VIBRATIONAL RELAXATION OF DIATOMIC MOLECULES IN SOLIDS AT LOW TEMPERATURES

LAWRENCE L. HALCOMB
AND
DENNIS J. DIESTLER

DEPARTMENT OF CHEMISTRY
PURDUE UNIVERSITY

WEST LAFAYETTE, INDIANA

VIBRATIONAL RELAXATION OF DIATOMIC MOLECULES

IN SOLIDS AT LOW TEMPERATURES

Lawrence L. Halcomb[*] and Dennis J. Diestler

Department of Chemistry

Purdue University

West Lafayette, Indiana 47907

Abstract

A miscroscopic dynamical treatment of chemical systems comprising both light particles that require a quantal description and heavy ones that may be described adequately by classical mechanics has recently been presented [J. Chem. Phys. $\underline{78}$, 2240 (1983)]. The application of this ''hemiquantal'' method to the specific problem of the vibrational relaxation of a diatomic molecule embedded in a one-dimensional lattice is presented. The vectorization of a CYBER 205 algorithm which integrates the 10^3-10^4 simultaneous ''hemiquantal'' differential equations is examined with comments on optimization. Results of the simulations are briefly discussed.

[*]David Ross Fellow

I. Introduction

A microscopic dynamical description of a chemical system composed of both light particles that require a quantal description and heavy ones that may be described adequately by classical mechanics has been proposed recently [J. Chem. Phys. 78, 2240 (1983)]. The description consists of a self-consistent set of ''hemiquantal'' equations (HQE) arrived at by taking a partial classical limit of Heisenberg's equations of motion for the system. In form, the HQE appear to consist of Heisenberg's equations for the light particles coupled to Hamilton's equations for the heavy particles. The coupling is self-consistent in that there is an instantaneous feedback between the light and heavy subsystems, with total energy and probability of presence of the quantal subsystem being conserved.

This paper will focus on the numerical solution of the HQE on the CYBER 205 for the special case of a diatomic molecule embedded in a cold, one-dimensional lattice. In Section II, we detail the model and specific form of the HQE, while the CYBER 205 algorithm and steps taken to optimize performance are included in Section III. Results of the simulations and some discussion of their physical significance are presented in Section IV.

II. Model and Equations of Motion

Figure 1 depicts the physical situation, i.e. a single diatomic molecule BC occupying a substitutional site in an otherwise pure one-dimensional lattice of atoms A; the end atoms of the lattice are assumed free. So that the normal modes of the lattice are known analytically, the mass of BC is taken to be equal to that of A. The heavy, classically behaving degrees of freedom are considered to be the displacements (u_i) of the lattice atoms, including the

86

center of mass of BC, from their equilibrium positions. The internal vibration (q) of BC is treated quantally and, for simplicity, as a harmonic, two-state system. We assume that only nearest-neighbor atoms interact with one another: A-A interactions are harmonic; A-B and A-C interactions are approximated by Morse potentials.

Under these conditions, the HQE take the form

$$\dot{c}_i(t) = -i\hbar^{-1}[\varepsilon_i c_i(t) + \sum_j V_{ij}(\{u_k(t)\})c_j(t)]$$

$$\dot{u}_i(t) = p_i(t)/m_A \tag{1}$$

$$\dot{p}_i(t) = -\frac{\partial}{\partial u_i}U(\{u_j(t)\}) + \sum_{jk} c_j^*(t)c_k(t)F_{ijk}(\{u_m(t)\}) \; .$$

Here c_i is the occupation probability amplitude for quantal state i; p_i is the momentum conjugate to u_i; U is the harmonic part of the potential, i.e.

$$U = \frac{k}{2} \{\sum_{i=1}^{n-2} (u_{i+1} - u_i)^2 + \sum_{i=n+1}^{N-1} (u_{i+1} - u_i)^2\}, \tag{2}$$

where N is the number of lattice atoms. F is the quantal force defined by

$$F_{ijk} = \partial V_{ij}/\partial u_k \tag{3}$$

where

$$V_{ij}(\{u_k\}) = \langle i|V_{AB} + V_{AC}|j\rangle \ , \tag{4}$$

and the Morse potential V_{AB} is explicitly

$$V_{AB} = D_{AB}\{exp[-a_{AB}(u_n-u_{n-1} + L - \gamma_B q)]-1\}^2 \tag{5}$$

with a similar expression for V_{AC}.

Since the c_i are complex, the HQE consist of 2N+4 coupled first-order ordinary differential equations. Given initial conditions appropriate to the physical situation, we can integrate these numerically by standard techniques. Our principal problem now is to develop and optimize an algorithm appropriate to the CYBER 205.

III. CYBER 205 Algorithm

The HQE [Eqs. (1)] can be cast in terms of the vector differential equation $\dot{X} = f(X(t))$, defined by

$$\dot{x}_1(t) = f_1(x_1, \ \ldots, \ x_n), \ x_1(0) = x_1^o \ ,$$
$$\vdots \qquad \qquad \vdots \tag{6}$$
$$\dot{x}_n(t) = f_n(x_1, \ \ldots, \ x_n), \ x_n(0) = x_n^o \ .$$

The vector X can be written as

$X = [C,U,P]$ where, for example,

$$C = [C1, \ C2, \ C3, \ C4] \ . \tag{7}$$

88

From experience, we have found the HQE extremely well-behaved. Therefore, they can be handled with a relatively simple differential equation solver. We employ the familiar fourth-order Runge-Kutta algorithm (RK4) which, for our case, is summarized by the following equations:

$$\mathbf{K}_1 = T\ f(\mathbf{X})$$

$$\mathbf{K}_2 = T\ f(\mathbf{X} + \mathbf{K}_1/2)$$

$$\mathbf{K}_3 = T\ f(\mathbf{X} + \mathbf{K}_2/2) \qquad\qquad (8)$$

$$\mathbf{K}_4 = T\ f(\mathbf{X} + \mathbf{K}_3)$$

$$\mathbf{X}[(n+1)T] = \mathbf{X}(nT) + (\mathbf{K}_1+\mathbf{K}_4)/6 + (\mathbf{K}_2+\mathbf{K}_3)/3$$

where T is an appropriately chosen time step. Our choice of RK4 is guided by several considerations; it is quite stable, self-starting and easily coded for the CYBER 205. In addition, we need no direct method of estimating truncation error since we can calculate total energy and probability of the system as a check. Eventually, the RK4 algorithm will be used to calculate input values for a more sophisticated predictor-corrector routine.

Since our simulations require widely varying amounts of memory, we would like to assign storage at execution time. Clearly, the vector pipelines are used more efficiently if the entire derivative vector is manipulated at once. If we are to deal almost exclusively on the dynamic stack, we need a method of parsing the vector $\mathbf{X}$ into subvectors $\mathbf{C},\mathbf{U},\mathbf{P}$ which can then be handled independently. This ''breaking up'' is accomplished by building descriptors using SHIFT and OR operations on an integer equivalenced to a descriptor which points to an area in dynamic space. The subroutine BREAKUP is presented in the Appendix. This routine allows the RK4 mainline to allocate storage dynamically while permitting the derivative routine to access each subvector individually.

89

We now concentrate on the vector function subprogram that calculates the derivative $f(X)$. In our model, the four probability amplitudes must be accessed individually each time the function is called. Rather than waste a vector instruction to store the subvector C in a temporary array, it is faster and more convenient to use the following sequence of hardware calls to load them directly into registers:

```
ASSIGN TEMP, C

CALL Q8LOD (TEMP,, C1)

CALL Q8IX(TEMP, 64)

CALL Q8LOD(TEMP,, C2), etc.
```

The constants needed to calculate the potential and force functions are computed in advance and passed via labeled common. By reviewing an assembly listing of the program, one can minimize the number of loads necessary to access these constants. The evaluation of $\dot{U}$ is easily done by a vector multiplication with a stored reciprocal mass.

$\dot{P}$ can be conveniently calculated by evaluating the derivative of a fully harmonic potential U'. Thus we have

$$-\frac{\partial}{\partial u_i} U'(\{u_j\}) = k(-2u_i + u_{i-1} + u_{i+1}) \text{ where}$$

$$u_o = u_1 \, , \, u_{N+1} = u_N, \tag{9}$$

which can be effected by two vector additions and two vector multiplications as follows:

$$-\frac{\partial}{\partial U}U'(\{u_j\}) = UTEMP(1;N) = K*(-2.*UTEMP(1;N) + UTEMP(0;N) + UTEMP(2;N))$$

where UTEMP is a temporary array set to the current values of U. Finally, $\dot{P}$ is obtained by replacing the n-1, n, and n+1 elements of UTEMP by the proper values reflecting the Morse potentials at the diatomic. To accomplish this, it is necessary to access the five displacements $\{u_i, i = n-2, n+2\}$. Alternatively, descriptors could be built to define the necessary vectors on U and the values stored in UTEMP. In this case, hardware calls would be required to set the first and last elements of UTEMP, to access the five elements of U around u_n, and to store values in the three middle positions.

The conservation of total energy and probability gives us two necessary criteria to check the accuracy of the numerical solution. The total energy is given by

$$E = U(\{u_i\}) + P\cdot P/(2m_A)$$
$$+ |c_0|^2\varepsilon_0 + |c_1|^2\varepsilon_1 \tag{10}$$
$$+ |c_0|^2V_{oo} + 2Re\{c_0{}^*c_1\}V_{10} + |c_1|^2V_{11}$$

while total probability is simply

$$P = |c_0|^2 + |c_1|^2, \tag{11}$$

which must remain unity. These checks were made every 1000 iterations using values calculated in the first pass through the derivative routine. To calculate U' [Eq. (9)], the following code is used:

91

```
ASSIGN TEMP,.DYN. N-1

TEMP= Q8VDELT(U;TEMP)

EU= (K/2)* Q8SDOT(TEMP,TEMP).
```

In Table I, sample iteration times and estimates of floating point operations per second are given. The timings are for loops without I/O or accuracy checks. The results of several simulations are presented in the next Section.

IV. Results of Simulations

Our simulations all take the diatomic to be in its excited state and the lattice to be at OK initially. This means that all elements of $X(0)$ are zero, except the real component of $c_1(0)$, which is unity. The time step size is .01 ω^{-1}, where ω is the transition frequency of the diatomic. The quantity of principal interest here is $|c_1|^2$, the probability of the diatomic being excited. The physical constants for the system, which are chosen roughly to mimic HCl in Ar, are listed in Table 2. The only variable quantities are ω and N. The transition frequency is chosen low in order to observe relaxation on the time-scale of the simulation.

Figure 1 displays plots of $|c_1|^2$ versus time for a sampling of simulations. Frames (a)-(c) demonstrate the effect of increasing the diatomic's transition frequency ω, (cm^{-1}) holding the number of lattice atoms fixed. It appears that the rate of loss of energy from the diatomic increases with increasing frequency up to a point. In fact, frame (c) suggests that the diatomic evolves to a metastable state in which it loses no further energy. To test this hypothesis, we increased the number of lattice atoms to N = 2000. The result, shown in frame (f), bears this notion out. For purposes of comparison, we

include a simulation for a smaller lattice (N = 200). Here we see the effect of a pulse, which bounces back and forth, interfering with the monotonic relaxation of the diatomic.

V. Conclusion

These simulations represent the first application of a new description of the dynamics of chemical processes. Most previous approaches employ long-time asymptotic approximations, in which the coupling between the subsystems is weak and the decay is therefore very slow on the time scale of molecular motions $(10^{-14}s)$. The advancement of ultrafast laser spectroscopy now allows chemists to monitor directly fast relaxation processes $(10^{-12}s)$. In this regime, the coupling is more significant, and accurately solving the equations of motion becomes crucial. The HQE can be used for this purpose. However, any practical implementation will require a vector processor, such as the CYBER 205. Our calculations would be essentially impossible on Purdue University's 6500/6500/6600 system, for example. The calculations would take 50-100 times longer, even if the storage for the vectors were available.

The main feature of our CYBER 205 algorithm is a mainline that assigns storage at execution time. The vector function subprogram that evaluates the derivative can access the subvectors individually while the mainline processes the entire vector. This is accomplished by building the appropriate descriptors using the BREAKUP subroutine (see Appendix).

Some preliminary results were presented in Section IV. Future research will deal with the actual mechanism of energy exchange between the two subsystems. Also planned are some N-state models with applications in surface chemistry.

Acknowledgements

Table I. Increase of calculation speed with increase
of number of equations

Equations	Iteration Time	Mega FLOPS
24	.157 ms	6.1
204	.204 ms	22.8
804	.256 ms	37.9
2004	.671 ms	69.3
4004	1.19 ms	77.7
10004	2.75 ms	83.8
20003	5.37 ms	85.6

Table II. Parameters of model system

$$D_{AB} = 9.25 \times 10^{-15} \text{ ergs} \qquad D_{AC} = 1.24 \times 10^{-14} \text{ ergs}$$

$$a_{AB} = 1.83 \times 10^8 \text{ cm}^{-1} \qquad a_{AC} = 1.66 \times 10^8 \text{ cm}^{-1}$$

$$k = 814 \text{ ergs/cm}^2 \qquad m_A = 6.64 \times 10^{-23} \text{ g}$$

$$m_B = 1.67 \times 10^{-24} \text{ g} \qquad m_C = 5.88 \times 10^{-23} \text{ g}$$

Appendix

```fortran
      SUBROUTINE BREAKUP(X,NSUB,LENSUB,DESSUB,NDIM)
      IMPLICIT INTEGER(A-Z)
C
C     BREAKUP- TAKES A DESCRIPTOR (X) AND MANUFACTURES OTHER
C        DESCRIPTORS [DESSUB(N)] THAT POINT TO SUBVECTORS OF
C        LENGTHS LENSUB(N) WHICH COMPRISE THE VECTOR POINTED
C        TO BY X
C
C     ARGUMENTS:
C           X- DESCRIPTOR TO BE 'BROKEN UP'
C           NSUB- NUMBER OF SUBVECTORS
C           LENSUB- ARRAY CONTAINING THE SUBVECTOR LENGTHS
C           DESSUB- ARRAY CONTAINING THE RESULTING DESCRIPTORS
C           NDIM- DIMENSION OF LENSUB AND DESSUB
C
      DESCRIPTOR D,X,DESSUB(NDIM)
      DIMENSION LENSUB(NDIM)
      EQUIVALENCE (D,DTEMP)

      ASSIGN D,X
      ADD= SHIFT( SHIFT( DTEMP,16 ), -16)
      DO 100 N=1,NSUB
         LENGTH= SHIFT( LENSUB(N),48 )
         DTEMP= OR( ADD,LENGTH )
         ASSIGN DESSUB(N),D
         ADD= ADD + 64*LENSUB(N)
100      CONTINUE

      RETURN
      END
```
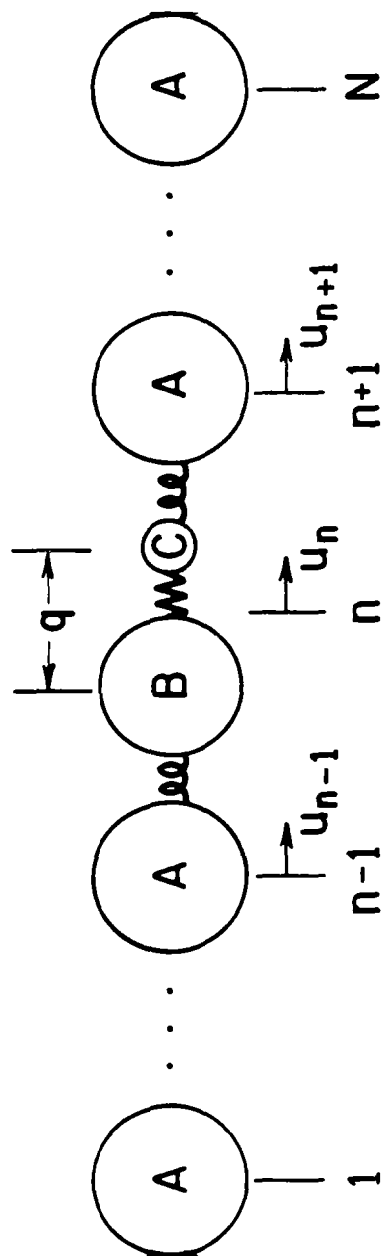
Figure 1. One-dimensional model of a substitutional diatomic molecule BC in an otherwise pure lattice of atoms A.
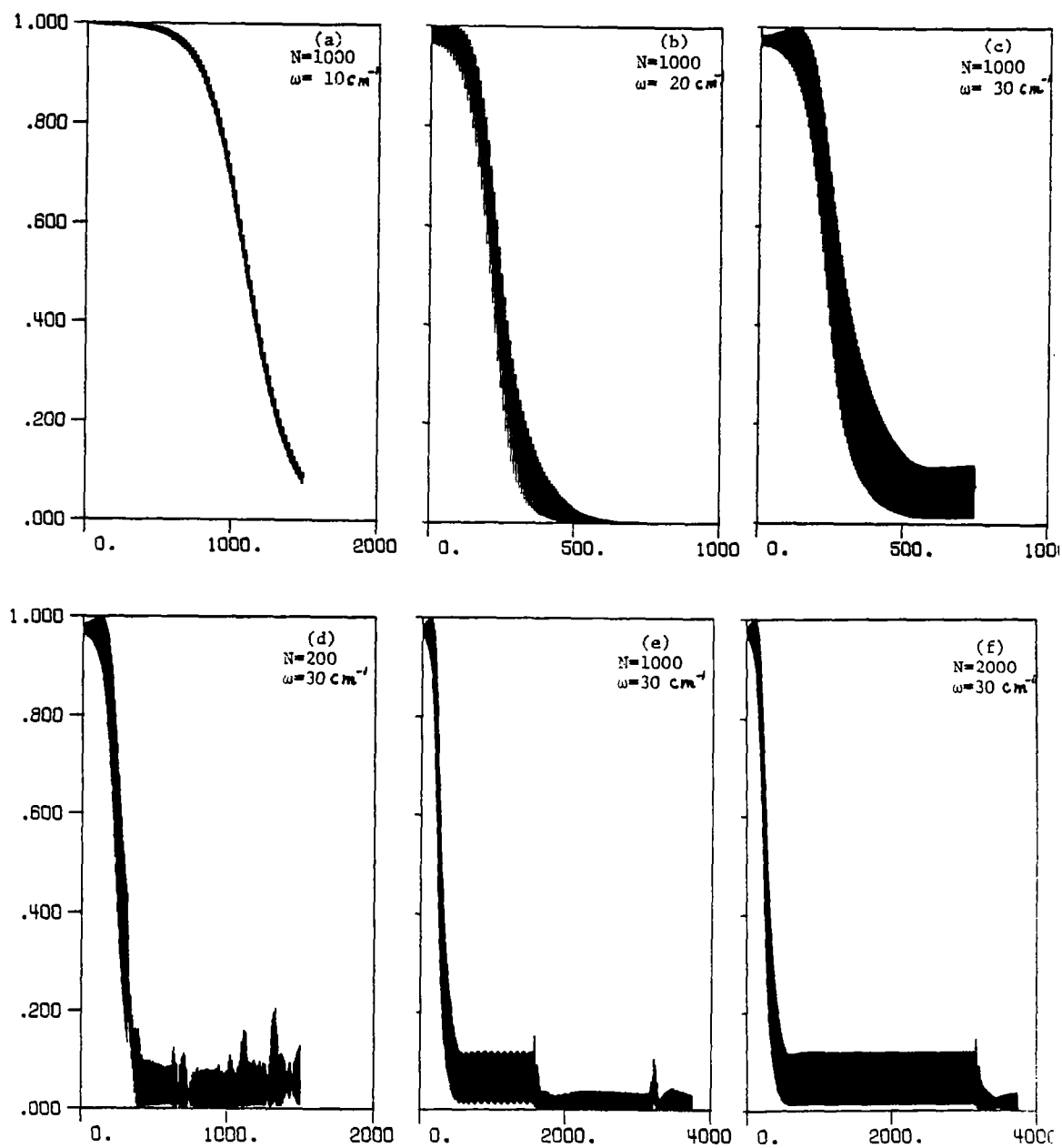
Figure 2. Plots of probability of finding diatomic in the excited
state versus time for a selection of simulations of the system defined
by parameters of Table 2. Time is in units of 0.18 picoseconds.

99