}H_{A'} t'} (\lambda_{\epsilon_{02}} \cdot \hat{\mu}_{A'K}) e^{-\frac{i}{\hbar}H_X t'} |\Phi_0\rangle,
\]
(163)
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 \((\pm \omega_j, \pm \omega_j)\) 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. (151):
\[
P_\beta(\epsilon, \Delta t) = \hbar^{-4} (\lambda_{\epsilon_{01}} \cdot \hat{\mu}_{X'K})^2 (\lambda_{\epsilon_{02}} \cdot \hat{\mu}_{A'K})^2 \sum_j \rho_j \int_{-\infty}^{\infty} dt \int_{-\infty}^{\infty} dt'' e^{i(\omega_2 + \epsilon)(t''-t)} \]
\[
\times \int_{-\infty}^{t} dt' \int_{-\infty}^{t''} dt''' e^{\frac{i}{\hbar}E_j(t''-t')} F_1(t' - t_0) F_2(t - t_0 - \Delta t) \]
\[
\times F_1(t''' - t_0) F_2(t'' - t_0 - \Delta t) \xi_j(t - t'''', t'' - t', t'' - t'),
\]
(164)
where
\[
\xi_j(t - t'''', t'' - t', t'' - t') = \langle \Phi_2| e^{\frac{i}{\hbar}H_{A'}(t-t''')} e^{\frac{i}{\hbar}H_K(t'''-t)} e^{\frac{i}{\hbar}H_{A'}(t'-t'')} |\Phi_j\rangle,
\]
(165)
and \(\rho_j = Z^{-1} \exp(-\beta E_j)\), with \(Z = \sum_j \exp(-\beta E_j)\). If desired, Eq. (164) could also be written as
\[
P_\beta(\epsilon, \Delta t) = \int_{-\infty}^{\infty} dt \rho_j e^{i\epsilon t} C(t, \Delta t),
\]
(166)
where \(C(t, \Delta t)\) is readily identifiable from Eq. (164):
\[
C(t, \Delta t) = \hbar^{-4} (\lambda_{\epsilon_{01}} \cdot \hat{\mu}_{X'K})^2 (\lambda_{\epsilon_{02}} \cdot \hat{\mu}_{A'K})^2 \sum_j \rho_j \int_{-\infty}^{\infty} dt'' \int_{-\infty}^{t''} dt''' \int_{-\infty}^{t+t''} dt''' e^{\frac{i}{\hbar}(E_j + \omega_1)(t'''-t')} \]
\[
\times F_1(t' - t_0) F_2(t'' + t - t_0 - \Delta t) F_1(t''' - t_0) F_2(t + t'' - t_0 - \Delta t) e^{\frac{i}{\hbar}\omega_2 t} \]
\[
\times \xi_j(t'' - t'''', t, t'' - t' - t'' - t).
\]
(167)
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 [17]. 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],

$$I_2^-(X) + h\omega_1 \rightarrow I_2(K, \nu) + e^-(\epsilon).$$  \hspace{1cm} (168)

As in Sec. [20], the initial state of $I_2^-$ is

$$|\Phi_0\rangle = |\psi_g\rangle|\chi_g\rangle,$$  \hspace{1cm} (169)

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

$$|\Phi_f\rangle = |\psi_K\rangle|\chi_{E_K}\rangle,$$  \hspace{1cm} (170)

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. (139), the photoabsorption lineshape (at 0 K) for the one-photon photodetachment process is,

$$P_0(\epsilon) = \frac{3}{2\pi\hbar} \sum_K \int dE_K \int_{-\infty}^{\infty} dt e^{i\omega t} \langle \Phi_0 | e^{i\hat{H}_K} (\lambda \vec{e}_0 \cdot \hat{\mu}) e^{-i\hat{H}_K(\nu + \epsilon)t} | \Phi_k \rangle \langle \Phi_k | (\lambda \vec{e}_0 \cdot \hat{\mu}) | \Phi_0 \rangle,$$  \hspace{1cm} (171)

and using the relation

$$\int dE_K |\chi_{E_K}\rangle \langle \chi_{E_K}| e^{-i\hat{H}_{E_K}t} = e^{-i\hbar H_{E_K}t},$$  \hspace{1cm} (172)

the finite temperature distribution is

$$P_\beta(\epsilon) = \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} dt e^{-\hbar t} C(t),$$  \hspace{1cm} (173)

with

$$C(t) = 3 \sum_j \rho_j e^{i(E_j + \hbar\omega t)} \sum_K \langle \Phi_j | (\lambda \vec{e}_0 \cdot \hat{\mu}) e^{-i\hat{H}_Kt} (\lambda \vec{e}_0 \cdot \hat{\mu}) | \Phi_j \rangle.$$  \hspace{1cm} (174)

21.1 Computational Problem 17

17.1. Compute the photoabsorption spectrum of $I_2$. Assume that the transition dipole moments are independent of nuclear coordinates, and that the only allowed electronic transition induced by photoabsorption of $I_2$ is the $B \leftarrow X$ excitation. Assume the ground (g) and excited (e) states of $I_2$ can be described by the Morse Potential $V(R) = D_e \left(1 - e^{-\beta(R - R_{eq})}\right)^2 + V_0$, where $R$ is the bond-length of $I_2$ and $V_0(g) = 0.00$ eV; $V_0(e) = 0.94$ eV; $D_e(g) = 18941$ cm$^{-1}$; $D_e(e) = 4911$ cm$^{-1}$; $\beta(g) = 1.517$ Å$^{-1}$; $\beta(e) = 1.535$ Å$^{-1}$; $R_{eq}(g) = 2.66$ Å and $R_{eq}(e) = 3.105$ Å.
17.2. Compute the direct photoelectron detachment spectrum of \( I_2^- \) assuming that the electronic transitions induced by photoelectron detachment of \( I_2^- (X) \) generate \( I_2 \) in the electronic states \( X \) and \( B \).

Assume that the potential energy surfaces of the states \( I_2^- (X) \), \( I_2 (X) \) and \( I_2 (B) \) can be described by simple Morse potentials, as reported by Batista and Coker \([J. Chem. Phys. (1997) 106:7102-7116]\).
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:

$$T^{(\delta)}(j, j') = \frac{\hbar^2 \pi^2}{2m(x_{\text{max}} - x_{\text{min}})^2} \frac{2}{N} \sum_{k=1}^{N-1} k^2 \sin \left( \frac{k \pi j}{N} \right) \sin \left( \frac{k \pi j'}{N} \right),$$

(175)

where the delta functions $\delta(x - x_j)$ are equally spaced as follows:

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

(167)

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,

$$H_0(x, p) = \frac{p^2}{2} - \alpha(x^2 - \beta x^4),$$

(177)

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 $2-\pi$ pulses when each pulse is described by the following operator:

$$\hat{U}^{2\pi} = 1 - 2|\Phi_0 \rangle \langle \Phi_0|,$$

(178)

where $\Phi_0(x)$ is the initial state defined as follows:

$$\Phi_0(x) = \frac{1}{\sqrt{2}} (\chi_0(x) + \chi_1(x)),$$

(179)

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:

$$I_0(\omega) = \frac{3}{2\pi} \int_{-\infty}^{\infty} dt e^{i(\omega \omega + E_0)t} \xi(t),$$

(180)

where $|\Psi_0\rangle = \lambda \vec{e}_0 \cdot \hat{\mu} |\Phi_0\rangle$, 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,

$$\vec{e}(t) = \lambda \vec{e}_0 (e^{i\omega t} + e^{-i\omega t}),$$

(181)

where $\lambda << 1$ is a small dimensionless parameter that defines the dipolar interaction,

$$\hat{H}_1(t) = -\lambda \vec{e}_0 \cdot \hat{\mu} (e^{i\omega t} + e^{-i\omega t}),$$

(182)

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 probability to a generic state $|\Phi_k\rangle$ and then summing the contributions from all possible final states $|\Phi_k\rangle$ as follows:

$$P_0 = \sum_k P_0^{(k)} = \lim_{t_f \to \infty} \sum_k |c_k^{(2)}(t_f)|^2.$$  

(183)

Prove that, according to second order time-dependent perturbation theory, $c_k^{(2)}(t_f)$ is defined as follows:

$$c_k^{(2)}(t_f) = \hbar^{-2} \int_{-\infty}^{t_f} dt' \int_{-\infty}^{t_f'} dt'' |\Phi_k\rangle e^{\frac{i}{\hbar} \hat{H} t'} \hat{H}_1(t'') e^{-\frac{i}{\hbar} \hat{H} (t'' - t')} \hat{H}_1(t') e^{-\frac{i}{\hbar} \hat{H} t'} |\Phi_0\rangle,$$

(184)

where $\hat{H}_1(t)$ is the dipolar radiation-matter interaction.
22.1 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,

$$ H\chi_j - C_j E_j = 0, \quad (185) $$

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:

$$ x_j = x_{\text{min}} + j \Delta, \quad \text{with} \quad \Delta = (x_{\text{max}} - x_{\text{min}})/N, \quad (186) $$

with $j = 1$–(N-1). In such a representation, the Hamiltonian matrix elements are:

$$ H(j, j') = V(x_j) \delta_{jj'} + T(j, j'), \quad (187) $$

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:

$$ T^{(\delta)}(j, j') = \frac{\hbar^2 \pi^2}{2m(x_{\text{max}} - x_{\text{min}})^2} \frac{1}{N} \sum_{k=1}^{N-1} k^2 \sin \left( \frac{k \pi j}{N} \right) \sin \left( \frac{k \pi j'}{N} \right), \quad (188) $$

where the delta functions $\delta(x - x_j)$ are equally spaced as follows:

$$ x_j = x_{\text{min}} + j \Delta, \quad \text{with} \quad \Delta = (x_{\text{max}} - x_{\text{min}})/N, \quad (189) $$

with $j = 1$–(N-1).

We consider the Hamiltonian,

$$ \hat{H} = \hat{T} + V(\hat{x}), \quad (190) $$

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

$$ V^{(\delta)}(j, k) = \langle j | V(\hat{x}) | k \rangle = \int dx \delta^*(x - x_j) V(\hat{x}) \delta(x - x_k), \quad (191) $$

$$ = V(x_k) \delta_{jk}. $$
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 \( x = (x_{\text{min}}, x_{\text{max}}) \), and then rotating \( T^{(\phi)} \) to the representation of delta functions by using the following similarity transformation:

\[
T^{(\delta)} = \Gamma^{-1} T^{(\phi)} \Gamma,
\]

(192)

where \( \Gamma \) is the transformation matrix defined by the linear combinations,

\[
\phi_k(x) = \sum_j \Gamma(j, k) \delta(x - x_j) \Delta',
\]

(193)

where

\[
\Gamma(j, k) = \phi_k(x_j).
\]

(194)

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:

\[
\phi_k(x) = \sqrt{\frac{2}{x_{\text{max}} - x_{\text{min}}}} \text{Sin} \left( k \frac{\pi (x - x_{\text{min}})}{x_{\text{max}} - x_{\text{min}}} \right),
\]

(195)

with \( \phi_k(x_{\text{min}}) = 0 \) and \( \phi_k(x_{\text{max}}) = 0 \). Therefore,

\[
\hat{T} \phi_k(x) = \frac{(\hbar \pi k)^2}{2m} \phi_k(x),
\]

(196)

and \( T^{(\phi)} \) is diagonal with matrix elements,

\[
\langle j, i | \hat{T} | k, i \rangle = \langle \phi_j | \hat{T} | \phi_k \rangle = \langle \phi_j | \phi_k \rangle = \frac{(\hbar k)^2}{2m} \frac{\pi^2}{(x_{\text{max}} - x_{\text{min}})^2} \delta_{jk}.
\]

(197)

Therefore, substituting Eq. (197) and Eq. (194) into Eq. (192) we obtain,

\[
T^{(\delta)}(i, i') = \sum_{j, k=1}^{N-1} \Gamma^{-1}(i, j) T^{(\phi)}(j, k) \Gamma(k, i') = \sum_{j, k=1}^{N-1} \Gamma(j, i) T^{(\phi)}(j, k) \Gamma(k, i'),
\]

\[
= \frac{\Delta \pi^2}{(x_{\text{max}} - x_{\text{min}})^2} \sum_{j, k=1}^{N-1} \phi_j(x_i) \frac{(h k)^2}{2m} \delta_{jk} \phi_k(x_i') = \frac{\Delta \pi^2}{(x_{\text{max}} - x_{\text{min}})^2} \sum_{k=1}^{N-1} \phi_k(x_i) \frac{(h k)^2}{2m} \phi_k(x_i'),
\]

\[
= \frac{\Delta \hbar^2 \pi^2}{2m(x_{\text{max}} - x_{\text{min}})^2 (x_{\text{max}} - x_{\text{min}})} \sum_{k=1}^{N-1} k^2 \text{Sin} \left( k \pi \frac{(x_i - x_{\text{min}})}{x_{\text{max}} - x_{\text{min}}} \right) \text{Sin} \left( k \pi \frac{(x_i' - x_{\text{min}})}{x_{\text{max}} - x_{\text{min}}} \right).
\]

(198)

Finally, substituting Eq. (189) into Eq. (198) we obtain:

\[
T^{(\delta)}(j, j') = \frac{\hbar^2 \pi^2}{2m(x_{\text{max}} - x_{\text{min}})^2} \sum_{k=1}^{N-1} k^2 \text{Sin} \left( k \pi \frac{j}{N} \right) \text{Sin} \left( k \pi \frac{j'}{N} \right).
\]

(199)
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,

\[ H_0(x, p) = \frac{p^2}{2} - \alpha(x^2 - \beta x^4), \]

(200)

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 \( \Omega \). To show this, we consider the propagation of the initial state,

\[ \Phi_0(x) = c_0 \chi_0(x) + c_1 \chi_1(x) + \ldots, \]

(201)

where \( \chi_0(x) \) and \( \chi_1(x) \) are the ground and first excited states of the double-well. Since \( \hat{H} | \chi_j \rangle = E_j | \chi_j \rangle \),

\[ | \Phi_t \rangle = c_0 \chi_0(x)e^{-iE_0 t} + c_1 \chi_1(x)e^{-iE_1 t} + \ldots, \]

(202)

and

\[ |\xi(t)|^2 = |\langle \Phi_0 | \Phi_t \rangle|^2 = |c_0|^4 + |c_1|^4 + 2|c_0 c_1|^2 \cos[\Omega t] + \ldots, \]

(203)

where \( \Omega = (E_1 - E_0)/\hbar \) is the tunneling frequency. Therefore, according to Eq. (203), the Fourier transform of \( \xi(t) \) should have a prominent peak at \( \Omega \).

(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-\( \pi \) pulses when each pulse is described by the following operator:

\[ \hat{U}^{2\pi} = 1 - 2|\Phi_0 \rangle \langle \Phi_0 |, \]

(204)

where \( \Phi_0(x) \) is the initial state defined as follows:

\[ \Phi_0(x) = \frac{1}{\sqrt{2}} (\chi_0(x) + \chi_1(x)), \]

(205)

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-\( \pi \) pulses, applied at 2\( \tau \) intervals, generates the time-evolved state,

\[ |\Psi_{t+2N\tau} \rangle = c_0 \left( e^{-i\hat{H}\tau} \hat{U}^{2\pi} e^{-i\frac{1}{2}\hat{H}\tau} \right)^N |\Phi_0 \rangle + e^{-i\frac{1}{2}\hat{H}2N\tau} (c_1 |\Phi_1 \rangle + \ldots), \]

(206)

\[ = c_0(-1)^N e^{-i(E_0+E_1)2N\tau} |\Phi_0 \rangle + e^{-i\frac{1}{2}\hat{H}2N\tau} (c_1 |\Phi_1 \rangle + \ldots). \]
The second equality in Eq. (206) is obtained by substituting $\hat{U}^{2\pi}$ as defined by Eq. (204) and $\Phi_0$ according to Eq. (205).

Equation (206) shows that the square of the expansion coefficient associated with state $\Phi_0$ remains constant, for as long as the train of 2-$\pi$ 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:

$$I_0(\omega) = \frac{3}{2\pi \hbar} \int_{-\infty}^{\infty} dt e^{\frac{i}{\hbar}(\hbar \omega + E_0)t} \xi(t),$$

(207)

where $|\Psi_0\rangle = \lambda \vec{\varepsilon}_0 \cdot \hat{\mu} |\Phi_0\rangle$, 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,

$$\vec{\epsilon}(t) = \lambda \vec{\varepsilon}_0 (e^{i\omega t} + e^{-i\omega t}),$$

(208)

where $\lambda << 1$ is a small dimensionless parameter that defines the dipolar interaction,

$$\hat{H}_1(t) = -\lambda \vec{\varepsilon}_0 \cdot \hat{\mu} (e^{i\omega t} + e^{-i\omega t}),$$

(209)

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:

$$P_0^{(k)}(\omega) = \lim_{t \to \infty} |c_k^{(1)}(t)|^2,$$

(210)

where $c_k^{(1)}(t)$ is defined by the Golden Rule expression of first order time-dependent perturbation theory.

$$c_k^{(1)}(t) = -\frac{i}{\hbar} \int_{-\infty}^{t} dt' \langle \Phi_k | e^{\frac{i}{\hbar} \hat{H} t'} \hat{H}_1(t') e^{-\frac{i}{\hbar} \hat{H} t'} | \Phi_0 \rangle.$$

(211)

Substituting the expression of the dipolar interaction, introduced by Eq. (209), into Eq. (211), we obtain:

$$c_k^{(1)}(t) = -\frac{i}{\hbar} \int_{-\infty}^{t} dt' \langle \Phi_k | \lambda \vec{\varepsilon}_0 \cdot \hat{\mu} \left[ e^{-\frac{i}{\hbar}(\hat{H} - E_k - \hbar \omega)t'} + e^{-\frac{i}{\hbar}(\hat{H} - E_k + \hbar \omega)t'} \right] | \Phi_0 \rangle,$$

(212)

and substituting Eq. (212) into Eq. (210) we obtain:

$$P_0^{(k)}(\omega) = |\langle \Phi_k | \lambda \vec{\varepsilon}_0 \cdot \hat{\mu} | \Phi_0 \rangle|^2 \left[ \delta(E_0 + \hbar \omega - E_k) + \delta(E_0 - \hbar \omega - E_k) \right].$$

(213)
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:

$$\alpha_0(\omega) = \sum_k (E_k - E_0)|\langle \Phi_k | \overrightarrow{\lambda} \epsilon_0 \cdot \hat{\mu} | \Phi_0 \rangle|^2 \left[ \delta(E_0 + \hbar \omega - E_k) + \delta(E_0 - \hbar \omega - E_k) \right].$$ (214)

The absorption spectrum $\alpha_{\beta}(\omega)$, at finite temperature $T = 1/(\beta k_B)$, can be obtained from Eq. (214) as follows:

$$\alpha(\omega) = \sum_j \rho_j \sum_k (E_k - E_j) \delta(E_j + \hbar \omega - E_k)|\langle \Phi_k | \lambda \overrightarrow{\epsilon}_0 \cdot \hat{\mu} | \Phi_j \rangle|^2$$

$$+ \sum_j \rho_j \sum_k \sum_k (E_k - E_j) \delta(E_j - \hbar \omega - E_k)|\langle \Phi_k | \overrightarrow{\lambda} \epsilon_0 \cdot \hat{\mu} | \Phi_j \rangle|^2;$$ (215)

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. (215) and noting that $\rho_k = \rho_j e^{-\beta(E_k - E_j)}$ we obtain:

$$\alpha(\omega) = \sum_j \sum_k \rho_j (E_k - E_j) \delta(E_j + \hbar \omega - E_k)|\langle \Phi_k | \lambda \overrightarrow{\epsilon}_0 \cdot \hat{\mu} | \Phi_j \rangle|^2$$

$$- \rho_j e^{-\beta(E_k - E_j)} (E_k - E_j) \delta(E_j + \hbar \omega - E_k)|\langle \Phi_j | \overrightarrow{\lambda} \epsilon_0 \cdot \hat{\mu} | \Phi_k \rangle|^2;$$ (216)

which gives the absorption lineshape

$$I(\omega) = \frac{3\alpha(\omega)}{\hbar \omega (1 - e^{-\beta \omega})} = 3 \sum_j \sum_k \rho_j \delta(E_j + \hbar \omega - E_k)|\langle \Phi_k | \lambda \overrightarrow{\epsilon}_0 \cdot \hat{\mu} | \Phi_j \rangle|^2.$$ (217)

At 0 K, the absorption lineshape is obtained from Eq. (217) as follows:

$$I_0(\omega) = 3 \sum_k \delta(E_0 + \hbar \omega - E_k)|\langle \Phi_k | \lambda \overrightarrow{\epsilon}_0 \cdot \hat{\mu} | \Phi_0 \rangle|^2,$$ (218)

that is equivalent to Eq. (207), since according to Eq. (207),

$$I_0(\omega) = \frac{3}{2\pi \hbar} \int_{-\infty}^{\infty} dt \langle \Phi_0 | (\lambda \overrightarrow{\epsilon}_0 \cdot \hat{\mu}) e^{\frac{t}{\hbar} (\hbar \omega + E_0 - \hat{H})} (\lambda \overrightarrow{\epsilon}_0 \cdot \hat{\mu}) | \Phi_0 \rangle.$$ (219)

**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 probability to a generic state $|\Phi_k\rangle$ and then summing the contributions from all possible final states $|\Phi_k\rangle$ as follows:

$$P_0 = \sum_k P_0^{(k)} = \lim_{t_f \to \infty} \sum_k |c_k^{(2)}(t_f)|^2.$$ (220)
Prove that, according to second order time-dependent perturbation theory, \( c_k^{(2)}(t_f) \) is defined as follows:

\[
c_k^{(2)}(t_f) = \hbar^{-2} \int_{-\infty}^{t_f} dt' \int_{-\infty}^{t'} dt'' \langle \Phi_k | e^{\frac{i}{\hbar} \hat{H} t''} \hat{H}_1(t'' - t') e^{-\frac{i}{\hbar} \hat{H} t'} | \Phi_0 \rangle,
\]

(221)

where \( \hat{H}_1(t) \) is the dipolar radiation-matter interaction.

Given an arbitrary 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 \( \hat{H} \Phi_j = E_j \Phi_j \) and \( i\hbar \frac{\partial \tilde{\psi}}{\partial t} = \hat{H} \tilde{\psi} \), let us obtain the solution of the time dependent Schrödinger equation:

\[
i\hbar \frac{\partial \psi}{\partial t} = [\hat{H} + \lambda \tilde{\omega}(t)] \psi,
\]

(222)

assuming that such a solution can be written as a rapidly convergent expansion in powers of \( \lambda \),

\[
\psi_\lambda(x, t) = \sum_j \sum_{l=0}^{\infty} C_{jl}(t) \lambda^l \Phi_j(x) e^{-\frac{i}{\hbar} E_j t}.
\]

(223)

Substituting Eq. (223) into Eq. (222) we obtain,

\[
i\hbar \sum_{l=0}^{\infty} \left( \hat{C}_{kl}(t) \lambda^l + C_{kl}(t) \lambda^l (-\frac{i}{\hbar} E_k) \right) e^{-\frac{i}{\hbar} E_k t} = \sum_j \sum_{l=0}^{\infty} C_{jl}(t) \lambda^l \langle \Phi_k \Phi_j \rangle E_j + \lambda \langle \Phi_k | \tilde{\omega} | \Phi_j \rangle \rangle e^{-\frac{i}{\hbar} E_j t}.
\]

(224)

Terms with \( \lambda^2 \): (Second-order time dependent perturbation theory)

\[
i\hbar [\hat{C}_{k2}(t) + C_{k2}(t)(-\frac{i}{\hbar} E_k)] e^{-\frac{i}{\hbar} E_k t} = \sum_j \left[ C_{j2}(t) \delta_{kj} E_j + C_{j1}(t) \langle \Phi_k | \tilde{\omega} | \Phi_j \rangle \rangle \right] e^{-\frac{i}{\hbar} E_j t},
\]

\[
\hat{C}_{k2}(t) = -\frac{i}{\hbar} \sum_j \langle \Phi_k | e^{\frac{i}{\hbar} \hat{H} t} \tilde{\omega} e^{-\frac{i}{\hbar} \hat{H} t} | \Phi_j \rangle \rangle \rangle C_{j1}(t),
\]

\[
C_{k2}(t) = \left( -\frac{i}{\hbar} \right)^2 \int_{-\infty}^{t_f} dt' \int_{-\infty}^{t'} dt'' \langle \Phi_k | e^{\frac{i}{\hbar} \hat{H} t''} \tilde{\omega} e^{-\frac{i}{\hbar} \hat{H} t'} | \Phi_j \rangle \rangle \rangle < \Phi_j \rangle \rangle e^{\frac{i}{\hbar} \hat{H} t''} \tilde{\omega} e^{-\frac{i}{\hbar} \hat{H} t'} | \tilde{\psi}_0 \rangle \rangle.
\]

Since \( 1 = \sum_j | \Phi_j \rangle \rangle < \Phi_j | \),

\[
C_{k2}(t) = \left( -\frac{i}{\hbar} \right)^2 \int_{-\infty}^{t} dt' \int_{-\infty}^{t'} dt'' \langle \Phi_k | e^{\frac{i}{\hbar} \hat{H} t''} \tilde{\omega} e^{-\frac{i}{\hbar} \hat{H} (t' - t'')} \tilde{\omega} e^{-\frac{i}{\hbar} \hat{H} t'} | \tilde{\psi}_0 \rangle \rangle.
\]

This expression gives the correction of the expansion coefficients to second order in \( \lambda \).
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 $\xi_j$, 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 $\xi^0_j$. 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 $(\xi_j - \xi^0_j)$:

$$V(\xi_1, \xi_2, ..., \xi_{3N-6}) = V(\xi^0_1, \xi^0_2, ..., \xi^0_{3N-6}) + \frac{1}{2} \sum_{j=1}^{3N-6} \mu_j \omega_j^2 \left( \xi_j - \xi^0_j \right)^2,$$

with $\mu_j$ and $\omega_j$ 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:

$$V(\mathbf{r}, \theta, \phi, \mathbf{d}) = V_{str}(\mathbf{r}) + V_{bend}(\theta) + V_{torsion}(\phi) + V_{non-bond}(\mathbf{d}),$$

in terms of quadratic expansions in powers of bond-lengths $\mathbf{r}$, bending angles $\theta$, torsion angles $\phi$, and distances $\mathbf{d}$ for non-bonding interactions. The model potential introduced by Eq. (226) is inspired in molecular mechanics models where atoms and bonds are described as spheres and springs. 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:

$$B^- + HA \rightarrow BH + AY^-.$$
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\}_i 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 $q$, subject to the constraints of fixed values for $r_1$ and $r_2$:

$$
\frac{\partial V(r_1, r_2, ..., q_i, ...)}{\partial q_i} = 0,
$$

$$
(228)
$$

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. (228), are the equilibrium positions $q_j^0$ when the system is on the reaction surface $V_0(r_1, r_2) = V(r_1, r_2, ..., q_j^0, ...)$.

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:

$$
V(r_1, r_2, q_3, ..., q_{3N-6}) = V_0(r_1, r_2) + \sum_{j,k=3}^{3N-6} \frac{1}{2} (q_j - q_j^0) \left( \frac{\partial^2 V}{\partial q_j \partial q_k}(q_j = q_j^0) \right) (q_k - q_k^0).
$$

$$
(229)
$$

The potential has no linear term in $(q_k - q_k^0)$ because the first order derivatives are equal to zero (see Eq. (228)) by definition of $V_0(r) = V_0(r_1, r_2)$.

The complete reaction surface Hamiltonian that can be used for multidimensional quantum dynamics simulations is

$$
H(r, P_r, q, P_q) = \sum_{j=1}^{2} \frac{P_r^2}{2\mu_j} + \sum_{j=3}^{3N-6} \frac{P_q^2}{2\mu_j} + \sum_{j=3}^{3N-6} \sum_{k=3}^{3N-6} \sum_{j/k} \frac{1}{2} (q_j - q_j^0(r)) \left( \frac{\partial^2 V(r, q)}{\partial q_j \partial q_k}(q_k = q_k^0(r)) \right).
$$

$$
(230)
$$

24 Wigner Transform Formulation of Quantum Dynamics


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

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

$$
(231)
$$
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 $p$:

$$\int_{-\infty}^{\infty} dp \rho_t^W(p,q) = \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} dq \int_{-\infty}^{\infty} ds \, e^{ips/\hbar} \Psi_t^*(q + s/2) \Psi_t(q - s/2),$$

$$= \int_{-\infty}^{\infty} ds \, \delta(s) \Psi_t^*(q + s/2) \Psi_t(q - s/2),$$

$$= \Psi_t^*(q) \Psi_t(q),$$

$$= |\Psi_t(q)|^2. \quad (232)$$

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

$$\int_{-\infty}^{\infty} dp \int_{-\infty}^{\infty} dq \rho_t^W(p,q) = \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} dq \int_{-\infty}^{\infty} ds \, e^{ips/\hbar} \Psi_t^*(q + s/2) \Psi_t(q - s/2),$$

$$= \int_{-\infty}^{\infty} dx' \int_{-\infty}^{\infty} dx \, e^{ip(x-x')/\hbar} \Psi_t^*(x) \Psi_t(x'),$$

$$= \left| \frac{1}{\sqrt{2\pi\hbar}} \int_{-\infty}^{\infty} dx \, e^{-ipx/\hbar} \Psi_t(x) \right|^2,$$

$$= |\tilde{\Psi}_t(p)|^2. \quad (233)$$

where in the second line of Eq. (233) we introduced the variable transformation $x \equiv q + s/2$ and $x' \equiv q - s/2$, with Jacobian $\det\left[\frac{\partial(q,s)}{\partial(x,x')}\right] = 1$. 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:

$$E_t = \frac{\int dq \int dp \rho_t^W(p,q) H(q,p)}{\int dq \int dp \rho_t^W(p,q)}. \quad (234)$$

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

$$\frac{\partial \Psi_t(q \pm s/2)}{\partial t} = -\frac{1}{i\hbar} \frac{\hbar^2}{2m} \frac{\partial^2 \Psi_t(q \pm s/2)}{\partial q^2} + \frac{1}{i\hbar} V(q \pm s/2) \Psi_t(q \pm s/2),$$

$$= -\frac{1}{i\hbar} \frac{\hbar^2}{2m} \frac{\partial^2 \Psi_t(q \pm s/2)}{\partial s^2} + \frac{1}{i\hbar} V(q \pm s/2) \Psi_t(q \pm s/2), \quad (235)$$

64
where in the second line of Eq. (236) 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

\[
\frac{\partial \rho_t^W(p, q)}{\partial t} = \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} ds \ e^{\frac{i}{\hbar}ps} \left[ \frac{\partial \Psi_t^*(q + s/2)}{\partial t} \Psi_t(q - s/2) + \Psi_t^*(q + s/2) \frac{\partial \Psi_t(q - s/2)}{\partial t} \right],
\]

and

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

\[
- \frac{i}{\hbar} \frac{\hbar^2}{2m} 4 \left[ \Psi_t(q - s/2) \frac{\partial^2}{\partial s^2} \Psi_t(q + s/2) - \Psi_t^*(q + s/2) \frac{\partial^2}{\partial s^2} \Psi_t(q - s/2) \right],
\]

\[
\frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} ds \ e^{\frac{i}{\hbar}ps} \left\{ \frac{i}{\hbar} \left[ V(q + s/2) - V(q - s/2) \right] \Psi_t^*(q + s/2) \Psi_t(q - s/2) 
\right.
\]

\[
- \frac{i}{\hbar} \frac{\hbar^2}{2m} 2 \frac{\partial}{\partial s} \left[ \Psi_t(q - s/2) \frac{\partial}{\partial q} \Psi_t(q + s/2) + \Psi_t^*(q + s/2) \frac{\partial}{\partial q} \Psi_t(q - s/2) \right].
\]

Therefore,

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

\[
+ \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} ds \ e^{\frac{i}{\hbar}ps} \frac{\hbar}{m} \frac{\partial}{\partial s} \frac{\partial}{\partial q} \left[ \Psi_t(q - s/2) \Psi_t^*(q + s/2) \right].
\]

Integrating by parts the second line of Eq. (237), using \( \int_a^b uv = uv|_a^b - \int_a^b vdu \) with \( u \equiv e^{\frac{i}{\hbar}ps} \) and \( v \equiv \frac{\hbar}{m} \frac{\partial}{\partial q} \left[ \Psi_t(q - s/2) \Psi_t^*(q + s/2) \right], \) we obtain:

\[
\frac{\partial \rho_t^W(p, q)}{\partial t} = \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} ds \ e^{\frac{i}{\hbar}ps} \frac{i}{\hbar} \left[ V(q + s/2) - V(q - s/2) \right] \Psi_t^*(q + s/2) \Psi_t(q - s/2)
\]

\[
- \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} ds \ e^{\frac{i}{\hbar}ps} \frac{p}{m} \frac{\partial}{\partial q} \left[ \Psi_t(q - s/2) \Psi_t^*(q + s/2) \right],
\]

\[
\frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} ds \ e^{\frac{i}{\hbar}ps} \frac{i}{\hbar} \left[ V(q + s/2) - V(q - s/2) \right] \Psi_t^*(q + s/2) \Psi_t(q - s/2)
\]

\[
- \frac{p}{m} \frac{\partial \rho_t^W(p, q)}{\partial q}.
\]

65
24.1 Linearized Wigner Transform:

To find a solution of Eq. (238) in terms of coordinates and momenta \( \{ q(t), p(t) \} \) that obey the usual Hamilton’s equations, we express \( \Psi^*_t(q) \) in real valued polar coordinates:

\[
\Psi^*_t(q) = \rho(q(t))^{1/2} e^{\frac{i}{\hbar} S_t(q)}, \tag{239}
\]

and we introduce the following linearization:

\[
\Psi_t(q \pm s/2) \approx \left[ \rho(q(t))^{1/2} \pm \frac{s}{2} \frac{\partial}{\partial q} \rho(q(t))^{1/2} \right] e^{\frac{i}{\hbar} \left[ S_t(q) \pm \frac{s}{2} m v(q; t) \right]}, \tag{240}
\]

24.2 Motion of Auxiliary Variables:

To find a solution of Eq. (238) 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:

\[
\Psi^*_t(q + s/2) \Psi_t(q - s/2) \equiv \int dR_0 \int dP_0 e^{-\frac{i}{\hbar} P_t s} \delta(R_t - q) \rho^W_0(P_0, R_0). \tag{241}
\]

Substituting Eq. (241) into Eq. (238) we obtain:

\[
\frac{\partial \rho^W_t(p, q)}{\partial t} = \frac{1}{2 \pi \hbar} \int_{-\infty}^{\infty} ds \int dR_0 \int dP_0 e^{\frac{i}{\hbar} (p - P_t)s} \left\{ \frac{i}{\hbar} \left[ V(q + s/2) - V(q - s/2) \right] \delta(R_t - q) - \frac{p}{m} \frac{\partial \delta(R_t - q)}{\partial q} \right\} \rho^W_0(P_0, R_0). \tag{242}
\]

Furthermore, substituting Eq. (241) into Eq. (231) gives:

\[
\rho^W_t(p, q) = \frac{1}{2 \pi \hbar} \int_{-\infty}^{\infty} ds \int dR_0 \int dP_0 e^{\frac{i}{\hbar} (p - P_t)s} \delta(R_t - q) \rho^W_0(P_0, R_0), \tag{243}
\]

and computing the time-derivative of both sides of Eq. (243) we obtain:

\[
\frac{\partial \rho^W_t(p, q)}{\partial t} = \frac{1}{2 \pi \hbar} \int_{-\infty}^{\infty} ds \int dR_0 \int dP_0 e^{\frac{i}{\hbar} (p - P_t)s} \left[ \frac{i}{\hbar} \frac{\partial P_t}{\partial t} s \delta(R_t - q) + \frac{\partial \delta(R_t - q)}{\partial t} \right] \rho^W_0(P_0, R_0),
\]

\[
= \frac{1}{2 \pi \hbar} \int_{-\infty}^{\infty} ds \int dR_0 \int dP_0 e^{\frac{i}{\hbar} (p - P_t)s} \left[ \frac{i}{\hbar} \frac{\partial P_t}{\partial t} s \delta(R_t - q) + \frac{\partial \delta(R_t - q)}{\partial R_t} \frac{\partial R_t}{\partial t} \right] \rho^W_0(P_0, R_0),
\]

\[
= \frac{1}{2 \pi \hbar} \int_{-\infty}^{\infty} ds \int dR_0 \int dP_0 e^{\frac{i}{\hbar} (p - P_t)s} \left[ \frac{i}{\hbar} \frac{\partial P_t}{\partial t} s \delta(R_t - q) - \frac{\partial \delta(R_t - q)}{\partial q} \frac{\partial R_t}{\partial t} \right] \rho^W_0(P_0, R_0). \tag{244}
\]
Comparing the third line of Eq. (244) with Eq. (242) we obtain the equations of motion of \( \{ R(t), P(t) \} \):

\[
\begin{align*}
\frac{\partial P_t}{\partial t} & = -\frac{(V(q + s/2) - V(q - s/2))}{s}, \\
\frac{\partial R_t}{\partial t} & = \frac{p}{m}.
\end{align*}
\]  

(245)

In the classical limit, \( \hbar \to 0 \), Eqs. (245) become Hamilton’s equations,

\[
\begin{align*}
\frac{\partial P_t}{\partial t} & = -\frac{\partial V(q)}{\partial q}, \\
\frac{\partial R_t}{\partial t} & = \frac{P_t}{m}.
\end{align*}
\]  

(246)

since the most significant contributions to \( \rho^W_t(p, q) \), as defined by Eq. (241), 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

\[
\rho^W_t(p, q) = \int dR_0 \int dP_0 \delta(p - P_t)\delta(R_t - q)\rho^W_0(P_0, R_0),
\]

(247)

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

25 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

\[
i\hbar \frac{\partial \Psi_t(q)}{\partial t} = \hat{H} \Psi_t(q),
\]

\[
= -\frac{\hbar^2}{2m} \frac{\partial^2 \Psi_t(q)}{\partial q^2} + V(q) \Psi_t(q),
\]

(248)

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. (248).
25.1 Hamilton Jacobi Equation

We consider the Hamiltonian

\[ H(q, p) = \frac{p^2}{2m} + V(q), \]  

(249)

and we define a canonical transformation in terms of the generating function \( S = S(q, P, t) \) as follows:

\[ p = \frac{S(q, P, t)}{\partial q}, \]
\[ Q = \frac{S(q, P, t)}{\partial P}, \]  

(250)

with

\[ \frac{\partial S(q, P, t)}{\partial t} + H(q, \frac{\partial S(q, P, t)}{\partial q}) = 0, \]  

(251)

and

\[ \tilde{H}(Q, P, t) \equiv \frac{\partial S(q, P, t)}{\partial t} + H(q, \frac{\partial S(q, P, t)}{\partial q}), \]  

(252)

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. (251) and (252), \( \tilde{H}(Q, P, t) = 0 \). Therefore,

\[ \dot{Q} = \frac{\partial \tilde{H}(Q, P, t)}{\partial P} = 0, \]
\[ \dot{P} = -\frac{\partial \tilde{H}(Q, P, t)}{\partial Q} = 0. \]  

(253)

Equation (251) 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. (251) for \( S(q, P, t) \). Then, compute \( p(t) = \partial S/\partial q \) by partial differentiation as defined in Eq. (250). 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. (253)).

25.2 Quantum Dynamics: Motion of Hidden Variables

To solve the time-dependent Schrödinger equation, introduced by Eq. (248), we write \( \Psi_t(x) \) in terms of the amplitude \( A_t(x) \) and phase \( S_t(x) \) functions, as follows:

\[ \Psi_t(q) = A_t(q)e^{iS_t(q)}, \]  

(254)
where $A_t(q)$ and $S_t(q)$ are defined as real functions. Substituting Eq. (254) into Eq. (248), we obtain:

$$i\hbar \frac{\partial \Psi_t(q)}{\partial t} = \left[ i\hbar \frac{\partial A_t(q)}{\partial t} - A_t(q) \frac{\partial S_t(q)}{\partial t} \right] e^{iS_t(q)} ,$$

$$\hat{H}_t(q) = \left[ -\frac{\hbar^2}{2m} \frac{\partial^2 A_t(q)}{\partial q^2} - \frac{i\hbar}{m} \frac{\partial A_t(q)}{\partial q} - \frac{1}{2m} A_t(q) \left( \frac{\partial S_t(q)}{\partial q} \right)^2 + A_t(q)V(q) \right] e^{iS_t(q)} .$$

(255)

Since the first line of Eq. (255) must be equal to the second line, the real parts of the left hand sides of Eq. (255) must be equal:

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

(256)

where

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

(257)

is a time-dependent “quantum” potential determined by $A_t$. Note that Eq. (256) is the Hamilton-Jacobi equation for a system described by the Hamiltonian:

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

(258)

with $p = \partial S/\partial q$, and $V_Q(q, t)$ the time-dependent “external” field potential defined by Eq. (257).

Furthermore, since the imaginary parts of the left hand sides of Eq. (255) must be equal, we obtain:

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

(259)

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. (259), gives:

$$\frac{1}{2} \rho_t^{-1/2} \frac{\partial \rho_t}{\partial t} + \frac{1}{2} \rho_t^{-1/2} \rho_t \frac{\partial S_t(q)}{\partial q} \frac{1}{m} + \rho_t^{1/2} \frac{\partial^2 S_t(q)}{\partial q^2} \frac{1}{2m} = 0 ,$$

$$\frac{\partial \rho_t}{\partial t} + \rho_t \frac{\partial S_t(q)}{\partial q} \frac{1}{m} + \rho_t \frac{\partial^2 S_t(q)}{\partial q^2} \frac{1}{m} = 0 ,$$

(260)

$$\frac{\partial \rho_t}{\partial t} + \frac{\partial j_t}{\partial q} = 0$$

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

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

\[
\begin{align*}
\dot{q}(t) &= \frac{\partial H(q, p)}{\partial p}, \\
\dot{p}(t) &= -\frac{\partial H(q, p)}{\partial q},
\end{align*}
\]

(261)

with \( H(q, p) \) defined according to Eq. (258). 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.

25.4 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) gedanken-experiment [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 particle 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].
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

\[
H(q, p) = \frac{p^2}{2m} + V(q)
\]

(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 interpretation 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 \( \hat{H} = \frac{\hat{p}^2}{2m} + V(\hat{x}) \) by using the Wigner transform instead of the wave function.

(15 points) 2.2: Explain how to compute the photoelectron detachment spectrum of \( I_2^- \).

26.1 Answer Key

Exercise 1:
(20 points) 1.1:
Solving the Hamilton-Jacobi equation

\[
\frac{\partial S(q, P, t)}{\partial t} + H(q, \frac{\partial S(q, P, t)}{\partial q}) = 0,
\]

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 \( q(t) \),

\[
p(t) = \frac{\partial S(q, P, t)}{\partial q},
\]

and to obtain \( q(t) \) we first compute \( Q \) as the partial derivative of \( S(q, P, t) \) with respect to \( P \),

\[
Q = \frac{\partial S(q, P, t)}{\partial P},
\]
and then we solve for \( q(t) \) as a function of \( Q \) and \( P \). Both \( Q \) and \( P \) are constant in time since

\[
\tilde{H}(Q, P) = \frac{\partial S(q, P, t)}{\partial t} + H(q, \frac{\partial S(q, P, t)}{\partial q}) = 0,
\]

and

\[
\dot{Q} = \frac{\partial \tilde{H}(Q, P)}{\partial P}, \\
\dot{P} = -\frac{\partial \tilde{H}(Q, P)}{\partial Q}.
\]

The resulting equations of motion for \( p(t) \) and \( q(t) \) are Hamilton’s equations:

\[
\dot{q}(t) = \frac{\partial H(q, p)}{\partial p}, \\
\dot{p}(t) = -\frac{\partial H(q, p)}{\partial q}.
\]

(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,

\[
i\hbar \frac{\partial \Psi(q)}{\partial t} = \hat{H} \Psi(q),
\]

by substituting \( \Psi(q) \), in terms of the real amplitude \( A_t(q) \) and phase \( S_t(q) \) functions,

\[
\Psi_t(q) = A_t(q) e^{\frac{i}{\hbar} S_t(q)},
\]

The left hand side of Eq. (268) gives

\[
i\hbar \frac{\partial \Psi_t(q)}{\partial t} = \left[ i\hbar \frac{\partial A_t(q)}{\partial t} - A_t(q) \frac{\partial S_t(q)}{\partial t} \right] e^{\frac{i}{\hbar} S_t(q)},
\]

and the right hand side of Eq. (268) gives

\[
\hat{H} \Psi_t(q) = \left[ -\frac{\hbar^2}{2m} \frac{\partial^2 A_t}{\partial q^2} - i\hbar \frac{\partial A_t}{\partial q} \frac{\partial S_t}{\partial q} + \frac{1}{2m} A_t(q) \left( \frac{\partial S_t}{\partial q} \right)^2 + A_t(q) V(q) \right] e^{\frac{i}{\hbar} S_t(q)}.
\]

Considering that the real parts of the r.h.s’ of Eqs. (270) and (271) must be equal, we obtain:

\[
\frac{\partial S_t(q)}{\partial t} + \frac{1}{2m} \left( \frac{\partial S_t}{\partial q} \right)^2 + V(q) + V_Q(q, t) = 0,
\]

where

\[
V_Q(q, t) = -\frac{\hbar^2}{2m} \frac{\partial^2 A_t}{\partial q^2} \frac{1}{A_t(q)}.
\]
As shown in (1.2), Eq. (272) is the Hamilton-Jacobi equation for the time-dependent Hamiltonian

\[ H(q, p) = \frac{p^2}{2m} + V(q) + V_Q(q, t), \]  

(274)

with \( p = \frac{\partial S}{\partial q} \) and \( V_Q(q, t) \) the time-dependent external field potential defined by Eq. (273). The equations of motion of the “hidden” variables \( q(t) \) and \( p(t) \) are Hamilton’s equations (see Eq. (267)) with the Hamiltonian defined according to Eq. (274).

(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 auxiliary 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 position and momentum as simultaneously existing quantities. In contrast, the Bohmian formulation assumes that the properties of the system are intrinsic 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 existence (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 $\rho^W_t(p, q)$ is defined as follows:

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

(275)

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:

$$\langle \Psi_t | \hat{H} | \Psi_t \rangle = \int dq \int dp H(q, p) \rho^W_t(p, q),$$

(276)
since

$$\int dq \int dp H(q, p) \rho^W_t(p, q) = \frac{1}{2\pi\hbar} \int dq \int dp \frac{p^2}{2m} \int dse^{i\hbar p s} \Psi_t^*(q + s/2) \Psi_t(q - s/2)
+ \frac{1}{2\pi\hbar} \int dp \int dse^{i\hbar p s} \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^{i\hbar p(x' - x'')} \Psi_t^* (x') \Psi_t(x'')
+ \int ds \delta(s) \int dq V(q) \Psi_t^*(q + s/2) \Psi_t(q - s/2),
= \int dp \frac{p^2}{2m} \tilde{\Psi}_t^*(p) \tilde{\Psi}_t(p) + \int dq V(q) \Psi_t^*(q) \Psi_t(q),
= \int dp \langle \Psi_t | \frac{p^2}{2m} | \Psi_t \rangle + \langle \Psi_t | V(q) | \Psi_t \rangle.
= \langle \Psi_t | \frac{p^2}{2m} | \Psi_t \rangle + \langle \Psi_t | V(q) | \Psi_t \rangle.

(277)

(15 points) 2.2:
The photoabsorption lineshape (at 0 K) is,

$$I_0(\omega) = \frac{3}{2\pi\hbar} \int_{-\infty}^\infty dt e^{i\omega t} \langle \Phi_0 | (\lambda \bar{\varepsilon}_0^\dagger \cdot \hat{\mu}(t))(\lambda \bar{\varepsilon}_0^\dagger \cdot \hat{\mu}(0)) | \Phi_0 \rangle,$$

$$= \frac{3}{2\pi\hbar} \sum_f \int_{-\infty}^\infty dt e^{i\omega t} \langle \Phi_0 | e^{i\hat{H}t}(\lambda \bar{\varepsilon}_0^\dagger \cdot \hat{\mu}) e^{-i\hat{H}t} | \Phi_f \rangle \langle \Phi_f | (\lambda \bar{\varepsilon}_0^\dagger \cdot \hat{\mu}) | \Phi_0 \rangle,$$

(278)

where the final states are of the form

$$| \Phi_f \rangle = | \psi_K \rangle | \chi_{E_K} \rangle,$$

(279)

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,

\[ P_0(\epsilon) = \frac{3}{2\pi\hbar} \sum_K \int dE_K \int_{-\infty}^{\infty} dt e^{i\omega t} \langle \Phi_0 | e^{\frac{i}{\hbar} \hat{H}t} (\lambda \vec{\epsilon}_0 \cdot \hat{\mu}) e^{-\frac{i}{\hbar} (E_K + \epsilon) t} | \Phi_K \rangle \langle \Phi_K | (\lambda \vec{\epsilon}_0 \cdot \hat{\mu}) | \Phi_0 \rangle, \quad (280) \]

and using the relation

\[ \int dE_K |\chi_{E_K}\rangle \langle \chi_{E_K}| e^{-\frac{i}{\hbar} E_K t} = e^{-\frac{i}{\hbar} H_K t}, \quad (281) \]

the finite temperature distribution is

\[ P_\beta(\epsilon) = \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} dt e^{-\frac{i}{\hbar} \omega t} C(t), \quad (282) \]

with

\[ C(t) = 3 \sum_j \rho_j e^{\frac{i}{\hbar} (E_j + \omega t)} \sum_K \langle \Phi_j | (\lambda \vec{\epsilon}_0 \cdot \hat{\mu}) e^{\frac{i}{\hbar} \hat{H}K t} (\lambda \vec{\epsilon}_0 \cdot \hat{\mu}) | \Phi_j \rangle. \quad (283) \]

Therefore, according to Eqs. (283) and (282), 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.
Semiclassical Dynamics

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

$$G(x) = i\hbar \frac{\partial \psi}{\partial t} + \frac{\hbar^2}{2m} \frac{\partial^2 \psi}{\partial x^2} - V(x,t)\psi = 0. \quad (284)$$


We propose the Gaussian-beam ansatz

$$\psi(x,t) = e^{iS(t)/\hbar} \varphi(x,t), \quad (285)$$

with

$$\varphi(x,t) = \pi^{-1/4} \hbar^{-1/4} Q^{-1/2} e^{-\gamma(x-q)^2/(2\hbar)+i\hbar p(x-q)}, \quad (286)$$

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

A Taylor expansion gives,

$$G(x) = G(q) + G'(q)(x-q) + \frac{1}{2}G''(q)(x-q)^2 + ... \quad (287)$$

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

$$i\hbar \frac{\partial \psi}{\partial t} = \left(-\dot{S} - \frac{i\hbar \dot{Q}}{2} - i \left(\dot{P}Q^{-1} - PQ^{-2}\dot{Q}\right) (x-q)^2/2 + iPQ^{-1}(x-q)\dot{q} - \dot{p}(x-q) + p\dot{q}\right)\psi, \quad (288)$$

and

$$\frac{\hbar^2}{2m} \frac{\partial^2 \psi}{\partial x^2} = \frac{\hbar^2}{2m} \left([\frac{-PQ^{-1}(x-q)}{\hbar} + i\frac{p}{\hbar}]^2 - \frac{PQ^{-1}}{\hbar}\right)\psi, \quad (289)$$

we obtain

$$G(x) = \left(-\dot{S} - \frac{i\hbar \dot{Q}}{2} - i \left(\dot{P}Q^{-1} - PQ^{-2}\dot{Q}\right) (x-q)^2/2 + iPQ^{-1}(x-q)\dot{q} - \dot{p}(x-q) + p\dot{q}\right.$$

$$+ \frac{\hbar^2}{2m} \left[-PQ^{-1}(x-q)/\hbar + i\frac{p}{\hbar}\right]^2 - \frac{\hbar}{2m} PQ^{-1} - V(x)\right)\psi. \quad (290)$$
with,
\[
G'(x) = G(x) \frac{\psi'}{\psi} + \left( -i \left( \hat{P} Q^{-1} - P Q^{-2} \right) (x - q) + i P Q^{-1} \dot{q} - \dot{p} \right) \\
- \frac{\hbar}{m} \left[ -P Q^{-1} (x - q) + \frac{i}{\hbar} \right] P Q^{-1} - V'(x) \psi.
\] (291)

and
\[
G''(x) = G(x) \frac{\psi''}{\psi} + G'(x) \frac{\psi'}{\psi} - G(x) \frac{\psi'^2}{\psi^2} \\
+ \left[ G'(x) \frac{1}{\psi} - G(x) \frac{\psi'}{\psi^2} \right] \psi' \\
+ \left( -i \left( \hat{P} Q^{-1} - P Q^{-2} \right) \right) \\
+ \frac{1}{m} \left[ P Q^{-1} \right]^2 - V''(x) \psi.
\] (292)

Therefore, making \( G(q) = G'(q) = 0 \), we obtain:
\[
G'(q) = \left( i P Q^{-1} \dot{q} - \dot{p} - \frac{P}{m} P Q^{-1} - V'(q) \right) \psi(q) = 0.
\] (293)

This equation must be satisfied even when \( \gamma = P Q^{-1} \) is real. Therefore, since the real and imaginary parts of the bracket must be zero,
\[
\dot{q} = \frac{P}{m}, \\
\dot{p} = -V'(q).
\] (294)

In addition,
\[
G(q) = \left( -\dot{S} - \frac{i \hbar}{2} \dot{Q} - P Q^{-1} \frac{\hbar}{2m} - V(q) - \frac{P^2}{2m} + p \dot{q} \right) \psi(q) = 0,
\] (295)

which must hold true even when \( \gamma = P Q^{-1} \) is imaginary. Therefore,
\[
\dot{S} = p \dot{q} - \left( V(q) + \frac{P^2}{2m} \right), \\
\dot{Q} = i \frac{P}{m}.
\] (296)

Finally,
\[
G''(q) = \left( -i \left( \hat{P} Q^{-1} - P Q^{-2} \right) + \frac{1}{m} \left[ P Q^{-1} \right]^2 - V''(q) \right) \psi(q) = 0,
\] (297)

which is verified when
\[
\hat{P} = i V''(q) Q.
\] (298)
Semiclassical Dynamics in the Gaussian-Hermite Basis

Consider the time-dependent Schrödinger equation,

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

(299)

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. (299) can be written, as follows:

\[ \psi_0(x,t) = e^{iS(t)/\hbar} \varphi_0(x,t), \]  

(300)

with \( \varphi_0 \) the first element of the Gaussian-Hermite basis-set,

\[ \varphi_\nu(x,t) = H_\nu(\hbar^{-1/2}|Q|^{-1}(x - q))A(Q,\nu)e^{-\gamma(x-q)^2/(2\hbar) + \hat{p}(x-q)}, \]  

(301)

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

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

(302)

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

\[ \varphi_0(x,t) = \pi^{-1/4}\hbar^{-1/4}Q^{-1/2}e^{-\gamma(x-q)^2/(2\hbar) + \hat{p}(x-q)}. \]  

(303)

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

\[ \dot{S} = \frac{p^2}{2m} - V(q,t), \]

\[ \dot{q} = \frac{p}{m}, \]

\[ \dot{p} = -\frac{\partial V}{\partial q}. \]  

(304)

Finally, the equations of motion for \( Q \) and \( P \) are obtained by substituting \( \psi_0(x,t) \) into Eq. (299):

\[ \dot{Q} = i\frac{P}{m}, \]

\[ \dot{P} = i\frac{\partial^2 V}{\partial q^2}Q, \]  

(305)

or,

\[ \ddot{Q} = -\frac{1}{m}\frac{\partial^2 V}{\partial q^2}Q \]

\[ = -\frac{k(t)}{m}Q, \]

\[ = -\omega(t)^2Q, \]  

(306)
with $Q$ and $P$ defined as linear combinations of the partial derivatives of $q(t)$ and $p(t)$ with respect to their initial values, as follows:

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

(307)

since $Q$ and $P$, defined according to Eqs. (307), satisfy Eqs. (305). According to Eqs. (307), $Q$ and $P$ satisfy the following relation:

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

(308)

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

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

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

Therefore, $\gamma(t) = |Q(t)|^{-2} + i\Im[P(t)/Q(t)]$. Note that, according Eqs. (305), $Q(t)$ and $P(t)$ remain real and purely imaginary, respectively, when choosing initial conditions with $Q(0) = \Re[Q(0)]$ real and $P(0) = i\Im[P(0)]$. With such initial conditions, $\gamma(t) = Q(t)^{-2} + P(t)/Q(t)$.

In particular, when choosing $P(0) = i/Q(0)$, with real $Q(0)$, $\Re[\gamma(0)] = \Im[\gamma(0)] = Q(0)^{-2}$, or $\gamma(0) = (1 + i)/Q(0)^2$.

Other possible solutions can be obtained by defining the raising and lowering ladder operators $\hat{L}_+$ and $\hat{L}_-$, respectively, as follows [Hagedorn, G. A.: Raising and lowering operators for semiclassical wave packets. *Ann. Phys.* 269, 77-104 (1998)]:

$$
\hat{L}_+ = (2\hbar)^{-1/2}(\bar{P}(x - q) - i\bar{Q}(\hat{p} - p)),
$$
$$
\hat{L}_- = (2\hbar)^{-1/2}(\bar{P}(x - q) + i\bar{Q}(\hat{p} - p)),
$$

(309)

and noting that

$$
[\hat{L}_-, \hat{L}_+] = 1,
$$
$$
\frac{1}{2}(\hat{L}_- \hat{L}_+ + \hat{L}_+ \hat{L}_-) \psi_\nu = \left( \nu + \frac{1}{2} \right) \psi_\nu,
$$

(310)

we see that

$$
\hat{L}_+ \psi_\nu = \sqrt{\nu + 1} \psi_{\nu + 1}.
$$

(311)

Therefore,

$$
\psi_\nu(x, t) = \hat{L}_+^\nu \psi_0(x; t) = \sqrt{\nu!} e^{iS(t)/\hbar} \varphi_\nu(x, t),
$$

(312)

with $\nu = 0, 1, 2, \ldots$ are also solutions of Eq. (299).
28.1 Multidimensional Semiclassical Dynamics

The 2-dimensional generalization of Eq. (303) is:

$$\varphi_0(x, t) = \pi^{-n/4}h^{-n/4}[^{\text{det}}(Q)]^{1/2}e^{-\left((x-q) \cdot \frac{1}{2}PQ^{-1} - (x-q)/(2h) + \frac{i}{2}P(x-q)\right)}$$

with \(n = 2\). Here, \(P\) and \(Q\) are \(n \times n\) matrices that evolve in time, as follows:

\[
\dot{Q} = \frac{i}{m}P, \quad \dot{P} = ikQ,
\]

or,

\[
\ddot{Q} = -\frac{1}{m}kQ,
\]

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

To calculate the \(\text{det}(Q)\) for Eq. (313), we use the log-derivative substitution:

\[
R = \frac{\partial}{\partial t} \log[Q],
\]

giving,

\[
\dot{Q} = RQ.
\]

Integrating Eq. (317), we obtain:

\[
Q = \exp \left[ \int_0^t dt' R(t') \right],
\]

Therefore,

\[
\text{det}(Q) = \prod_{k=1}^n e^{\Delta tR_k},
\]

The equation of motion for \(R\) is obtained from Eq. (316), as follows:

\[
\ddot{R} = \dot{Q}Q^{-1} - \left(\dot{Q}Q^{-1}\right)^2.
\]

Substituting Eqs. (316) and (306) into Eq. (320), we obtain:

\[
\ddot{R} = -\omega(t)^2 - R^2.
\]
28.2 Log Derivative Propagation

For constant $\omega$, the propagation of $R = \dot{Q}Q^{-1}$ can be based on Eq. (315):

$$\ddot{Q} = -\frac{k}{m}Q,$$  (322)

with

$$Q(t) = Q(0)\cos\left(\sqrt{\frac{k}{m}}t\right) + \frac{\dot{Q}(0)}{\sqrt{\frac{k}{m}}}\sin\left(\sqrt{\frac{k}{m}}t\right),$$  (323)

$$\dot{Q}(t) = \dot{Q}(0)\cos\left(\sqrt{\frac{k}{m}}t\right) - Q(0)\sqrt{\frac{k}{m}}\sin\left(\sqrt{\frac{k}{m}}t\right).$$

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

$$R(t) = \left(R(0)\cos[\omega t] - \omega\sin[\omega t]\right) \left[\cos[\omega t] + R(0)\sin[\omega t]/\omega \right]^{-1}. \quad (324)$$

When $\omega$ changes slowly with time,

$$R(t + \delta) = \left(R(t)\cos[\omega(t)\delta] - \omega(t)\sin[\omega(t)\delta]\right) \left[\cos[\omega(t)\delta] + R(t)\sin[\omega(t)\delta]/\omega(t) \right]^{-1}. \quad (325)$$

When the instantaneous normal modes (i.e., eigenvectors of $K$) are approximately constant, we can solve Eq. (322) by transforming it into a problem of $n$ 1-dimensional equations, as follows [[J. Chem. Phys. (1999) 110:9922-9936]]:

$$\dot{\tilde{Q}}(j,j) = i\frac{\tilde{P}(j,j)}{m},$$  (326)

$$\dot{\tilde{P}}(j,j) = i\tilde{k}(j)\tilde{Q}(j,j),$$

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

$$L^\dagger \cdot K(t) \cdot L = m\tilde{\omega}(t)^2. \quad (327)$$

where $\tilde{\omega}$ is a diagonal matrix. The new variables $\tilde{Q}$ and $\tilde{P}$, introduced by Eq. (326), are defined according to the analogous transformations,

$$\tilde{Q}(t) = L^\dagger \cdot Q(t) \cdot L,$$

$$\tilde{P}(t) = L^\dagger \cdot P(t) \cdot L. \quad (328)$$

Assuming that $L$ are approximately constant, we obtain Eq. (326) by computing the time-derivative of Eqs. (328) and substituting $\tilde{Q}$ and $\tilde{P}$ according to Eqs. (314). Note that $\det(Q) = \det(\tilde{Q})$ since the $\det(L) = \det(L^\dagger) = 1$. 

81
An approximate solution of Eqs. (326) could also be obtained by solving them according to the WKB approximation, as follows. Combining the Eqs. (326), we obtain:

\[ \ddot{Q}(j,j) = -\tilde{\omega}(j)^2 \dot{Q}(j,j), \]  

The possible WKB solutions,

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

satisfy Eq. (329), as follows:

\[ \ddot{\tilde{Q}}_a(j,j; t) = -\tilde{\omega}(j; t)^2 \tilde{Q}_a(j,j; t) + \Delta, \]

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

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

\[ \begin{align*}
\ddot{\tilde{Q}}_a(j,j; t) &= \dot{\tilde{Q}}_a(j,j; t) \cos \left( \int_0^t \tilde{\omega}(j; t') dt' \right) + \tilde{Q}_a(j,j; 0) \sin \left( \int_0^t \tilde{\omega}(j; t') dt' \right), \\
\dot{\tilde{Q}}_a(j,j; t) &= \dot{\tilde{Q}}_a(j,j; 0) \cos \left( \int_0^t \tilde{\omega}(j; t') dt' \right) - \tilde{Q}_a(j,j; 0) \tilde{\omega}(j; t') \sin \left( \int_0^t \tilde{\omega}(j; t') dt' \right).
\end{align*} \]

### 28.3 Normalization of Multidimensional Gaussians

The goal of this section is to show that the multidimensional Gaussian, introduced by Eq. (313), is normalized as follows:

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

\[ \begin{align*}
&= \int dx_1 dx_2 e^{-(x_1,x_2)c_{11}c_{12}} \begin{pmatrix} x_1 \\ c_{21}c_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \\
&= \int dx e^{-x^T \cdot e - x}, \\
&= \sqrt{\frac{\pi^2}{det(e)}},
\end{align*} \]

First, we introduce the orthogonal transformation \( x = \Gamma \xi \),

\[ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \Gamma_{11} & \Gamma_{12} \\ \Gamma_{21} & \Gamma_{22} \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix}, \]

82
where the columns of the matrix $\Gamma$ are the eigenvectors of $c$, so that $\Gamma^{-1}c\Gamma = \lambda$, or

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

(335)

Substituting Eq. (334) into Eq. (333), we obtain:

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

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

(336)

$$
= \sqrt{\frac{\pi^2}{\lambda_1 \lambda_2}} = \sqrt{\frac{\pi^2}{\det(\lambda)}}.
$$

Furthermore, according to Eq. (335),

$$
det(\lambda) = det(\Gamma)^{-1}det(c)det(\Gamma)
$$

$$
= det(c),
$$

(337)

Therefore, substituting Eq. (337) into Eq. (336), we obtain Eq. (333).

Note: The Jacobian of the orthogonal transformation, introduced by Eq. (334), is equal to 1 since

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

$$
= 1.
$$

(338)

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

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

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

(339)

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

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

(340)

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


29 Second Quantization

The goal of this section is to introduce the single-particle basis \( \{ \psi_{\nu_1}(r), \psi_{\nu_2}(r), \psi_{\nu_3}(r), \ldots \} \) for representation of the N-particle state \( \Psi(r_1, r_2, \ldots, r_N) \) in terms of symmetrized product states \( \hat{S}_\pm \prod_{j=1}^N \psi_{\nu_j}(r_j) \), and its correspondence to the occupation number representation \( |n_{\nu_1}, n_{\nu_2}, n_{\nu_3}, \ldots \rangle \), where \( n_{\nu_j} \) is the number of particles in state \( \psi_{\nu_j}(r) \) in the product state representation. Furthermore, we introduce the creation \( \hat{a}_j^{\dagger} \) and annihilation \( \hat{a}_j \) operators (i.e., operators that raise or lower the occupation numbers \( n_{\nu_j} \) 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:

\[
\hat{A} = \sum_{\nu_j, \nu_k} A_{\nu_j, \nu_k} \hat{a}_j^{\dagger} \hat{a}_k, \quad \text{with} \quad A_{\nu_j, \nu_k} = \langle \nu_j | \hat{A} | \nu_k \rangle.
\]

29.1 Single-Particle Basis

The state of the N-particle system \( \Psi(r_1, r_2, \ldots, r_N) \) can be represented in a complete orthonormal basis composed of single-particle states \( \{ \psi_{\nu_j}(r) \} \), satisfying that

\[
\sum_{\nu_j} \psi_{\nu_j}(r')^* \psi_{\nu_j}(r) = \delta(r' - r), \quad (341)
\]

and

\[
\int d\mathbf{r} \psi_{\nu_j}(\mathbf{r})^* \psi_{\nu_k}(\mathbf{r}) = \delta_{\nu_j, \nu_k}. \quad (342)
\]

To represent \( \Psi(r_1, r_2, \ldots, r_N) \), we first project the state along the basis set of \( r_1 \), as follows:

\[
\Psi(r_1, r_2', \ldots, r_N') = \sum_{\nu_1} \psi_{\nu_1}(r_1) \int d\mathbf{r}'_1 \psi_{\nu_1}(\mathbf{r}'_1)^* \Psi(\mathbf{r}'_1, r_2', \ldots, r_N'), \quad (343)
\]

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

\[
\Psi(r_1, r_2, \ldots, r_N) = \sum_{\nu_1, \ldots, \nu_N} c_{\nu_1, \ldots, \nu_N} \prod_{j=1}^N \psi_{\nu_j}(r_j), \quad (344)
\]

with

\[
c_{\nu_1, \ldots, \nu_N} = \int d\mathbf{r}'_1 \psi_{\nu_1}(\mathbf{r}'_1)^* \cdots \int d\mathbf{r}'_N \psi_{\nu_N}(\mathbf{r}'_N)^* \Psi(\mathbf{r}'_1, r_2, \ldots, r_N'). \quad (345)
\]

While the product states \( \prod_{j=1}^N \psi_{\nu_j}(r_j) \) 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
set of $N$ states. The occupation number representation

\[ \psi_{\nu_1}(r_1) \psi_{\nu_2}(r_2) \cdots \psi_{\nu_N}(r_N) \]

which are linear combinations of product states corresponding to all possible permutation on the set of $N$ coordinates. Each term of the Slater determinant has a sign $(-1)^p$, corresponding to the number of permutations $p$, while the bosonic permanent terms are all sing-less.

29.2 Occupation Number Basis

The product states, introduced by Eq. (346), 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 $n_{\nu_j}$ in each occupied state $\nu_j$, with $\sum_j n_{\nu_j} = N$. Such states are eigenstates of the number operators,

\[ \hat{n}_{\nu_k} |n_{\nu_1}, n_{\nu_2}, n_{\nu_3}, \cdots \rangle = n_{\nu_k} |n_{\nu_1}, n_{\nu_2}, n_{\nu_3}, \cdots \rangle. \]

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

29.3 Creation and Anihilation Operators

**Bosons:** The creation and anihilation operators of bosons, $\hat{b}^\dagger_j$ 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 $\nu_j$ as follows:

\[ \hat{n}_{\nu_j} |n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots \rangle = n_{\nu_j} |n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots \rangle, \]

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

\[ \hat{b}_j^\dagger |n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots \rangle = B_+(n_{\nu_j}) |n_{\nu_1}, n_{\nu_2}, \cdots (n_{\nu_j} + 1), \cdots \rangle, \]

\[ \hat{b}_j |n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots \rangle = B_-(n_{\nu_j}) |n_{\nu_1}, n_{\nu_2}, \cdots (n_{\nu_j} - 1), \cdots \rangle, \]

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 $\hat{b}_j |n_{\nu_1}, \nu_2, \cdots 0, \cdots \rangle = 0$. Furthermore, we define the normalization constants $B_+(0) = 1$ and $B_-(1) = 1$ so that

\[ \hat{b}_j^\dagger |n_{\nu_1}, n_{\nu_2}, \cdots 0, \cdots \rangle = |n_{\nu_1}, n_{\nu_2}, \cdots 1, \cdots \rangle, \]

\[ \hat{b}_j |n_{\nu_1}, n_{\nu_2}, \cdots 1, \cdots \rangle = |n_{\nu_1}, n_{\nu_2}, \cdots 0, \cdots \rangle. \]

Therefore,

\[ \hat{b}_j^\dagger \hat{b}_j |n_{\nu_1}, n_{\nu_2}, \cdots 0, \cdots \rangle = |n_{\nu_1}, n_{\nu_2}, \cdots 0, \cdots \rangle, \]

\[ \hat{b}_j^\dagger \hat{b}_j |n_{\nu_1}, n_{\nu_2}, \cdots 0, \cdots \rangle = 0, \]

85
which can be summarized as \( \hat{b}_j \hat{b}_j^\dagger = \hat{n}_j + 1 \) and \( [\hat{b}_j, \hat{b}_k^\dagger] = 1 \). When \( j \neq k \), however, \( [\hat{b}_j, \hat{b}_k^\dagger] = 0 \). The normalization constants for other states are found from Eq. (348), as follows:

\[
\langle n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots | \hat{b}_j \hat{b}_j^\dagger | n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots \rangle = n_{\nu_j},
\]

\[
\langle n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots | \hat{b}_j \hat{b}_j^\dagger | n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots \rangle = B_+(n_{\nu_j})^2,
\]

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

\[
\langle n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots | \hat{b}_j \hat{b}_j^\dagger | n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu_j}, \cdots \rangle = B_+(n_{\nu_j})^2,
\]

\[
(n_{\nu_j} + 1) = B_+(n_{\nu_j})^2,
\]

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

\[
\langle \hat{b}_j^{\dagger n_{\nu_j}} | n_{\nu_1}, n_{\nu_2}, \cdots 0, \cdots \rangle = \sqrt{n_{\nu_j}!} | n_{\nu_1}, n_{\nu_2}, \cdots n_{\nu}, \cdots \rangle.
\]

or

\[
| n_{\nu_1}, n_{\nu_2}, n_{\nu_3}, \cdots \rangle = \prod_j \frac{\hat{b}_j^{\dagger n_{\nu_j}}}{\sqrt{n_{\nu_j}!}} | 0, 0, 0, \cdots \rangle.
\]

**Fermions**: The creation and annihilation operators of fermions, \( \hat{c}_j^\dagger \) and \( \hat{c}_j \), are defined to ensure that the number operator \( \hat{n}_j = \hat{c}_j^\dagger \hat{c}_j \) gives the number of fermions \( n_{\nu_j} = 0, 1 \) in state \( \nu_j \). This requires that \( \hat{c}_j^\dagger |1\rangle = 0, \hat{c}_j^\dagger |0\rangle = |1\rangle, \hat{c}_j |0\rangle = 0, \) and \( \hat{c}_j^\dagger |0\rangle = |1\rangle \). Therefore, \( \hat{c}_j \hat{c}_j^\dagger |0\rangle = |0\rangle \) and \( \hat{c}_j \hat{c}_j^\dagger |0\rangle = |0\rangle \), or \( \hat{c}_j \hat{c}_j^\dagger + \hat{c}_j^\dagger \hat{c}_j = 0 \).

### 29.4 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 \( \hat{T}_k = \frac{p_k^2}{2m_k} \):

\[
\langle \psi_{\nu_1}, \psi_{\nu_2} \cdots \psi_{\nu_N} | \psi_{\nu_1}, \psi_{\nu_2} \cdots \psi_{\nu_N} \rangle = \sum_{\nu_j} \langle \psi_{\nu_j} | \hat{T} | \psi_{\nu_1}, \psi_{\nu_2} \cdots \psi_{\nu_N} \rangle
\]

\[
= \sum_{\nu_j} \langle \psi_{\nu_j} | \sum_{k=1}^{N} \hat{T}_k | \psi_{\nu_1}, \psi_{\nu_2} \cdots \psi_{\nu_N} \rangle
\]

\[
= \sum_{\nu_j} \langle \psi_{\nu_j} | \sum_{k=1}^{N} \hat{T}_k | \psi_{\nu_k} \rangle \langle \psi_{\nu_k} | \hat{b}_k^\dagger \hat{b}_k | \psi_{\nu_1}, \psi_{\nu_2} \cdots \psi_{\nu_N} \rangle
\]

\[
= \sum_{k=1}^{N} \sum_{\nu_j, \nu_k} \langle \psi_{\nu_j} | \delta_{\nu_1, \nu_k} T_{\nu_j, \nu_k} \langle \psi_{\nu_k} | \hat{b}_k^\dagger \hat{b}_k | \psi_{\nu_1}, \psi_{\nu_2} \cdots \psi_{\nu_N} \rangle
\]

\[
= \sum_{k=1}^{N} \sum_{\nu_j, \nu_k} \delta_{\nu_1, \nu_k} T_{\nu_j, \nu_k} \langle \psi_{\nu_j} | \hat{b}_k^\dagger \hat{b}_k | \psi_{\nu_1}, \psi_{\nu_2} \cdots \psi_{\nu_N} \rangle
\]

86
Therefore,

\[
\hat{T} \left[ \hat{b}_{\nu_1}^\dagger \cdots \hat{b}_{\nu_N}^\dagger |0\rangle \right] = \sum_{k=1}^{N} \sum_{\nu_j,\nu_k} \delta_{\nu_j,\nu_k} T_{\nu_j,\nu_k} \hat{b}_{\nu_j}^\dagger \hat{b}_{\nu_k} \hat{b}_{\nu_1}^\dagger \cdots \hat{b}_{\nu_N}^\dagger |0\rangle
\]

\[
= \sum_{k=1}^{N} \sum_{\nu_j,\nu_k} \delta_{\nu_j,\nu_k} T_{\nu_j,\nu_k} \hat{b}_{\nu_j}^\dagger \hat{b}_{\nu_k} \hat{b}_{\nu_1}^\dagger \cdots \hat{b}_{\nu_N}^\dagger |0\rangle
\]

\[
= \sum_{k=1}^{N} \sum_{\nu_j,\nu_k} \delta_{\nu_j,\nu_k} T_{\nu_j,\nu_k} \frac{\hat{b}_{\nu_j}^\dagger \hat{b}_{\nu_k} \hat{b}_{\nu_1}^\dagger \cdots \hat{b}_{\nu_N}^\dagger |0\rangle}{\hat{b}_{\nu_1}^\dagger \cdots \hat{b}_{\nu_N}^\dagger}
\]

\[
= \sum_{\nu_j,\nu_k} T_{\nu_j,\nu_k} \sum_{k=1}^{N} \delta_{\nu_j,\nu_k} \left[ \hat{b}_{\nu_1}^\dagger \cdots \hat{b}_{\nu_N}^\dagger \right]
\]

\[
= \sum_{\nu_j,\nu_k} T_{\nu_j,\nu_k} \frac{1}{n_{\nu_k}} \sum_{k=1}^{N} \delta_{\nu_j,\nu_k} \left[ \hat{b}_{\nu_1}^\dagger \cdots \hat{b}_{\nu_N}^\dagger \right]
\]

\[
= \sum_{\nu_j,\nu_k} T_{\nu_j,\nu_k} \hat{b}_{\nu_j}^\dagger \hat{b}_{\nu_k} \left[ \hat{b}_{\nu_2}^\dagger \cdots \hat{b}_{\nu_N}^\dagger \right]
\]

(357)

where \( p \) is the number of particles in state state \( \psi_{\nu_k} \) for the \( N \)-particle system described by state \( |\psi_{\nu_1} \psi_{\nu_2} \cdots \psi_{\nu_N}\rangle = \hat{b}_{\nu_1}^\dagger \cdots \hat{b}_{\nu_N}^\dagger |0\rangle \), so according to Eq. (353), \( \langle \psi_{\nu_1} \psi_{\nu_2} \cdots \psi_{\nu_N}| \hat{b}_{\nu_k} \hat{b}_{\nu_k}^\dagger |\psi_{\nu_2} \psi_{\nu_2} \cdots \psi_{\nu_N}\rangle = (n_{\nu_k} + 1) \). Therefore,

\[
\hat{T} = \sum_{\nu_j,\nu_k} T_{\nu_j,\nu_k} \hat{b}_{\nu_j}^\dagger \hat{b}_{\nu_k}.
\]

(358)

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

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

(359)

can be written in second quantization, as follows:

\[
\hat{V} = \sum_{\nu_j,\nu_k} V_{\nu_j,\nu_k,\nu_k} \hat{b}_{\nu_j}^\dagger \hat{b}_{\nu_k} \hat{b}_{\nu_k} \hat{b}_{\nu_k}
\]

(360)

where \( V_{\nu_j,\nu_k,\nu_k} = \langle \psi_{\nu_j} \psi_{\nu_k}| V(x_1, x_2) |\psi_{\nu_k} \psi_{\nu_k}\rangle \).
29.5 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 \( j = 1 \text{–} N \). 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:

\[
|\psi_{\mu j}\rangle = \sum_k \langle \psi_{\nu k} | \psi_{\mu j}\rangle |\psi_{\nu k}\rangle,
\]

where \( |\psi_{\nu k}\rangle = \hat{a}_{\nu k}^\dagger |0\rangle \) and \( |\psi_{\mu j}\rangle = \hat{a}_{\mu j}^\dagger |0\rangle \). Therefore,

\[
\hat{a}_{\mu j} |0\rangle = \sum_k \langle \psi_{\nu k} | \psi_{\mu j}\rangle \hat{a}_{\nu k} |0\rangle,
\]

or

\[
\hat{a}_{\mu j} = \sum_k \langle \psi_{\nu k} | \psi_{\mu j}\rangle \hat{a}_{\nu k},
\]

and

\[
\hat{a}_{\mu j}^\dagger = \sum_k \langle \psi_{\nu k} | \psi_{\mu j}\rangle^* \hat{a}_{\nu k}^\dagger.
\]

29.6 Mapping into Cartesian Coordinates

Introducing the Cartesian operators \( \tilde{x}_{\nu j} = \frac{1}{\sqrt{2}} [\hat{b}_{\nu j}^\dagger + \hat{b}_{\nu j}] \) and \( \tilde{p}_{\nu j} = \frac{i}{\sqrt{2}} [\hat{b}_{\nu j}^\dagger - \hat{b}_{\nu j}] \), we obtain:

\[
\hat{b}_{\nu j}^\dagger = \frac{1}{\sqrt{2}} \left[ \tilde{x}_{\nu j} - i \tilde{p}_{\nu j} \right],
\]

and

\[
\hat{b}_{\nu j} = \frac{1}{\sqrt{2}} \left[ \tilde{x}_{\nu j} + i \tilde{p}_{\nu j} \right].
\]

Therefore,

\[
\hat{n}_{\nu j} = \frac{1}{2} (\tilde{x}_{\nu j} - i \tilde{p}_{\nu j}) (\tilde{x}_{\nu j} + i \tilde{p}_{\nu j})
\]

\[
= \frac{1}{2} (\tilde{x}_{\nu j}^2 + i [\tilde{x}_{\nu j}, \tilde{p}_{\nu j}] + \tilde{p}_{\nu j}^2)
\]

\[
= \frac{1}{2} (\tilde{x}_{\nu j}^2 + \tilde{p}_{\nu j}^2 - \hbar)
\]

Substituting into Eq. (368), we obtain:

\[
\hat{T} = \frac{1}{2} \sum_{\nu_j, \nu_l} T_{\nu_j, \nu_l} \left[ \tilde{x}_{\nu_j} - i \tilde{p}_{\nu_j} \right] \left[ \tilde{x}_{\nu_l} + i \tilde{p}_{\nu_l} \right],
\]

\[
= \frac{1}{2} \sum_{\nu_j} T_{\nu_j, \nu_j} (\tilde{x}_{\nu_j}^2 + \tilde{p}_{\nu_j}^2 - \hbar) + \frac{1}{2} \sum_{\nu_j, \nu_l} T_{\nu_j, \nu_l} \left[ \tilde{x}_{\nu_j} - i \tilde{p}_{\nu_j} \right] \left[ \tilde{x}_{\nu_l} + i \tilde{p}_{\nu_l} \right]
\]

\[
= \frac{1}{2} \sum_{\nu_j} T_{\nu_j, \nu_j} (\tilde{x}_{\nu_j}^2 + \tilde{p}_{\nu_j}^2 - \hbar) + \frac{1}{2} \sum_{\nu_j, \nu_l} T_{\nu_j, \nu_l} \left[ \tilde{x}_{\nu_j} \tilde{x}_{\nu_l} + \tilde{p}_{\nu_j} \tilde{p}_{\nu_l} \right].
\]
since $[\tilde{x}_{\nu_j}, \tilde{p}_{\nu_j}] = i\hbar\delta_{\nu_j, \nu_j}$ while $[\tilde{x}_{\nu_j}, \tilde{x}_{\nu_j}] = 0$ and $[\tilde{p}_{\nu_j}, \tilde{p}_{\nu_j}] = 0$.

### 30 Spin-Boson Model

Consider the spin-boson Hamiltonian

$$\hat{H} = \frac{2}{\hbar} \left[ \epsilon \hat{S}_z + J \hat{S}_x + \hat{S}_z \times f(y) \right] + H_b(y, p_y), \quad (369)$$

with $f(y) = \sum_{i=1}^{N} c_i y_i$, and $H_b(y, p_y) = \sum_{i=1}^{N} p_i^2/(2m_i) + 1/2m_i\omega_i y_i^2$. For example, for a spin $S = 1/2$, $\hat{H}$ can be written as follows:

$$H = \begin{bmatrix} \epsilon & 0 \\ 0 & -\epsilon \end{bmatrix} + \begin{bmatrix} 0 & J \\ J & 0 \end{bmatrix} \times f(y) + H_b(y, p_y), \quad (370)$$

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:

$$\hat{S}_z |S, m_s\rangle = \hbar m_s |S, m_s\rangle. \quad (371)$$

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:

$$\langle m'_s | \hat{S}_x | m_s \rangle = \hbar (\delta_{m'_s, m_s + 1} + \delta_{m'_s, m_s + 1, m_s}) \frac{1}{2} \sqrt{S(S + 1) - m'_s m_s}$$

$$\langle m'_s | \hat{S}_y | m_s \rangle = \hbar (\delta_{m'_s, m_s + 1} - \delta_{m'_s, m_s + 1, m_s}) \frac{1}{2} \sqrt{S(S + 1) - m'_s m_s}$$

$$\langle m'_s | \hat{S}_z | m_s \rangle = \hbar \delta_{m'_s, m_s} m_s$$

$$\langle m'_s | \hat{S}^+ | m_s \rangle = \hbar \delta_{m'_s, m_s + 1} \sqrt{S(S + 1) - m'_s m_s}$$

$$\langle m'_s | \hat{S}^- | m_s \rangle = \hbar \delta_{m'_s + 1, m_s} \sqrt{S(S + 1) - m'_s m_s}$$

$$\langle m'_s | \hat{S}^2 | m_s \rangle = \hbar \delta_{m'_s, m_s} S(S + 1)$$

so the same Hamiltonian introduced by Eq. (369) can be used to model a spin-boson model with
an arbitrary large number of states, as for example for 6 energy levels \((S=5/2)\) with

\[
S^+ = \hbar \begin{bmatrix}
0 & \sqrt{5} & 0 & 0 & 0 & 0 \\
0 & 0 & \sqrt{8} & 0 & 0 & 0 \\
0 & 0 & 0 & \sqrt{9} & 0 & 0 \\
0 & 0 & 0 & 0 & \sqrt{8} & 0 \\
0 & 0 & 0 & 0 & 0 & \sqrt{5} \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix},
\]

(373)

\[
S^- = \hbar \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
\sqrt{5} & 0 & 0 & 0 & 0 & 0 \\
0 & \sqrt{8} & 0 & 0 & 0 & 0 \\
0 & 0 & \sqrt{9} & 0 & 0 & 0 \\
0 & 0 & 0 & \sqrt{8} & 0 & 0 \\
0 & 0 & 0 & 0 & \sqrt{5} & 0
\end{bmatrix},
\]

and

\[
S_x = \frac{\hbar}{2} \begin{bmatrix}
0 & \sqrt{5} & 0 & 0 & 0 & 0 \\
\sqrt{5} & 0 & \sqrt{8} & 0 & 0 & 0 \\
0 & \sqrt{8} & 0 & \sqrt{9} & 0 & 0 \\
0 & 0 & \sqrt{9} & 0 & \sqrt{8} & 0 \\
0 & 0 & 0 & \sqrt{8} & 0 & \sqrt{5} \\
0 & 0 & 0 & 0 & \sqrt{5} & 0
\end{bmatrix},
\]

(374)

\[
S_y = \frac{\hbar}{2i} \begin{bmatrix}
0 & -\sqrt{5} & 0 & 0 & 0 & 0 \\
-\sqrt{5} & 0 & \sqrt{8} & 0 & 0 & 0 \\
0 & -\sqrt{8} & 0 & \sqrt{9} & 0 & 0 \\
0 & 0 & -\sqrt{9} & 0 & \sqrt{8} & 0 \\
0 & 0 & 0 & -\sqrt{8} & 0 & \sqrt{5} \\
0 & 0 & 0 & 0 & -\sqrt{5} & 0
\end{bmatrix},
\]

\[
S_z = \frac{\hbar}{2} \begin{bmatrix}
5 & 0 & 0 & 0 & 0 & 0 \\
0 & 3 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & -3 & 0 \\
0 & 0 & 0 & 0 & 0 & -5
\end{bmatrix},
\]

According to the second quantization mapping, the spin-boson Hamiltonian can be written as follows:

\[
\hat{H} = H_b(y, p_y) + \sum_{m_s, m_s'} \langle m_s' | (\epsilon + f(y)) \frac{2}{\hbar} \hat{S}_z + J \frac{2}{\hbar} \hat{S}_x | m_s \rangle \hat{a}_{m_s'} \hat{a}_{m_s}.
\]

(375)
and in Cartesian coordinates,
\[
\hat{H} = H_b(y, p_y) + \sum_{m_s} \frac{2}{\hbar} (\epsilon + f(y)) |m_s, \hat{S}_z| m_s \rangle [\hat{x}_{m_s}^2 + \hat{p}_{m_s}^2 - \hbar]
\]
\[
+ \sum_{m_s} \sum_{m'_s \neq m_s} \frac{2}{\hbar} (m_s | \hat{S}_z | m'_s) J [\hat{x}_{m_s} \hat{x}_{m'_s} + \hat{p}_{m_s} \hat{p}_{m'_s}]
\]

31 Holstein-Primakoff Mapping

According to the Holstein-Primakoff (HP) transformation, we re-write the spin operators in terms of boson operators, as follows:
\[
\hat{S}^+ = \hbar \sqrt{2S - \hat{N}\hat{a}},
\]
and
\[
\hat{S}^- = \hat{a}\hbar \sqrt{2S - \hat{N}},
\]
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 \(\hat{a}\) 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
\[
\hat{S}_z = \frac{\hbar}{2} (\sqrt{2S - \hat{N}} (1 + \hat{N}) \sqrt{2S - \hat{N}} - \hat{a}^\dagger (2S - \hat{N}) \hat{a}) = \frac{\hbar}{2} ((2S - \hat{N})(1 + \hat{N}) - \hat{a}^\dagger \hat{a} 2S + \hat{a}^\dagger \hat{a} \hat{N}) = \frac{\hbar}{2} (2S - \hat{N} + 2S \hat{N} - \hat{N}^2 - \hat{N} 2S + \hat{a}^\dagger \hat{N}) = \frac{\hbar}{2} (2S - \hat{N} + 2S \hat{N} - \hat{N}^2 - \hat{N} 2S + \hat{N}^2 - \hat{N}) = \hbar (S - \hat{N}).
\]
Therefore,
\[
\hat{S}_z = \hbar (S - \hat{N}),
\]
and
\[
\hat{S}_x = \frac{1}{2} [\hat{S}^+ + \hat{S}^-] = \frac{\hbar}{2} \left[ \sqrt{2S - \hat{N}} \hat{a}^\dagger \sqrt{2S - \hat{N}} \right],
\]

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 \(\tilde{p} = \frac{1}{\sqrt{2}} [\hat{a}^\dagger - \hat{a}]\), we obtain:
\[
\tilde{x} - i\tilde{p} = \sqrt{2}\hat{a}^\dagger,
\]
and
\[
\tilde{x} + i\tilde{p} = \sqrt{2}\hat{a}.
\]
Therefore,
\[
\hat{N} = \frac{1}{2} (\tilde{x} - i\tilde{p})(\tilde{x} + i\tilde{p})
\]
\[
= \frac{1}{2} (\tilde{x}^2 + i[\tilde{x}, \tilde{p}] + \tilde{p}^2)
\]
\[
= \frac{1}{2} (\tilde{x}^2 + \tilde{p}^2 - \hbar)
\]
Therefore, $S_z = \hbar (S - \frac{1}{2}(\bar{x}^2 + \bar{p}^2 - \hbar))$. Analogously, we obtain:

$$\hat{S}_x = \frac{\hbar}{2} \sqrt{S - \frac{1}{4}(\bar{x}^2 + \bar{p}^2 - \hbar)(\bar{x} + i\bar{p}) + (\bar{x} - i\bar{p})\sqrt{S - \frac{1}{4}(\bar{x}^2 + \bar{p}^2 - \hbar)}}. \quad (384)$$

and

$$\hat{\mathcal{H}} = H_b(\mathbf{y}, \mathbf{p}_y) + (\epsilon + f(\mathbf{y}))(2S - (\bar{x}^2 + \bar{p}^2 - \hbar)) + J \hat{x}\sqrt{4S - (\bar{x}^2 + \bar{p}^2 - \hbar)} \quad (385)$$

which in the classical limit (i.e., with $[\sqrt{S - \hat{N}/2}, \mathbf{p}] \approx 0$, and $[\sqrt{S - \hat{N}/2}, \mathbf{x}] \approx 0$), gives:

$$\hat{\mathcal{H}} = H_b(\mathbf{y}, \mathbf{p}_y) + (\epsilon + f(\mathbf{y}))(2S - (\bar{x}^2 + \bar{p}^2 - \hbar)) + J\mathbf{x}\sqrt{4S - (\bar{x}^2 + \bar{p}^2 - \hbar)} \quad (386)$$

In the limit of $S \gg 1$, we can approximate the raising and lowering operators, as follows: $S^+ = \hbar \hat{a} \sqrt{2S}$ and $S^- = \hbar \hat{a}^\dagger \sqrt{2S}$, since the square roots can be expanded in Taylor series, as follows:

$$\sqrt{\hat{S} - \hat{\mathcal{N}}/2} = \sqrt{\hat{S}} + \frac{1}{2}(2\hat{S})^{-1/2}(-\frac{\hat{\mathcal{N}}}{2}) + \ldots \quad (387)$$

Therefore, truncating the expansion introduced by Eq. (387) after the first term and substituting into Eq. (385), we obtain the 'simple spin-wave (SW) theory' expression:

$$\hat{\mathcal{H}} = H_b(\mathbf{y}, \mathbf{p}_y) + (\epsilon + f(\mathbf{y}))(2S - (\bar{x}^2 + \bar{p}^2 - \hbar)) + 2J\sqrt{S}\mathbf{x} \quad (388)$$

### 32 Schwinger Mapping

Spin operators can also be mapped according to the Schwinger-boson representation, in terms of pairs of (constrained) bosons, as follows:

$$\hat{S}^+ = \hbar \hat{b}^\dagger \hat{a}, \quad \hat{S}^- = \hbar \hat{a} \hat{b},$$  
$$\hat{S}_x = \frac{1}{2} \left[ \hat{S}^+, \hat{S}^- \right] = \frac{\hbar}{2} \left( \hat{b}^\dagger \hat{b} - \hat{a}^\dagger \hat{a} \right),$$  
$$\hat{S}_y = \frac{1}{2i} \left[ \hat{S}^+ - \hat{S}^- \right] = \frac{\hbar}{2i} \left( \hat{b}^\dagger \hat{a} + \hat{a}^\dagger \hat{b} \right),$$  
$$\hat{S}_y = \frac{1}{2i} \left( \hat{S}^+ - \hat{S}^- \right) = \frac{\hbar}{2i} \left( \hat{b}^\dagger \hat{a} - \hat{a}^\dagger \hat{b} \right), \quad (389)$$
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
\[
S = \frac{1}{2}(\hat{b}^\dagger \hat{b} + \hat{a}^\dagger \hat{a}).
\] (390)

A specific form of the constraint, introduced by Eq. (390), 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. (377)–(380).

Substituting the spin operators in Eq. (369), according to the Schwinger mapping, introduced by Eqs. (389), we obtain:
\[
\hat{H} = (\epsilon + f(y)) \left( \hat{b}^\dagger \hat{b} - \hat{a}^\dagger \hat{a} \right) + J \left( \hat{b}^\dagger \hat{a} + \hat{a}^\dagger \hat{b} \right) + H_b(y, p_y),
\] (391)

Changing variables, \(\hat{b} = \hat{b}_{\nu_1}\) and \(\hat{a} = \hat{b}_{\nu_2}\), defined in terms of the Cartesian coordinates, as follows:
\[
\hat{b}^\dagger_{\nu_j} = \frac{1}{\sqrt{2}} \left[ \tilde{x}_{\nu_j} - i \tilde{p}_{\nu_j} \right],
\] (392)

and
\[
\hat{b}_{\nu_j} = \frac{1}{\sqrt{2}} \left[ \tilde{x}_{\nu_j} + i \tilde{p}_{\nu_j} \right],
\] (393)

with
\[
\hat{n}_{\nu_j} = \frac{1}{2} (\tilde{x}^2_{\nu_j} + \tilde{p}^2_{\nu_j} - \hbar)
\] (394)

we obtain
\[
\hat{H} = H_b(y, p_y) + (\epsilon + f(y)) \left[ (\tilde{x}^2_{\nu_1} + \tilde{p}^2_{\nu_1} - \hbar) - (\tilde{x}^2_{\nu_2} + \tilde{p}^2_{\nu_2} - \hbar) \right] + J (\tilde{x}_{\nu_1} \tilde{x}_{\nu_2} + \tilde{p}_{\nu_1} \tilde{p}_{\nu_2}),
\] (395)

which is the same Hamiltonian given by Eq. (376), according to the second quantization mapping, although with the constraint
\[
S = \frac{1}{4} \left[ (\tilde{x}^2_{\nu_1} + \tilde{p}^2_{\nu_1} - \hbar) + (\tilde{x}^2_{\nu_2} + \tilde{p}^2_{\nu_2} - \hbar) \right].
\] (396)
33 Solutions to Computational Assignments

33.1 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 \( x_j = x_{\text{min}} + (j - 1)\Delta \) with finite resolution \( \Delta = (x_{\text{max}} - x_{\text{min}})/(n - 1) \) 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:

```bash
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

```bash
quit
```
PROGRAM Problem_1
  call Initialize()
  CALL SAVEWF(0)
END

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)
    *exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
  end do
  RETURN
END

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
33.2 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
```

97
PROGRAM Problem2
    call Initialize()
    CALL SAVEFT()
END

SUBROUTINE Initialize()
    ! Wave Packet Initialization: Gaussian centered at xk, with momentum pk
    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

SUBROUTINE SAVEFT()
    ! Save wave-packet in momentum space
    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
d end
ccc

subroutine FT_analy(Psip,p)
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)
ccc

99
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

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

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
               TEMP=DATA(I3)
               TEMPI=DATA(I3+1)
               DATA(I3)=DATA(I3REV)
               DATA(I3+1)=DATA(I3REV+1)
               DATA(I3REV)=TEMP
               DATA(I3REV+1)=TEMPI
            12 CONTINUE
            13 CONTINUE
      ELSE
         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
      ENDIF
   14 CONTINUE
   IFP1=IP1
   2 IF(IFP1.LT.IP2)THEN
      IFP2=2*IFP1
      THETA=ISIGN*6.28318530717959D0/(IFP2/IP1)
      WPR=-2.0D0+DSIN(0.5D0*THETA)**2
      WPI=DSIN(THETA)
   100

100
WR=1.D0
WI=0.D0
DO 17 I3=1,IP1,IFP1
   DO 16 I1=I3,I3+IP1-2,2
      DO 15 I2=I1,IP3,IPF2
         K1=I2
         K2=K1+IFP1
         TEMPL=SNGL(WR)*DATA(K2)-SNGL(WI)*DATA(K2+1)
         TEMPI=SNGL(WR)*DATA(K2+1)+SNGL(WI)*DATA(K2)
         DATA(K2)=DATA(K1)-TEMPL
         DATA(K2+1)=DATA(K1+1)-TEMPI
         DATA(K1)=DATA(K1)+TEMPL
         DATA(K1+1)=DATA(K1+1)+TEMPI
      CONTINUE
   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
33.3  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 \( \langle \Psi_t | \hat{x} | \Psi_t \rangle \).
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

SUBROUTINE Initialize()
!
Wave Packet Initialization: Gaussian centered at xk, with momentum pk
!
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)*exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
end do
RETURN
END

SUBROUTINE PE(RV)
!
Expectation Value of the Potential Energy
!
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

SUBROUTINE Px(RV)
  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

SUBROUTINE VA(V,x)
  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
33.4 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 $\Psi_0$ 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.
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

SUBROUTINE Initialize()

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

SUBROUTINE PE(RV)

IMPLICIT NONE
INTEGER nptx,npts,k
COMPLEX chi
REAL Vpot,RV,xmin,xmax,dx,x
PARAMETER(npts=10,nptx=2**npts)
SUBROUTINE KE(RKE)

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
Expectation value of the momentum

```
IMPLICIT NONE
INTEGER kk,nptx,kx,nx,npts
REAL dp,pe,p,xmin,xmax,pi,alenx,dx,mass,xk,pk
COMPLEX eye,chi,Psip,chi
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
```
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
      I2REV=I2REV+IP1/2
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)

1 IF(IPF1.LT.IP2) THEN
   IFP2=2*IFP1
   THETA=ISIGN*6.28318530717959D0/(IFP2/IP1)
   WPR=-2.0D0*DSIN(0.5D0*THETA)**2
   WPI=DSIN(THETA)
   WR=1.0D0
   WI=0.0D0
   DO 17 I3=1,IFP1,IP1
      DO 16 I1=I3, I3+IP1-2, 2
         DO 15 I2=I1, IP3, IPF2
            K1=I2
            K2=K1+IFP1
15 CONTINUE
16 CONTINUE
17 CONTINUE
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
33.5 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 \( \tau \).

Expanding the left-hand-side (l.h.s.) of Eq. (18) from the lecture notes gives:

\[
e^{-i\hat{H}\tau} = 1 - i\hat{H}\tau - \frac{1}{2}\hat{H}^2\tau^2 + O(\tau^3),
\]

where \( \hat{H} = \hat{p}^2/(2m) + \hat{V} \). Therefore,

\[
e^{-i\hat{H}\tau} = 1 - i\hat{H}\tau - \frac{1}{2}\hat{p}^4\tau^2 - \frac{1}{2}\hat{V}^2\tau^2 - \frac{1}{2}\hat{p}^2\hat{V}\tau^2 - \frac{1}{2}\hat{V}\hat{p}^2\tau^2 + O(\tau^3),
\]

In order to show that the Trotter expansion, introduced by Eq. (18), is accurate to second order in \( \tau \), we must expand the right-hand-side (r.h.s.) of Eq. (18) and show that such an expansion equals the r.h.s. of Eq. (398).

Expanding the right-hand-side (r.h.s.) of Eq. (18) gives,

\[
e^{-i\hat{V}(\hat{x})\tau/2}e^{-i\hat{p}^2\tau/(2m)}e^{-i\hat{V}(\hat{x})\tau/2} = \left(1 - i\hat{V}\tau/2 - \frac{1}{2}\hat{V}^2\tau^2/4 + O(\tau^3)\right)\left(1 - i\hat{p}^2\tau - \frac{1}{2}\hat{p}^4\tau^2 + O(\tau^3)\right)
\]

\[
\times \left(1 - i\hat{V}\tau/2 - \frac{1}{2}\hat{V}^2\tau^2/4 + O(\tau^3)\right),
\]

\[
(399)
\]

\[
e^{-i\hat{V}(\hat{x})\tau/2}e^{-i\hat{p}^2\tau/(2m)}e^{-i\hat{V}(\hat{x})\tau/2} = \left(1 - i\hat{V}\tau/2 - \frac{1}{2}\hat{V}^2\tau^2/4 - i\hat{p}^2\tau/2m - \hat{V}\hat{p}^2\tau^2/2 - \frac{1}{2}\hat{p}^4\tau^2 + O(\tau^3)\right)
\]

\[
\times \left(1 - i\hat{V}\tau/2 - \frac{1}{2}\hat{V}^2\tau^2/4 + O(\tau^3)\right),
\]

\[
(400)
\]

\[
e^{-i\hat{V}(\hat{x})\tau/2}e^{-i\hat{p}^2\tau/(2m)}e^{-i\hat{V}(\hat{x})\tau/2} = 1 - i\hat{V}\tau/2 - \frac{1}{2}\hat{V}^2\tau^2/4 - i\hat{p}^2\tau/2m - \hat{V}\hat{p}^2\tau^2/2 - \frac{1}{2}\hat{p}^4\tau^2
\]

\[
- i\hat{V}\tau/2 - \hat{V}^2\tau^2/4 - \frac{\hat{p}^2}{2m}\hat{V}\tau^2/2 - \frac{1}{2}\hat{V}^2\tau^2/4 + O(\tau^3),
\]

\[
(401)
\]

\[
e^{-i\hat{V}(\hat{x})\tau/2}e^{-i\hat{p}^2\tau/(2m)}e^{-i\hat{V}(\hat{x})\tau/2} = 1 - i\hat{V}\tau - \frac{1}{2}\hat{V}^2\tau^2 - \frac{1}{2}\hat{V}^2\tau^2/2 - \frac{1}{2}\hat{p}^4\tau^2
\]

\[
- \frac{\hat{p}^2}{2m}\hat{V}\tau^2/2 + O(\tau^3).
\]

Note that the r.h.s. of Eq. (402) is identical to the r.h.s. of Eq. (398), completing the proof that the Trotter expansion, introduced by Eq. (18), is accurate to second order in \( \tau \).
33.6 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
```
PROGRAM Problem6

1-D wave packet propagation

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

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)
      PRINT *, "Step=", istep-1," Final step=", nstep
   IF(istep.GE.1) CALL PROPAGATE(vprop,tprop)
   IF(mod((istep-1),ndump).EQ.0) THEN
      CALL SAVEWF(istep,ndump,dt)
   END IF
END DO
22 FORMAT(6(e13.6,2x))
END

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine ReadParam(nstep,ndump,dt)

Parameters defining the grid (xmin, xmax), integration time step (dt),
mass (rmass), initial position (xk), initial momentum (pk),
number of propagation steps (nstep), and how often to save a pic (ndump)

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

xmin=-10.0
xmax= 10.0
dt=0.1
rmass=1.0
xk=-2.5
pk=1.0
nstep=1
ndump=1
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)
xk=kk
pc=pk

Wave Packet Initialization: Gaussian centered at xk, with momentum pk
alpha=rmass*omega
do kk=1,nptx
  x=xmin+kk*dx
  chi(kk,1)=((alpha/pi)**0.25)*exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
  chi0(kk,1)=chi(kk,1)
end do

Hamiltonian Matrix CRV
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

SUBROUTINE HAMIL(CRV,x)

Hamiltonian Matrix

IMPLICIT NONE
INTEGER NN
REAL x,VPOT1
COMPLEX CRV
PARAMETER(NN=1)
DIMENSION CRV(NN,NN)

CALL VA(VPOT1,x)
CRV(1,1)=VPOT1

RETURN
END

SUBROUTINE VA(V,x)

Potential Energy Surface: Harmonic Oscillator

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**2
RETURN
END

SUBROUTINE SetKinProp(dt,tprop)

Kinetic Energy part of the Trotter Expansion: \( \exp(-i \frac{p^2 dt}{2m}) \)

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
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

return
end

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

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)

  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
end function Psia
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
cl=rgammap*x+alpha*xk+eye*pk
c2=rgamma+alpha/2.

Psia = A*(alpha/pi)**.25*sqrt(pi/c2)*
expc0+c1**2/(4.0*c2)

SUBROUTINE SAVEWF(je2,ndump,dt)

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

Save Wave-packet components

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
Save Adiabatic states

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
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
SUBROUTINE PE(RV)
c
Expectation Value of the Potential Energy
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
subroutine KE(RKE)
  
  c Expectation value of the kinetic energy
  
  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)
  
  pi = acos(-1.0)
  dx=(xmax-xmin)/nptx
  dp=2.*pi/(xmax-xmin)
  
  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

SUBROUTINE PROPAGATE(vprop,tprop)
  
  c Split Operator Fourier Transform Propagation Method
  
  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)

Apply potential energy part of the Trotter Expansion

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

Fourier Transform wave-packet to the momentum representation

CALL fourn(chin1,nptx,1,-1)

Apply kinetic energy part of the Trotter Expansion

DO i=1,nptx
  chin1(i)=tprop(i)*chin1(i)
END DO

Inverse Fourier Transform wave-packet to the coordinate representation

CALL fourn(chin1,nptx,1,1)

Apply potential energy part of the Trotter Expansion

DO i=1,nptx
  DO j=1,NN
    chi(i,j)=vprop(i,j,1)*chin1(i)
  END DO
END DO

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
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:

\[
\begin{align*}
p_{j+1} &= p_j + (F(x_j) + F(x_{j+1}))\tau/2 \\
x_{j+1} &= x_j + p_j\tau/m + F(x_j)\tau^2/(2m).
\end{align*}
\]

(403)

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 \( \langle \Psi_t | \hat{x} | \Psi_t \rangle \) 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 \( \langle \Psi_t | \hat{p} | \Psi_t \rangle \) 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 \( \langle \Psi_t | \hat{p} | \Psi_t \rangle \) vs. \( \langle \Psi_t | \hat{x} | \Psi_t \rangle \) can be obtained by typing

123
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 1  
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
```

124
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

127
plot "arch.0091" u 1:2 lw 3
pause .1
plot "arch.0092" u 1:2 lw 3
pause .1
plot "arch.0093" u 1:2 lw 3
pause .1
plot "arch.0094" u 1:2 lw 3
pause .1
plot "arch.0095" u 1:2 lw 3
pause .1
plot "arch.0096" u 1:2 lw 3
pause .1
plot "arch.0097" u 1:2 lw 3
pause .1
plot "arch.0098" u 1:2 lw 3
pause .1
plot "arch.0099" u 1:2 lw 3
pause .1
PROGRAM Problem7

c
1-D wave packet propagation and Velocity-Verlet propagation
on a Harmonic potential energy surface

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

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)
     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
       real(x_mean(1)),real(p_mean(1)),xc,pc
     END IF
  END IF
END DO
CLOSE(10)
22 FORMAT(6(e13.6,2x))
END

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine ReadParam(nstep,ndump,dt)

Parameters defining the grid (xmin, xmax), integration time step (dt),
rmass (rmass), initial position (xk), initial momentum (pk),
number of propagation steps (nstep), and how often to save a pic (ndump)

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
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
SUBROUTINE VV(dt)

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
Compute Force

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
Advance momenta half a step

c
pc=pc+0.5*F*dt

c
Advance coordinates a step

c
xc=xc+v*dt+0.5*dt**2*F/rmass

c
Compute Force

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
cend

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

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 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

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

SUBROUTINE HAMIL(CRV,x)

c Hamiltonian Matrix

IMPLICIT NONE
INTEGER NN
REAL x,VPOT1
COMPLEX CRV
PARAMETER(NN=1)
DIMENSION CRV(NN,NN)
c

131
CALL VA(VPOT1,x)
CRV(1,1)=VPOT1

RETURN
END

SUBROUTINE VA(V,x)

c Potential Energy Surface: Harmonic Oscillator

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

subroutine SetKinProp(dt,tprop)

c Kinetic Energy part of the Trotter Expansion: exp(-i p^2 dt/(2 m))

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

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

return
end

subroutine SetPotProp(dt,vprop)

132
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)

do ii=1, nptx
x=xmin+ii*dx
CALL VA(VPOT, x)
vprop(ii, 1, 1)=\( \exp(-\text{eye} \cdot 0.5 \cdot dt \cdot VPOT) \sqrt{\text{nptx} \cdot 1.0} \)
END DO
RETURN
END

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
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))))

1 1/(exp(beta*omega)-exp(-beta*omega))

rgamma=0.5*rmass*omega*(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.

Psia = A*(alpha/pi)**.25*sqrt(pi/c2)*
1 exp(-rgamma *x**2)*exp(c0+c1**2/(4.0*c2))

SUBROUTINE SAVEWF(je2,ndump,dt)

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,cla,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)

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

Save Wave-packet components

do kk=1,nptx
  x=xmin+kk*dx
  cl=chi(kk,1)*conjg(chi(kk,1))
  cla=Psia(x,je2,dt)*conjg(Psia(x,je2,dt))
  write(1,33) x,sqrt(cl)+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
Save Adiabatic states

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
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
SUBROUTINE XM(RV)
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
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
SUBROUTINE PE(RV)
c
Expectation Value of the Potential Energy
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

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

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
  end do
END DO
RETURN
END
end if
p=0.
if(nx.ne.0) p = real(nx)*dp
chic(kx)=p**2/(2.0*rmass)*chic(kx)/nptx
end do
CALL fourn(chic,nptx,1,1)
do kk=1,nptx
  RKE(j)=RKE(j)+conjg(chi(kk,j))*chic(kk)*dx
end do
END DO
return
end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine PM(RKE)
c
  c  Expectation value of the kinetic energy
  c
  IMPLICIT NONE
  INTEGER NN,kk,nptx,kx,nx,npts,j
  REAL dp,theta,wm,p,xmin,xmax,rmass,xk,pi,alenx,pk,rm,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
SUBROUTINE PROPAGATE(vprop,tprop)

Split Operator Fourier Transform Propagation Method

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)

Apply potential energy part of the Trotter Expansion

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

Fourier Transform wave-packet to the momentum representation

CALL fourn(chin1,nptx,1,-1)

Apply kinetic energy part of the Trotter Expansion

DO i=1,nptx
   chin1(i)=tprop(i)*chin1(i)
END DO

Inverse Fourier Transform wave-packet to the coordinate representation

CALL fourn(chin1,nptx,1,1)

Apply potential energy part of the Trotter Expansion

DO i=1,nptx
   DO j=1,NN
      chi(i,j)=vprop(i,j,1)*chin1(i)
   END DO
END DO

Subroutine for FFT from Numerical Recipes
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=-2.0*D0*DSIN(0.5D0*THETA)**2
   WPI=DSIN(THETA)
   WR=1.0D0
   WI=0.0D0
   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
END IF
18 CONTINUE
RETURN
END
33.8  Problem 8

**Computational Problem 8**: Change the potential to that of a Morse oscillator $V(\hat{x}) = D_e(1 - \exp(-a(\hat{x} - x_e)))^2$, 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., 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 $\langle \Psi_t | \hat{x} | \Psi_t \rangle$ 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 $\langle \Psi_t | \hat{p} | \Psi_t \rangle$ 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 $\langle \Psi_t | \hat{p} | \Psi_t \rangle$ vs. $\langle \Psi_t | \hat{x} | \Psi_t \rangle$ 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](http://ursula.chem.yale.edu/~batista/classes/summer/P8/pp_8)

```plaintext
set yrange[0:9]
set xrange[-5:25]
set dat sty 1
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
```
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

143
pause .1
plot "arch.0069" u 1:2 lw 3
pause .1
plot "arch.0070" u 1:2 lw 3
pause .1
plot "arch.0071" u 1:2 lw 3
pause .1
plot "arch.0072" u 1:2 lw 3
pause .1
plot "arch.0073" u 1:2 lw 3
pause .1
plot "arch.0074" u 1:2 lw 3
pause .1
plot "arch.0075" u 1:2 lw 3
pause .1
plot "arch.0076" u 1:2 lw 3
pause .1
plot "arch.0077" u 1:2 lw 3
pause .1
plot "arch.0078" u 1:2 lw 3
pause .1
plot "arch.0079" u 1:2 lw 3
pause .1
plot "arch.0080" u 1:2 lw 3
pause .1
plot "arch.0081" u 1:2 lw 3
pause .1
plot "arch.0082" u 1:2 lw 3
pause .1
plot "arch.0083" u 1:2 lw 3
pause .1
plot "arch.0084" u 1:2 lw 3
pause .1
plot "arch.0085" u 1:2 lw 3
pause .1
plot "arch.0086" u 1:2 lw 3
pause .1
plot "arch.0087" u 1:2 lw 3
pause .1
plot "arch.0088" u 1:2 lw 3
pause .1
plot "arch.0089" u 1:2 lw 3
pause .1
plot "arch.0090" u 1:2 lw 3
pause .1
plot "arch.0091" u 1:2 lw 3
pause .1
plot "arch.0092" u 1:2 lw 3
pause .1
plot "arch.0093" u 1:2 lw 3
pause .1
plot "arch.0094" u 1:2 lw 3
pause .1
plot "arch.0095" u 1:2 lw 3
pause .1
plot "arch.0096" u 1:2 lw 3
pause .1
plot "arch.0097" u 1:2 lw 3
pause .1
plot "arch.0098" u 1:2 lw 3
pause .1
plot "arch.0099" u 1:2 lw 3
pause .1
PROGRAM Problem8

c
1-D wave packet propagation and Velocity-Verlet propagation

on a Morse potential energy surface

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

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)
      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
         ,real(x_mean(1)),real(p_mean(1)),xc,pc
      END IF
   END IF
END DO
CLOSE(10)
22 FORMAT(6(e13.6,2x))
END

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

subroutine ReadParam(nstep,ndump,dt)

Parameters defining the grid (xmin, xmax), integration time step (dt),
rmass (rmass), initial position (xk), initial momentum (pk),
number of propagation steps (nstep), and how often to save a pic (ndump)

IMPLICIT NONE
INTEGER ntype,nstep,nrpt,ireport,ndump,nlit
REAL xmin,xmax,pk,rmass,xk,dt

147
common /packet/ rmass,xk,pk
common /xy/ xmin,xmax

xmin=-5.0
xmax=25.0
dt=0.2
rmass=1.0
xk=-.5
pk=0.0
nstep=100
ndump=1

return
end

SUBROUTINE VV(dt)


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

Compute Force

dx=0.01
xt=xc+dx
CALL VA(VPOT1,xt)
x=xt-dx
CALL VA(VPOT2,xt)
F=-(VPOT1-VPOT2)/(2.0*dx)
v=pc/rmass

Advance momenta half a step

pc=pc+0.5*F*dt

Advance coordinates a step

xc=xc+v*dt+0.5*dt**2*F/rmass

Compute Force

dx=0.01
xt=xc+dx
CALL VA(VPOT1,xt)
x=xt-dx
CALL VA(VPOT2,xt)
F=-(VPOT1-VPOT2)/(2.0*dx)
c Advance momenta half a step
c
c pc=pc+0.5*F*dt
c
return
end

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

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,omega1,omega2
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

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)*exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
  chi0(kk,1)=chi(kk,1)
end do
RETURN
END

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
SUBROUTINE HAMIL(CRV,x)

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

SUBROUTINE VA(V,x)

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

SUBROUTINE SetKinProp(dt,tprop)

Kinetic Energy part of the Trotter Expansion: exp(-i p^2 dt/(2 m))

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

return
subroutine SetPotProp(dt,vprop)
  
  Potential Energy part of the Trotter Expansion: \( \exp(-i V dt/2) \)
  
  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)
  
  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

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

SUBROUTINE SAVEWF(je2, ndump, dt)
  
  Dump Time Evolved Wave packet
  
  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, cla, 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

151
COMMON / wfunc/ chi(nptx,NN)
COMMON /ENER/ energy(NN)

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)
c
ncount=(je2-1)/ndump

c Save Wave-packet components

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 Save Adiabatic states

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

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
SUBROUTINE XM(RV)
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)
SUBROUTINE PE(RV)

! Expectation Value of the Potential Energy
!

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

SUBROUTINE KE(RKE)

! Expectation value of the kinetic energy
!

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)
pi = acos(-1.0)
dx=(xmax-xmin)/nptx
dp=2.*pi/(xmax-xmin)

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

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine PM(RKE)
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
Expectation value of the kinetic energy
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
IMPLICIT NONE
INTEGER NN,kk,nptx,kx,nx,npts,j
REAL dp,theta,wm,p,xmin,xmax,rmass,xk,pi,alenx,pk,rm,ri,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)

pi = acos(-1.0)
dx=(xmax-xmin)/nptx
dp=2.*pi/(xmax-xmin)

DO j=1,NN
RKE(j)=0.0
do kk=1,nptx
   chic(kk)=chi(kk,j)
end do

154
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

SUBROUTINE PROPAGATE(vprop,tprop)
c
  Split Operator Fourier Transform Propagation Method
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
  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
END DO
c
  Fourier Transform wave-packet to the momentum representation
c
CALL fourn(chin1,nptx,1,-1)
c
  Apply kinetic energy part of the Trotter Expansion
c
DO i=1,nptx
    chin1(i)=tprop(i)*chin1(i)
END DO
c
Inverse Fourier Transform wave-packet to the coordinate representation

CALL fourn(chin1, nptx, 1, 1)

Apply potential energy part of the Trotter Expansion

DO i=1,nptx
  DO j=1,NN
    chi(i,j)=vprop(i,j,1)*chin1(i)
  END DO
END DO
END

Subroutine for FFT from Numerical Recipes

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
14    CONTINUE
15    CONTINUE
    IBIT=IP2/2
1    IF((IBIT.GE.IP1).AND.(I2REV.GT.IBIT)) THEN
      I2REV=I2REV-IBIT
      IBIT=IBIT/2
      GO TO 1
    ENDIF
17    I2REV=I2REV+IBIT
CONTINUE
IFP1=IP1

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,IP1,IP1
   DO 16 I1=I3,I3+IP1-2,2
     DO 15 I2=I1,IP3,IPF2
       K1=I2
       K2=K1+IPF1
       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

CONTINUE
RETURN
END
33.9 Problem 9

**Computational Problem 9:** Simulate the propagation of a wave-packet with $x_0 = -5.5$ and initial momentum $p_0 = 2$ colliding with a barrier potential $V(x) = 3$, if $\text{abs}(x) < 0.5$, and $V(x) = 0$, otherwise. Hint: In order to avoid artificial recurrences you might need to add an absorbing imaginary potential $V_a(x) = i(\text{abs}(x) - 10)^4$, if $\text{abs}(x) > 10$, and $V_a(x) = 0$, otherwise.

The output of this program can be generated and visualized as follows. Cut the source code attached below, save it in a file named Problem9.f, compile it by typing

```
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](http://ursula.chem.yale.edu/~batista/classes/summer/P9/pp_9)
plot "arch.0088" u 1:2 lw 3 pause .1
plot "arch.0089" u 1:2 lw 3 pause .1
plot "arch.0090" u 1:2 lw 3 pause .1
plot "arch.0091" u 1:2 lw 3 pause .1
plot "arch.0092" u 1:2 lw 3 pause .1
plot "arch.0093" u 1:2 lw 3 pause .1
plot "arch.0094" u 1:2 lw 3 pause .1
plot "arch.0095" u 1:2 lw 3 pause .1
plot "arch.0096" u 1:2 lw 3 pause .1
plot "arch.0097" u 1:2 lw 3 pause .1
plot "arch.0098" u 1:2 lw 3 pause .1
plot "arch.0099" u 1:2 lw 3 pause .1
PROGRAM Problem9

1-D wave packet propagation of tunneling through a barrier

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

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 IF
END DO
END

Subroutine ReadParam(nstep,ndump,dt)

Parameters defining the grid (xmin, xmax), integration time step (dt),
rmass (rmass), initial position (xk), initial momentum (pk),
number of propagation steps (nstep), and how often to save a pic (ndump)

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

xmin=-13.0
xmax=13.0
dt=0.1
rmass=1.0
xk=-4.5
pk=1.
nstep=100
ndump=1

163
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,omega,xk2
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

Wave Packet Initialization: Gaussian centered at xk, with momentum pk

alpha=rmass*omega
do kk=1,nptx
   x=xmin+kk*dx
   chi(kk,1)=((alpha/pi)**0.25)
   *exp(-alpha/2.*(x-xk)**2+EYE*pk*(x-xk))
   chi0(kk,1)=chi(kk,1)
end do
RETURN
END

SUBROUTINE HAMIL(CRV,x)

IMPLICIT NONE
INTEGER NN
REAL x,VPOT1
COMPLEX CRV
PARAMETER(NN=1)
DIMENSION CRV(NN,NN)

CALL VA(VPOT1,x)
CRV(1,1)=VPOT1
RETURN

RETURN
end

return
SUBROUTINE VA(V,x)

    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

SUBROUTINE SetKinProp(dt,tprop)

    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

SUBROUTINE SetPotProp(dt,vprop)

    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
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
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
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
SUBROUTINE SAVEWF(je2,ndump,dt)
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,cla,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

Save Wave-packet components

do kk=1,nptx
  x=xmin+kk*dx
  cl=chi(kk,1)+conjg(chi(kk,1))
  write(1,33) x,sqrt(cl)+real(energy(1))
end do
write(1,33)
do kk=1,nptx
  x=xmin+kk*dx
  write(1,33) x,real(chi(kk,1))+real(energy(1))
end do
write(1,33)

Save Adiabatic states

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

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
SUBROUTINE PE(RV)

Expectation Value of the Potential Energy

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)
  end do
RV(j)=RV(j)+chi(kk,j)\times Vpot\times\text{conjg}(chi(kk,j))\times dx
end do
END DO
RETURN
END

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

subroutine KE(RKE)
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 = \text{acos}(-1.0)
dx=(xmax-xmin)/nptx
dp=2.\times 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 = \text{real}(nx)\times dp
  chic(kx)=p**2/(2.0\times rmass)\times\text{chi}(kx)/nptx
end do
CALL fourn(chic,nptx,1,1)
do kk=1,nptx
  RKE(j)=RKE(j)+\text{conjg}(chi(kk,j))\times\text{chic}(kk)\times dx
end do
END DO
return
end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
SUBROUTINE PROPAGATE(vprop,tprop)
c
Split Operator Fourier Transform Propagation Method
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
Apply potential energy part of the Trotter Expansion
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 Fourier Transform wave-packet to the momentum representation
CALL fourn(chin1,nptx,1,-1)
c
Apply kinetic energy part of the Trotter Expansion
DO i=1,nptx
   chin1(i)=tprop(i)*chin1(i)
END DO

c Inverse Fourier Transform wave-packet to the coordinate representation
CALL fourn(chin1,nptx,1,1)
c
Apply potential energy part of the Trotter Expansion
DO i=1,nptx
   DO j=1,NN
      chi(i,j)=vprop(i,j,1)*chin1(i)
   END DO
END DO
END
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,IPF2
                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
170
\[
\begin{align*}
\text{WTEMP} &= \text{WR} \\
\text{WR} &= \text{WR} \times \text{WPR} - \text{WI} \times \text{WPI} + \text{WR} \\
\text{WI} &= \text{WI} \times \text{WPR} + \text{WTEMP} \times \text{WPI} + \text{WI}
\end{align*}
\]

17 CONTINUE
IFP1 = IFP2
ENDIF
GO TO 2
NPREV = N * NPREV
18 CONTINUE
RETURN
END
Problem 10: (a) Derive Eq. (38) by considering that,

\[
e^{-iV_c^2 \tau} = D^\dagger \begin{pmatrix} e^{iV_c(x)2\tau} & 0 \\ 0 & e^{-iV_c(x)2\tau} \end{pmatrix} D,
\]

with

\[
D = D^\dagger = \begin{pmatrix} -1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix},
\]

since

\[
e^{-iV_c^2 \tau} = 1 + (-iV_c 2\tau) + \frac{1}{2!} (-iV_c 2\tau)^2 + \ldots,
\]

and

\[
V_c \equiv \begin{pmatrix} 0 & V_c(x) \\ V_c(x) & 0 \end{pmatrix} = D^\dagger \begin{pmatrix} -V_c(x) & 0 \\ 0 & V_c(x) \end{pmatrix} D,
\]

with \(DD^\dagger = 1\).

In order to derive Eq. (28) we need to prove the following equation:

\[
e^{-i\bar{V}_0 \tau} e^{-iV_c^2 \tau} e^{-i\bar{V}_0 \tau} = \begin{pmatrix} e^{-iV_1(x)2\tau} \cos(2V_c(x)\tau) & -i \sin(2V_c(x)\tau) e^{-i(\bar{V}_1(x) + \bar{V}_2(x))\tau} \\ -i \sin(2V_c(x)\tau) e^{-i(\bar{V}_1(x) + \bar{V}_2(x))\tau} & \cos(2V_c(x)\tau) e^{-iV_2(x)2\tau} \end{pmatrix}.
\]

where

\[
e^{-i\bar{V}_0 \tau} = e^{-i \begin{pmatrix} V_1(x) & 0 \\ 0 & V_2(x) \end{pmatrix} \tau}.
\]

Expanding the exponential on the r.h.s. of Eq. (409) gives

\[
e^{-i\tau \begin{pmatrix} V_1(x) & 0 \\ 0 & V_2(x) \end{pmatrix}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} -i\tau V_1(x) & 0 \\ 0 & -i\tau V_2(x) \end{pmatrix} + \begin{pmatrix} \frac{1}{2!} V_1(x)^2 (-i\tau)^2 & 0 \\ 0 & \frac{1}{2!} V_2(x)^2 (-i\tau)^2 \end{pmatrix} + \ldots
\]

Therefore,

\[
e^{-i\tau \begin{pmatrix} V_1(x) & 0 \\ 0 & V_2(x) \end{pmatrix}} = \begin{pmatrix} e^{-iV_1(x)\tau} & 0 \\ 0 & e^{-iV_2(x)\tau} \end{pmatrix}.
\]

In addition, according to Eq. (30),

\[
e^{-iV_c^2 \tau} = D^\dagger \begin{pmatrix} e^{iV_c(x)2\tau} & 0 \\ 0 & e^{-iV_c(x)2\tau} \end{pmatrix} D,
\]
with
\[ D = D^\dagger \equiv \begin{pmatrix} -1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}, \] (413)

Therefore,
\[ e^{-iV_0 \tau} e^{-iV_c 2\tau} e^{-iV_0 \tau} = \begin{pmatrix} e^{-iV_1(x) \tau} & 0 \\ 0 & e^{-iV_2(x) \tau} \end{pmatrix} D^\dagger \begin{pmatrix} e^{iV_c(x) 2\tau} & 0 \\ 0 & e^{-iV_c(x) 2\tau} \end{pmatrix} D \begin{pmatrix} e^{-iV_1(x) \tau} & 0 \\ 0 & e^{-iV_2(x) \tau} \end{pmatrix}. \] (414)

The multiplication of the five matrices on the r.h.s. of Eq. (414) gives the matrix on the r.h.s. of Eq. (408).
33.11 Problem 11

Problem 11: Derive Eq. (37) by writing the matrix $V$ in the basis of adiabatic eigenstates

\[ \phi_1(x) = L_{11}(x)|1\rangle + L_{21}(x)|2\rangle, \]
\[ \phi_2(x) = L_{12}(x)|1\rangle + L_{22}(x)|2\rangle, \]  \hspace{1cm} (415)

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

\[ e^{-iV\tau} = 1 + (-iV\tau) + \frac{1}{2!}(-iV\tau)^2 + \ldots, \]  \hspace{1cm} (416)

show that in the adiabatic representation

\[ e^{-iV\tau} = \begin{pmatrix} e^{-iE_1(x)\tau} & 0 \\ 0 & e^{-iE_2(x)\tau} \end{pmatrix}. \]  \hspace{1cm} (417)

Finally, show that the diagonal matrix introduced by Eq. (417) can be rotated to the representation of diabatic states $|1\rangle, |2\rangle$ according to the similarity transformation

\[ L^{-1} \begin{pmatrix} e^{-iE_1(x)\tau} & 0 \\ 0 & e^{-iE_2(x)\tau} \end{pmatrix} L. \]  \hspace{1cm} (418)

According to the definition of the eigenstates of the potential energy matrix, given by Eq. (34),

\[ \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} = \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix} \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix}, \]  \hspace{1cm} (419)

and

\[ \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \begin{pmatrix} L_{12} \\ L_{22} \end{pmatrix} = \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix} \begin{pmatrix} L_{12} \\ L_{22} \end{pmatrix}. \]  \hspace{1cm} (420)

Therefore,

\[ \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \begin{pmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix}, \]  \hspace{1cm} (421)

and

\[ \begin{pmatrix} L_{11} & L_{21} \\ L_{12} & L_{22} \end{pmatrix} \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \begin{pmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{pmatrix} = \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix}, \]  \hspace{1cm} (422)

or

\[ \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix} \begin{pmatrix} L_{11} & L_{21} \\ L_{12} & L_{22} \end{pmatrix}. \]  \hspace{1cm} (423)
Therefore, defining

\[ \mathbf{L} = \begin{pmatrix} L_{11} & L_{21} \\ L_{12} & L_{22} \end{pmatrix}, \]

(424)
gives

\[ \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} = \mathbf{L}^{-1} \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix} \mathbf{L}. \]

(425)

Exponentiating both sides of Eq. (425), gives

\[ e^{-i\tau} \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} = e^{-i\tau L^{-1}} \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix} \mathbf{L}. \]

(426)

Expanding the r.h.s. of Eq. (426) gives,

\[ e^{-i\tau} \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \mathbf{L}^{-1} \begin{pmatrix} -i\tau E_1 & 0 \\ 0 & -i\tau E_2 \end{pmatrix} \mathbf{L} + \mathbf{L}^{-1} \begin{pmatrix} \frac{1}{2!} E_1^2 (-i\tau)^2 & 0 \\ 0 & \frac{1}{2!} E_2^2 (-i\tau)^2 \end{pmatrix} \mathbf{L} + \ldots, \]

(427)
since \( \mathbf{L}^{-1} \mathbf{L} = 1 \). Therefore,

\[ e^{-i\tau} \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} = \mathbf{L}^{-1} \begin{pmatrix} e^{-iE_1(x)\tau} & 0 \\ 0 & e^{-iE_2(x)\tau} \end{pmatrix} \mathbf{L}. \]

(428)
33.12 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. (37). Propagate $|\Psi(x;t)\rangle = \varphi_1(x;0) + \varphi_2(x;0)$, where $\varphi_1(x;0) = \varphi_1(x;0) = \Psi_0(x)$ 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

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

(429)

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 $\Psi(x;t)$ according to the potential energy matrix introduced by Eq. (429), 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](http://ursula.chem.yale.yale.edu/~batista/classes/summer/P12/P12_c/pp_12)
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

180
PROGRAM Problem12

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)
      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

ccccccccccccc
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

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine ReadParam(nstep,ndump,dt)
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

181
INTEGER ntype,nstep,nrpt,ireport,ndump,nlit
REAL xmin,xmax,pk,amassx,xk,dt
common /packet/ amassx,xk,pk
common /xy/ xmin,xmax

xmin=-6.0
xmax=6.0
dt=0.2
amassx=1.0
xk=-2.2
pk=0.
nstep=100
ndump=1

return
end

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
Dump Time Evolved Wave packet

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, cl1, x, xmin, xmax, dx, EVVALUES, dt
PARAMETER (npts=9, nptx=2**npts, NN=2)
DIMENSION CRV(NN, NN), EVVALUES(NN)
DIMENSION psi(NN, NN)
common /xy/ xmin, xmax
COMMON /wfunc/ chi(nptx, NN)
COMMON /ENER/ energy(NN)

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

Save Wave-packet components

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 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

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
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
   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
      if(nx.ne.0) xsc=real(nx)/alenx
      tprop(kx)=exp(eye*(propfacx*xsc**2))
   end do
   c
   return
end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine SetPotProp(dt,vprop)

Potential Energy part of the Trotter Expansion: exp(-i V dt/2)

IMPLICIT NONE
INTEGER N,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)

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

RETURN
END

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
SUBROUTINE PROPAGATE(vprop,tprop)

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
Split Operator Fourier Transform Propagation Method

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)

Apply potential energy part of the Trotter Expansion

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

Fourier Transform wave-packet to the momentum representation

CALL fourn(chin1,nptx,1,1)
CALL fourn(chin2,nptx,1,1)

Apply kinetic energy part of the Trotter Expansion

DO ii=1,nptx
  in=ii
  kk=ii
  chin1(in)=tprop(kk)*chin1(in)
  chin2(in)=tprop(kk)*chin2(in)
END DO

Inverse Fourier Transform wave-packet to the coordinate representation

CALL fourn(chin1,nptx,1,-1)
CALL fourn(chin2,nptx,1,-1)

Apply potential energy part of the Trotter Expansion

DO ii=1,nptx
  in=ii
  DO i=1,NN
    kk=ii
    chi(kk,i)=vprop(kk,i,1)*chin1(in)
  END DO
END DO
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
    RETURN
END

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

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
SUBROUTINE PE(RV)

c

 c Expectation Value of the Potential Energy

 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

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,ri,di,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)
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

SUBROUTINE SCHROC1(CRV,EVECT,EVALUES)
c
Hamiltonian Matrix Diagonalization
c
CRV: HERMITIAN MATRIX (INPUT)
EVECT: EIGENVECTORS (OUTPUT)
EVALUES: EIGENVALUES (OUTPUT)

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
END DO

189
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
190
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
      TEMPLR=SNGL(WR)*DATA(K2)-SNGL(WI)*DATA(K2+1)
      TEMPI=SNGL(WR)*DATA(K2+1)+SNGL(WI)*DATA(K2)
      DATA(K2)=DATA(K1)-TEMPLR
      DATA(K2+1)=DATA(K1+1)-TEMPI
      DATA(K1)=DATA(K1)+TEMPLR
      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

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
Subroutines to compute eigenvalues and eigenvectors
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
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
      191
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)
   CONTINUE
IF(L.GT.J) THEN
   DO 14 K = J + 1, L
      G = G + A(K,J) * A(I,K)
   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)
   CONTINUE
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)
         CONTINUE
         DO 20 K = 1, L
            A(K,J) = A(K,J) - G * A(K,I)
         CONTINUE
      CONTINUE
      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

SUBROUTINE TQLI(D,E,N,NP,Z)
IMPLICIT NONE
INTEGER N,NP,I,K,L,M,ITER
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
14 CONTINUE
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

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

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
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.p, compile it by typing

```
f77 Problem12p.p -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
```

196
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
pause .1
plot "arch.0091" u 1:2 lw 3
pause .1
plot "arch.0092" u 1:2 lw 3
pause .1
plot "arch.0093" u 1:2 lw 3
pause .1
plot "arch.0094" u 1:2 lw 3
pause .1
plot "arch.0095" u 1:2 lw 3
pause .1
plot "arch.0096" u 1:2 lw 3
pause .1
plot "arch.0097" u 1:2 lw 3
pause .1
plot "arch.0098" u 1:2 lw 3
pause .1
plot "arch.0099" u 1:2 lw 3
pause .1
PROGRAM Problem12p

c
SOFT Surface Hopping (SOFT/SH) Method (Chen and Batista 2006)
1-D nonadiabatic wave-packet propagation

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
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
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

subroutine ReadParam(nstep, ndump, dt)

Parameters defining the grid (xmin, xmax), integration time step (dt), mass (amassx), initial position (xk), initial momentum (pk), number of propagation steps (nstep), and how often to save a pic (ndump)

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

xmin=-6.0
xmax=6.0
dt=0.2
amassx=1.0
xk=-2.2
pk=0.
nstep=100
ndump=1

return
end

SUBROUTINE Initialize(kt)

IMPLICIT NONE
INTEGER NN, nptx, npts, kk, counter, j, kt, ns
COMPLEX chi0, chi, EYE, CRV, cl
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 / iwfnc/ chi0(nptx, NN)
COMMON / cumul/ cl(nptx, 200, 2), counter(200)
COMMON / OCCUP/ ns

EYE=(0.0, 1.0)
pi= acos(-1.0)
omega=1.

202
dx = (xmax - xmin) / real(nptx)
ns = 1

Wave Packet Initialization: Gaussian centered at xk, with momentum pk

alpha = amassx * omega
do kk = 1, nptx
  x = xmin + kk * dx
  chi(kk, 1) = ((alpha / pi)**0.5) * exp(-alpha / 2 * (x - xk)**2 + EYE * pk * (x - xk))
  chi(kk, 2) = chi(kk, 1) * 0
  chi0(kk, 1) = chi(kk, 1)
  chi0(kk, 2) = chi(kk, 2)
end do

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

RETURN
END

SUBROUTINE SAVEWF(je2, ndump, dt)

Dump Time Evolved Wave packet

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, cia, 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)

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

Save Wave-packet components

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)

Save Adiabatic states

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

SUBROUTINE ACCUM(je2,ndump,dt)
Accumulate Time Evolved Wave packet

IMPLICIT NONE
INTEGER je2, nptx, npts, kk, NN, ncount, ndump, jj, counter, ns
COMPLEX chi, CRV, energy, psi, Psia, ci, c2
character*9 B
REAL V, x1, cia, x, xmin, xmax, dx, EVALUES, dt
PARAMETER(npts=9, nptx=2**npts, NN=2)
DIMENSION CRV(NN, NN), EVALUES(NN)
DIMENSION psi(NN, NN)
DIMENSION c1(nptx, 200), counter(200)
common /xy/ xmin, xmax
COMMOM /wfunc/ chi(nptx, NN)
COMMOM /ENER/ energy(NN)
COMMOM /OCCUP/ ns

jj=je2/ndump
counter(jj)=counter(jj)+1

Accumulate Wave-packet components

do kk=1, nptx
   c1(kk, jj, ns)=c1(kk, jj, ns)+chi(kk, ns)
end do
RETURN
END

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
xsc=0.
if(nx.ne.0) xsc=real(nx)/alenx
tprop(kx)=exp(eye*(propfacx*xsc**2))
end do

return
end

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

subroutine SetPotProp(dt,vprop)

c
potential energy part of the trotter expansion: exp(-i V dt/2)

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)

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

RETURN
END

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

SUBROUTINE PROPAGATE(vprop,tprop,dt)

c
soft surface hopping (soft/sh) method (chen and batista 2006)

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

eye=(0.0,1.0)

Stochastic Jump

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

Apply potential energy part of the Trotter Expansion

DO ii=1,nptx
   chin(ii)=vprop(ii,ns_n,ns_o)*chi(ii,ns_o)
END DO

Fourier Transform wave-packet to the momentum representation

CALL fourn(chin,nptx,1,1)

Apply kinetic energy part of the Trotter Expansion

DO ii=1,nptx
   chin(ii)=tprop(ii)*chin(ii)
END DO

Inverse Fourier Transform wave-packet to the coordinate representation

CALL fourn(chin,nptx,1,-1)
Apply potential energy part of the Trotter Expansion

DO ii=1,nptx
   chi(ii,ns_n)=rc*vprop(ii,ns_n,ns_o)*chin(ii)
END DO

END

SUBROUTINE HAMIL(CRV,x)

IMPLICIT NONE
INTEGER NN
REAL x,VPOT1,VPOT2
COMPLEX CRV
PARAMETER(NN=2)
DIMENSION CRV(NN,NN)

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

RETURN
END

SUBROUTINE VA(V,x)

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

SUBROUTINE VB(V,x1)

IMPLICIT NONE
REAL V,x1,x
x=x1
V=-0.5*x**2+1.0/(16.0*1.3544)*x**4
SUBROUTINE PE(RV)

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

SUBROUTINE KE(RKE)

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 = acos(-1.0)
dx=(xmax-xmin)/nptx
dp=2.*pi/(xmax-xmin)

DO j=1,NN
   RKE(j)=0.0
END DO
do kk=1,nptx
    chic(kk)=chi(kk,j)
end do
CALL fourn(chic,nptx,1,1)
do kk=1,nptx
    if(kk.le.(nptx/2+1)) then
        nx=kk-1
    else
        nx=kk-1-nptx
    end if
    if(nx.ne.0) then
        p=real(nx)*dp
        chic(kk)=p**2/(2.0*amassx)*chic(kk)/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

SUBROUTINE SCHROC1(CRV,EVECT,EVALUES)
c
Hamiltonian Matrix Diagonalization
c
CRV: HERMITIAN MATRIX (INPUT)
EVECT: EIGENVECTORS (OUTPUT)
EVALUES: EIGENVALUES (OUTPUT)

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)

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)

DO I=1,N
    DO J=1,N
        EVECT(J,I)=CRV2(J,I)
    END DO
END DO
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=-2.0D0+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
      END
   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

Subroutines to compute eigenvalues and eigenvectors

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))
         END
      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
         END
      END
      F=A(I,L)
   END
18 CONTINUE
RETURN
END
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)
    CONTINUE
  IF(L.GT.J) THEN
    DO 14 K = J + 1, L
      G = G + A(K, J) * A(I, K)
    ENDIF
    CONTINUE
  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)
    CONTINUE
17 CONTINUE
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)
        CONTINUE
      DO 20 K = 1, L
        A(K, J) = A(K, J) - G * A(K, I)
        CONTINUE
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

SUBROUTINE TQLI(D,E,N,NP,Z)
IMPLICIT NONE
INTEGER N,NP,I,K,L,M,ITER
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.
      14 DO 15 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
        ENDIF
  15 CONTINUE
  DO 16 M=L,N
    16 CONTINUE
ENDIF
RETURN
END
\[ S = S \cdot C \]

ENDIF
\[ G = D(I+1) - P \]
\[ R = (D(I) - G) \cdot S + 2 \cdot C \cdot B \]
\[ P = S \cdot R \]
\[ D(I+1) = G + P \]
\[ G = C \cdot R - B \]
\[ DO \ 13 \ K = 1, N \]
\[ F = Z(K, I+1) \]
\[ Z(K, I+1) = S \cdot Z(K, I) + C \cdot F \]
\[ Z(K, I) = C \cdot Z(K, I) - S \cdot F \]
\[ 13 \ CONTINUE \]
\[ 14 \ CONTINUE \]
\[ D(L) = D(L) - P \]
\[ E(L) = G \]
\[ E(M) = 0. \]
\[ GO \ TO \ 1 \]
\[ 15 \ CONTINUE \]
\[ END \]

```
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
END
```

```
SUBROUTINE PIKSRT(N,ARR)
```

215
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
### 33.14 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](http://ursula.chem.yale.edu/~batista/classes/v572/dvrho1.f)

```fortran
PROGRAM DVR
  c
  c    This code computes the eigenvalues and eigenvectors of a Harmonic
  c    oscillator $V(x)=0.5 \cdot m \cdot \omega^2 \cdot (x-4.)^2$
  c    The KE matrix is described according to Eq. (81) of the lecture notes
  c
  CALL READPARAM()
  CALL Hamiltonian()
  CALL EIGV()
  CALL DUMP()
END

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

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=2.0d0/(i-ip)**2
      END IF
      VHAM(i,ip)=VTEMP*(-1)**(i-ip)
    END DO
  END DO
RETURN
END
```

217


```fortran
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

FUNCTION V(r)
implicit real *8(a-h,o-z)
V=0.5*1.*(r-4.)**2
RETURN
END

SUBROUTINE EIGV()
  c
c   Diagonalization
c  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

SUBROUTINE DUMP()
  c
  implicit real *8(a-h,o-z)
  REAL CRV, EVALUES,EVECT,E
```

218
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)

DO k=1, 10
   IND=npt2-(k-1)
   PRINT *, "E(",k,")=",EVALUES(IND)
END DO

d DIMENSION r(NC),j(NC)

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)
      rsum=rsum + EVECT(i,IND)**2
   END DO
   PRINT *, "norm(",l,")=",rsum
   END DO
CLOSE(10)
22 FORMAT(6(e13.6,2x))
RETURN
END

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

Subroutines to compute eigenvalues and eigenvectors from NR

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
               12 CONTINUE
            END IF
         END IF
       END DO
   END IF
RETURN
END

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
\[ H = H + A(I,K)^2 \]

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)
   CONTINUE
   IF (L.GT.J) THEN
      DO 14 K = J + 1, L
         G = G + A(K,J) * A(I,K)
      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)
   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)
         CONTINUE
      CONTINUE
      DO 20 K = 1, L
         A(K,J) = A(K,J) - G * A(K,I)
      CONTINUE
   ENDIF
23 CONTINUE

220
SUBROUTINE TQLI(D,E,N,NP,Z)
IMPLICIT NONE
INTEGER N,NP,I,K,L,M,ITER
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
      IF (DD.EQ.DD) GO TO 2
      DD=ABS(D(M))+ABS(D(M+1))
   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
            ENDIF
         14 CONTINUE
         M=L
      ELSE
         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
            ENDIF
         14 CONTINUE
         M=L
      ENDIF
   15 CONTINUE
1 CONTINUE
RETURN
END
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
ENDIF
GO TO 1
ENDIF
RETURN
END

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
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

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](http://ursula.chem.yale.edu/~batista/classes/v572/scr_ho1)

```
set yrange[0:6]
set xrange[-10:10]
set dat sty 1
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.
```
33.15 Problem 15.2

Computational Problem 15.2:
Change the potential of the code written in 15.1 to that of a Morse oscillator \( V(\hat{x}) = D_e(1 - \exp(-a(\hat{x} - x_e)))^2 \), with \( x_e = 0 \), \( D_e = 8 \), 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](http://ursula.chem.yale.edu/~batista/classes/v572/dvrmo1.f)

```fortran
PROGRAM DVR

c This code computes the eigenvalues and eigenvectors of a Morse oscillator \( V(r) = D_e \cdot (1 - \exp(-a \cdot (r - r_e)))^2 \)

c The KE matrix is described according to Eq. (82) of the lecture notes


c
CALL READPARAM()
CALL Hamiltonian()
CALL EIGV()
CALL DUMP()
END

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

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=2.0d0/(i-ip)**2
         VTEMP=VTEMP-2.0d0/(i+ip)**2 ! for radial coord.
      END IF
      VHAM(i,ip)=VTEMP*(-1)**(i-ip)
   END DO
END DO
RETURN
END
```

225
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

FUNCTION V(r)
  implicit real *8(a-h,o-z)
  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

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
SUBROUTINE DUMP()

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)

DO k=1, 10
    IND=npt2-(k-1)
    PRINT *, "E(",k,")=",EVALUES(IND)
END DO

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

Subroutines to compute eigenvalues and eigenvectors from NR

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
        END IF
18     CONTINUE
    DO 19 J=1,N
        D(J)=E(J)
        DO 17 K=1,J-1
            E(K)=E(K)+D(J)*EVECT(K+1,J)
            F=EVECT(J,K+1)
            G=D(J)*EVECT(J,K)
            HH=(F-G)**2+(H)**2
            A(I,J)=0.
            A(J,I)=0.
            A(I,J)=A(J,I)
            A(I,J)=A(I,J)/HH
            D(J)=D(J)*HH
            E(J)=E(J)/HH
            E(K)=E(K)+F*G/H
        END DO
        E(J)=E(J)/SCALE
        D(J)=D(J)/SCALE
17     CONTINUE
    E(A(I,J))=E(J)
19     CONTINUE
    RETURN
END
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
  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)
      CONTINUE
    IF (L .GT. J) THEN
      DO 14 K=J+1, L
        G = G + A(K, J) * A(I, K)
      CONTINUE
    ENDIF
    E(J) = G / H
    F = F + E(J) * A(I, J)
  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)
      CONTINUE
    CONTINUE
  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
        CONTINUE
      ENDIF
  ENDIF
228
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

SUBROUTINE TQLI(D,E,N,NP,Z)
IMPLICIT NONE
INTEGER N,NP,I,K,L,M,ITER
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.)
ELSE
C = F / G
R = SQRT(C**2 + 1.)
ENDIF


```fortran
E(I+1)=F*R
S=1./R
C=C*S
ELSE
S=F/G
R=SQRT(S**2+1.)
E(I+1)=G*R
C=1./R
S=S*C
ENDIF
G=D(I+1)-P
R=(D(I)-G)*S+2.*C*B
P=S*R
D(I+1)=G+P
G=C*R-B
DO 13 K=1,N
  F=Z(K,I+1)
  Z(K,I+1)=S*Z(K,I)+C*F
  Z(K,I)=C*Z(K,I)-S*F
13 CONTINUE
14 CONTINUE
D(L)=D(L)-P
E(L)=G
E(M)=0.
GO TO 1
ENDIF
END

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
  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)
12 CONTINUE
ENDIF
END
```

230
V(J,K) = P
12    CONTINUE
    ENDF
13    CONTINUE
RETURN
END

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

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 dvrmol

and run it by typing

./dvrmol

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](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
### 33.16 Problem 15.3

**Computational Problem 15.3:**

Generalize the program developed in 15.1 to solve the 2-dimensional Harmonic oscillator $V(x,y) = \frac{1}{2} m \omega^2 (x^2 + y^2)$ 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](http://ursula.chem.yale.edu/~batista/classes/v572/dvrho2.f)

```fortran
PROGRAM DVR
    c
    c This code computes the eigenvalues and eigenvectors of the
    c 2 dimensional harmonic oscillator $V(x,y) = 0.5 \times (x-5)^2 + 0.5 \times (y-5)^2$
    c The KE matrix is described according to Eq. (82) of the lecture notes
    c
    CALL READPARAM()
    CALL Hamiltonian()
    CALL EIGV()
    CALL DUMP()
END

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
END

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
        j(1)=1+mod((i-1),npt)
        j(2)=1+abs((i-1)/npt)
        jp(1)=1+mod((ip-1),npt)
    END DO
END
```
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))
   /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

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

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 /page/ rmin(2), rmax(2), rmass(2), dx(2)
    PARAMETER(npt=20, npt2=npt*npt)
    COMMON/ HAM/ VHAM(npt2, npt2)
    PARAMETER(N=npt2, NF=N)
    COMMON/EIG/ EVALUES(npt2), EVECT(npt2, npt2)
DIMENSION E(NP)

DO I=1,N
    EVVALUES(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)
RETURN
END

SUBROUTINE DUMP()

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)

DO k=1, 10
    IND=npt2-(k-1)
    PRINT *, "E(",k,")=",EVALUES(IND)
END DO

DO l=1,4
    IND=npt2-(l-1)
    WRITE(B, '(A,i4.4)') 'wave.', l
    OPEN(10, FILE=B)
    DO i=1,npt2
        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
    CLOSE(10)
END DO
22 FORMAT(6(E13.6,2X))
RETURN
END

double precision function rhosc(r)
implicit real *8(a-h,o-z)
rhosc=0.5*1.*(r-5.)**2
return
end

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

Subroutines to compute eigenvalues and eigenvectors from NR
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

SUBROUTINE TRED22(A,N,NP,D,E)
IMPLICIT NONE
INTEGER I,J,K,L,N,NP
REAL A,D,E,H,SCALE,F,G,HH
DIMENSION A(NP,NP),D(NP),E(NP)
IF(N.GT.1)THEN
   DO 18 I=N,2,-1
      L=I-1
      H=0.
      SCALE=0.
      IF(L.GT.1)THEN
         DO 11 K=1,L
            SCALE=SCALE+ABS(A(I,K))
            11 CONTINUE
         IF(SCALE.EQ.0.)THEN
            E(I)=A(I,L)
            ELSE
               DO 12 K=1,L
                  A(I,K)=A(I,K)/SCALE
                  H=H+A(I,K)**2
                  12 CONTINUE
               F=A(I,L)
               G=-SIGN(SQRT(H),F)
               E(I)=SCALE*G
               H=H-F*G
               A(I,L)=F-G
               F=0.
               DO 15 J=1,L
                  A(J,I)=A(I,J)/H
                  G=0.
                  DO 13 K=1,J
                     G=G+A(J,K)*A(I,K)
                     13 CONTINUE
                  IF(L.GT.J)THEN
                     DO 14 K=J+1,L
                        G=G+A(K,J)*A(I,K)
                        14 CONTINUE
                     ENDIF
                  E(J)=G/H
                  F=F+E(J)*A(I,J)
                  15 CONTINUE
               HH=F/(H+H)
               DO 17 J=1,L
                  F=A(I,J)
                  17 CONTINUE
   END
18 CONTINUE
END

236
\begin{align*}
G &= E(J) - HH*F \\
E(J) &= G \\
\text{DO } 16 \text{ K}=1, J \\
A(J,K) &= A(J,K) - F*E(K) - G*A(I,K) \\
16 & \text{ CONTINUE} \\
17 & \text{ CONTINUE} \\
\text{ENDIF} \\
\text{ELSE} \\
E(I) &= A(I,L) \\
\text{ENDIF} \\
D(I) &= H \\
18 & \text{ CONTINUE} \\
\text{ENDIF} \\
D(1) &= 0. \\
E(1) &= 0. \\
\text{DO } 23 \text{ I}=1, N \\
L &= I-1 \\
\text{IF}(D(I).NE.0.) \text{THEN} \\
\text{DO } 21 \text{ J}=1, L \\
G &= 0. \\
\text{DO } 19 \text{ K}=1, L \\
G &= G + A(I,K)*A(K,J) \\
19 & \text{ CONTINUE} \\
\text{DO } 20 \text{ K}=1, L \\
A(K,J) &= A(K,J) - G*A(K,I) \\
20 & \text{ CONTINUE} \\
21 & \text{ CONTINUE} \\
\text{ENDIF} \\
D(I) &= A(I,I) \\
A(I,I) &= 1. \\
\text{IF}(L.GE.1) \text{THEN} \\
\text{DO } 22 \text{ J}=1, L \\
A(I,J) &= 0. \\
A(J,I) &= 0. \\
22 & \text{ CONTINUE} \\
23 & \text{ CONTINUE} \\
\text{RETURN} \\
\end{align*}
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
15   CONTINUE
END
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

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

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](http://ursula.chem.yale.edu/~batista/classes/v572/scr.2)

```
set dat sty 1
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.
```

240
33.17 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

\[ V(\hat{x}, \hat{y}) = D_e \left( 1 - \exp(-a(\hat{x} - x_e)) \right)^2 + D_e \left( 1 - \exp(-a(y - y_e)) \right)^2 \]

with \( x_e = 0 \), \( D_e = 8 \), 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](http://ursula.chem.yale.edu/~batista/classes/v572/dvrmo2.f)
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
RETURN
END
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
SUBROUTINE EIGV()
c
  Diagonalization
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,NF=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)
RETURN
END

SUBROUTINE DUMP()

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)

DO k=1,10
    IND=npt2-(k-1)
    PRINT *, "E(",k, ")","EVALUES(IND)
END DO

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
    CLOSE(10)
END DO
22 FORMAT(6(e13.6,2x))
RETURN
END

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

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

Subroutines to compute eigenvalues and eigenvectors from NR

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

SUBROUTINE TRED22(A,N,NP,D,E)
IMPLICIT NONE
INTEGER I,J,K,L,N,NP
REAL A,D,E,H,SCALE,F,G,HH
DIMENSION A(NP,NP),D(NP),E(NP)
IF(N.GT.1)THEN
  DO 18 I=N,2,-1
    L=I-1
    H=0.
    SCALE=0.
    IF(L.GT.1)THEN
      DO 11 K=1,L
        SCALE=SCALE+ABS(A(I,K))
      11 CONTINUE
      IF(SCALE.EQ.0.)THEN
        E(I)=A(I,L)
      ELSE
        DO 12 K=1,L
          A(I,K)=A(I,K)/SCALE
          H=H+A(I,K)**2
        12 CONTINUE
        F=A(I,L)
        G=-SIGN(SQRT(H),F)
        E(I)=SCALE*G
        H=H-F*G
        A(I,L)=F-G
        F=0.
        DO 15 J=1,L
          A(J,I)=A(I,J)/H
          G=0.
          DO 13 K=J+1,L
            G=G+A(K,J)*A(I,K)
          13 CONTINUE
        15 CONTINUE
        HH=F/(H+H)

244
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)
   CONTINUE
16 CONTINUE
17 CONTINUE
ENDIF
ELSE
   E(I)=A(I,L)
ENDIF
D(I)=H
18 CONTINUE
ENDIF
D(I)=A(I,I)
A(I,I)=1.
IF(L.GE.1) THEN
   DO 21 J=1,L
      A(I,J)=0.
      A(J,I)=0.
   CONTINUE
21 CONTINUE
END
23 CONTINUE
RETURN
END

SUBROUTINE TQLI(D,E,N,NP,Z)
IMPLICIT NONE
INTEGER N,NP,I,K,L,M,ITER
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
  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
  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
ENDIF
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

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

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)

```bash
set dat sty 1
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.
```

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

\[ V(x) = -0.5x^2 + 1.0/(16.0 \times 1.3544)x^4, \]  

for the initial state

\[ \Phi_0(x) = \pi^{-1/4} e^{-0.5(x-x_0)^2}, \]

with \( x_0 = -2.1 \).

16.1: Propagate \( \Phi_0 \) for 1000 a.u., using a propagation step \( \tau = 0.1 \) a.u. and compute \( |\xi(t)|^2 \).

16.2: Propagate \( \Phi_0 \) for 1000 a.u. applying a sequence of 2-\( \pi \) pulses as described by Eq. (119) 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](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](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-\( \pi \) pulses applied can be visualized by downloading the script from [http://ursula.chem.yale.edu/~batista/classes/v572/pp_p](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](http://ursula.chem.yale.edu/~batista/classes/v572/pp_n) and typing

\[ gnuplot<pp_n \]

.
PROGRAM main

Coherent control of tunneling in a symmetric 1-dimensional double well by using a sequence of 2-pi pulses that repetitively affect the phase the wave packet component associated with the initial state relative to the other terms in the expansion

IMPLICIT NONE
INTEGER NN,igammax,nptx,ndump,NFLAG,istep,nstep,ii,i
REAL dt,p1g,p2g,p1l,p2l,rr,ra,re0
COMPLEX vprop,tpropl,vprop_e,tpropl_e,energy
PARAMETER(igammax=9,nptx=2**igammax,NN=2)
DIMENSION vprop(nptx,NN,NN),tpropl(nptx,NN),energy(NN)
DIMENSION vprop_e(nptx,NN,NN),tpropl_e(nptx)
COMMON/e00/re0

NFLAG=1
CALL ReadParam(nstep,ndump,dt)
call inithr()
CALL SetKinProp1(dt,tpropl)
CALL SetPotProp(dt,vprop)
CALL SetKinProp1_e(tpropl_e)
CALL SetPotProp_e(vprop_e)
DO istep=1,nstep+1
IF(istep.GE.1) CALL PROPAGATE(vprop,tpropl)
CALL ENERGY_s(vprop_e,tpropl_e,energy)
IF(istep.EQ.1) re0=energy(2)
IF(istep.GT.5000) NFLAG=0
IF(mod((istep-1),ndump).EQ.0) THEN
    CALL DUMPWF(istep,ndump,p1g,p2g,p1l,p2l,energy)
    WRITE(10,22) dt*istep*2.4189E-2,p1g
1       ,p1l,p2g,p2l,real(energy(1))
END IF
IF(((istep-1).GE.3050).AND.(NFLAG.EQ.1)) THEN
    CALL pulse()
END IF
IF(mod((istep-1),ndump).EQ.0) THEN
    PRINT *,"# steps", (istep-1)
    CALL DUMPWF(istep,ndump,p1g,p2g,p1l,p2l,energy)
END IF
END DO
22 FORMAT(6(e13.6,2x))
END

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine ReadParam(nstep,ndump,dt)
c
Parameters defining the grid (xmin, xmax), integration time step (dt), mass (mасс), initial position (xk), initial momentum (pk), 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

xmin = -10.0
xmax = 10.0
dt = .1
amassx = 1.0
xk = -2.1
pk = 0.0
nstep = 10000
ndump = 50

return
end

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
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)

Wave Packet Initialization: Gaussian centered at xk, with momentum pk

xk2 = xk
beta = .5
beta2 = .5
do kk = 1, nptx
    x = xmin + kk * dx
    chi(kk, 1) = ((2.0 * beta / pi)**0.25) * exp(-beta * (x - xk)**2 + EYE * pk * (x - xk))
    chi(kk, 2) = chi(kk, 1)
    chi0(kk, 1) = ((2.0 * beta2 / pi)**0.25) * exp(-beta2 * (x - xk2)**2 + EYE * pk * (x - xk2))
    chi0(kk, 2) = chi0(kk, 1)
end do

Hamiltonian Matrix CRV

251
do kk=1,nptx
  x=xmin+kk*dx
  CALL HAMIL(CRV,x)
  WRITE(11,22) x,real(CRV(1,1)),real(CRV(2,2))
  real(chi0(kk,1))
END DO
22 FORMAT(6(e13.6,2x))
RETURN
END

SUBROUTINE DUMPWF(istep,ndump,p1g,p2g,p1l,p2l,energy)

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

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
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
  write(1,33) x,re0
  write(2,33) x,real(chi(kk,1))+re0
  write(2,33) x,real(chi(kk,2))+re0
end do
write(2,33)

do kk=1,nptx
  x=xmin+kk*dx
  write(1,33) x,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))
end do
CLOSE(1)
CLOSE(2)
33 format(6(e13.6,2x))

c
RETURN
END

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
SUBROUTINE pulse()

c
  2 pi pulse

c
IMPLIED NONE
INTEGER je2,nptx,igammax,kk,NN,ncount,ndump
COMPLEX chi,pj2,chi0,EYE
REAL c1,c2,x,xmin,xmax,dx,p1,p2,p1,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

subroutine SetKinProp1(dt,tprop1)
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

subroutine SetPotProp(dt,vprop)
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)

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
do i=1,2
do j=1,2
  vp(i,j)=0.
do k=1,2
  vp(i,j)=vp(i,j)+psi(i,k)*dummy(k,j)
end do
end do
rsqnx=1.0/sqrt(1.0*nptx)
do i=1,2
do j=1,2
  kk=ii
  vprop(kk,i,j)=vp(i,j)*rsqnx
end do
end do

RETURN
END

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

subroutine SetKinProp1_e(tprop1)

 IMPLICIT NONE
 INTEGER nptx,kx,nx,igammax,NN
 REAL xsc,xmin,xmax,propfacx,amassx,kx,pi,alenx,dt,pk
 COMPLEX tpropl,eye

 Kinetic Energy part of the Trotter Expansion: exp(-i p^2 dt/(2 m))


parameter(igammax=9,nptx=2**igammax,NN=2)
DIMENSION tpropl(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
      tpropl(kx)=propfacx*xsc**2
  end do
c	n return
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

subroutine SetPotProp_e(vprop)
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
doi=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
end subroutine SetPotProp_e(vprop)
SUBROUTINE PROPAGATE(vprop,tprop1)
  
  c Split Operator Fourier Transform Propagation Method
  
  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 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 Fourier Transform wave-packet to the momentum representation
  c
CALL fourn(chin1,nptx,1,1)
CALL fourn(chin2,nptx,1,1)

Apply kinetic energy part of the Trotter Expansion

DO ii=1,nptx
  in=ii
  kk=ii
  chin1(in)=tprop1(kk)*chin1(in)
  chin2(in)=tprop1(kk)*chin2(in)
END DO

Inverse Fourier Transform wave-packet to the coordinate representation

CALL fourn(chin1,nptx,1,-1)
CALL fourn(chin2,nptx,1,-1)

Apply potential energy part of the Trotter Expansion

DO ii=1,nptx
  in=ii
  DO i=1,NN
    kk=ii
    chi(kk,i)=vprop(kk,i,1)*chin1(in)
    +vprop(kk,i,2)*chin2(in)
  END DO
END DO

SUBROUTINE ENERGY_s(vprop,tprop1,energy)

Split Operator Fourier Transform Propagation Method

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 tpropl(nptx),vprop(nptx,NN,NN)
COMMON / wfunc/ chi(nptx,NN)
common /xy/ xmin,xmax

Apply potential energy part of the Trotter Expansion

rsqnx=1.0/sqrt(1.0*nptx)
dx=(xmax-xmin)/real(nptx)

DO in=1,nptx
  chin1(in)=chi(in,1)
DO in=1,nptx
  chin1(in)=tprop1(in)*chin1(in)*rsqnx**2
  chin2(in)=tprop1(in)*chin2(in)*rsqnx**2
END DO

Inverse Fourier Transform wave-packet to the coordinate representation
CALL fourn(chin1,nptx,1,-1)
CALL fourn(chin2,nptx,1,-1)

DO in=1,nptx
  DO j=1,NN
    chin1(in)=chin1(in)+vprop(in,1,j)*chi(in,j)
    chin2(in)=chin2(in)+vprop(in,2,j)*chi(in,j)
  END DO
END DO

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

RETURN
END

SUBROUTINE HAMIL(CRV,x)

Hamiltonian Matrix

IMPLICIT NONE
INTEGER NN
REAL x,VPOT1,VPOT2
COMPLEX CRV
PARAMETER(NN=2)
DIMENSION CRV(NN,NN)

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

RETURN
END

SUBROUTINE EXCITEDA(V,x)
implicit none
REAL V,x
v=0.5*x*x
RETURN
END

SUBROUTINE EXCITEDB(V,x1)
implicit none
REAL V,x1,x
x=x1
if(abs(x).LE.(2.34)) x=2.34
V=-0.5*x**2+1.0/(16.0*1.3544)*x**4
RETURN
END

SUBROUTINE SCHROC1(CRV,EVECT,EVALUES)
implicit none
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)

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)

DO I = 1, N
    DO J = 1, N
        EVECT(J, I) = CRV2(J, I)
    END DO
END DO
RETURN
END

Subroutines from Numerical Recipes

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
14 CONTINUE
18 CONTINUE
END
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
      WTEMP=WR
      WR=WR*WPR-WI*WPI+WR
      WI=WI*WPR+WTEMP*WPI+WI
   16 CONTINUE
   IFP1=IFP2
   GO TO 2
ENDIF
NPREV=N*NPREV
18 CONTINUE
RETURN
END

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 17 J=L+1,N
            F=SNGL(A(I,J))
            G=SNGL(A(J,I))
            HH=(1.0D0-(F**2+G**2)**0.5)**2
            SCALE=SCALE-2.0D0*HH
            HH=6.0D0*HH*(F**2+G**2)**0.5
            DO 16 K=1,I-1
               A(K,J)=A(K,J)+H*SCALE
               A(K,I)=A(K,I)-H*SCALE
               A(I,K)=A(I,K)-H*SCALE
               A(J,K)=A(J,K)+H*SCALE
            16 CONTINUE
            A(J,I)=A(J,I)-2.0D0*H**2*SCALE
            A(I,J)=A(I,J)+2.0D0*H**2*SCALE
            DO 15 K=L+1,N
               A(J,K)=A(J,K)-H*SCALE
               A(K,J)=A(K,J)+H*SCALE
               A(K,I)=A(K,I)+H*SCALE
               A(I,K)=A(I,K)-H*SCALE
            15 CONTINUE
            DO 14 K=1,N
               A(K,J)=A(K,J)+H*SCALE
               A(K,I)=A(K,I)-H*SCALE
               A(I,K)=A(I,K)+H*SCALE
               A(J,K)=A(J,K)-H*SCALE
            14 CONTINUE
            DO 13 K=1,N
               A(K,J)=A(K,J)+H*SCALE
               A(K,I)=A(K,I)-H*SCALE
               A(I,K)=A(I,K)+H*SCALE
               A(J,K)=A(J,K)-H*SCALE
            13 CONTINUE
         17 CONTINUE
      18 CONTINUE
      RETURN
END
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

SUBROUTINE TQLI(D,E,N,NP,Z)
 IMPLICIT NONE
 INTEGER N,NP,I,K,L,M,ITER
 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
     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.
       264
     ELSE
       IF (ABS(E(M))+DD.EQ.DD) GO TO 2
     ENDIF
   15 CONTINUE
 ELSE
   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.
     264
   ENDIF
   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
D(L)=D(L)-P
E(L)=G
E(M)=0.
GO TO 1
END
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

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