LBL-27795

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

Materials & Chemical Sciences Division

Studies of Simple Chemical Reactions in Solution: Iodine Photodissociation, Recombination, and Relaxation and the Development of Autoregressive Procedures for Application to Stochastic Simulations

D.E. Smith
(Ph.D. Thesis)

September 1989

LOAN COPY
Circulates
for 2 weeks
Bldg. 50 Library.
LBL-27795
Copy 2

Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098.

DISCLAIMER

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor the Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or the Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or the Regents of the University of California.

Studies of Simple Chemical Reactions in Solution: Iodine Photodissociation, Recombination, and Relaxation and the Development of Autoregressive Procedures for Application to Stochastic Simulations

by

David Eugene Smith

B.A. (Colorado College at Colorado Springs) 1984

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

CHEMISTRY

Materials and Chemical Sciences Division

Lawrence Berkeley Laboratory

1 Cyclotron Road

Berkeley, CA 94720

1989

Studies of Simple Chemical Reactions in Solution:
Iodine Photodissociation, Recombination, and Relaxation and
the Development of Autoregressive Procedures for Application
to Stochastic Simulations

by

David Eugene Smith

Abstract

Picosecond time-resolved experimental studies of the iodine photodissociation and recombination reaction were performed in order to clarify the early time dynamics of the dissociation and solvent caging processes. Results suggest that predissociation is much more rapid ($\leq$ 1 ps) than previously thought, and also indicate that geminate recombination may be rapid ($\leq$ 2 ps). Both of these conclusions imply that curve crossing in solution is extremely facile and not the rate-limiting step in recombination. An important consequence of this is that molecular dynamics simulations seem to give reasonable results for recombination dynamics even when curve-crossing is treated by oversimplified models. Finally, it is shown that in order to properly interpret the early time bleach kinetics it is necessary to consider dynamics within the ground state population remaining after excitation as well as excited state dynamics.

A simple theoretical model for vibrational relaxation of diatomic molecules in solution, based on the generalized Langevin equation (GLE), was also developed. In this model, the memory function in the GLE is determined directly from equilibrium force autocorrelation functions for the individual atoms of the diatomic dissolved in the

solvent of interest. In the process, autoregressive procedures for modeling of the memory function to arbitrary order were developed and found to provide for efficient numerical integration of the GLE. These powerful techniques have potential applicability to a variety of stochastic simulation methods. The vibrational relaxation model was tested on the system of iodine in liquid xenon, and through comparison with molecular dynamics simulations was shown to perform very well. Results are discussed in terms of simplifying assumptions that the solvent interaction with the diatomic can be characterized by equilibrium dynamics of single atoms in solution.

ACKNOWLEDGEMENTS

"And whatever you do, whether in word or deed, do it all in the name of the Lord Jesus, giving thanks to the Father through Him." (Colossians 3:17)

It is truly impossible to acknowledge or define the support I have received over the past five years while pursuing this degree. The following list is therefore not meant to be a complete acknowledgement, only a brief mention of the various people who in my estimation have contributed most significantly to this work.

Janine, my wife of over eight months now, has single-handedly made the past year of my time here at Berkeley the smoothest and most productive of all. She is responsible for, among other things, making sure I don't think about Chemistry all of the time! Thanks, Janine, for your unconditional friendship and your great love.

I would like in addition to thank my family. My parents gave me a wonderful home (where I still feel very welcome) and continual support and encouragement throughout all of my life. My sister Janet has inspired me more than she probably realizes, and I have really enjoyed getting to know her again as well as her husband Dave (Go Vikings!) these past few years since they moved to California. To acknowledge each and every one of the rest of my family would, alas, take too long, just know that you are all very special to me.

The Harris group has really been an outstanding place to work. I would first like to thank Alex Harris and Mark Berg for getting me started on the projects which are included in this thesis. Keenan

Brown has probably been the one person I have worked most closely with, especially the work presented in Chapters 3 and 4. Mark Paige and Dan Russell have also contributed significantly to my scientific development over the duration of my graduate career. To the entire Harris group I would just like to say that it has been a real pleasure working with all of you, eating lunch with most of you (where's Becca?), playing softball with some of you (come on, Karen!), and riding bikes with a few of you (Eric in particular), etc. Thanks for welcoming my wife as well this past year, she has felt really comfortable with all of you.

For the day to day workings of the Harris group I must certainly acknowledge our secretary Vijaya Narasimhan, without whose skill and dedication we would all be overwhelmed by administrative tasks.

Finally, I would like to thank Charles Harris for providing me with what I believe to be an ideal research environment, one which includes a group of talented and inspiring coworkers as well as the freedom to pursue problems of my own interest within the framework of his general research direction. This research environment has also been characterized by outstanding experimental and computational resources. This work was supported by the National Science Foundation. Molecular dynamics simulations were performed at the San Diego Supercomputer Center. In addition, I acknowledge the support of the US Department of Energy, Office of Basic Energy Sciences, Chemical Sciences Division under Contract No. DE-AC03-76SF00098 for some of the specialized equipment used in the experiments described in chapter 2.

TABLE OF CONTENTS

1. INTRODUCTION 1

References 5

2. DIRECT MEASUREMENT OF PREDISSOCIATION AND GEMINATE RECOMBINATION TIMES FOR IODINE IN SOLUTION 6

I. Introduction 6

II. Experimental 14

III. Predissociation Dynamics 17

IV. Direct Dissociation Dynamics 21

V. Ground State Dynamics 27

VI. Recombination Dynamics 36

VII. Conclusions 38

References 40

3. NUMERICAL INTEGRATION OF THE GENERALIZED LANGEVIN EQUATION THROUGH AUTOREGRESSIVE MODELING OF THE MEMORY FUNCTION 43

I. Introduction 43

II. Autoregressive Modeling of $\gamma(t)$ 47

A. Determination of AR parameters 48

B. Functional form for r_τ 50

III. Example: I/Xe 52

IV. Numerical Applications 59

A. Time Step Generalization 59

B. Evaluation of the Friction Integral 63

C. Simulation of I/Xe 66

V. Conclusions 69

Appendix 72

References 79

4. VIBRATIONAL RELAXATION OF DIATOMIC MOLECULES IN SOLUTION..... 81

I. Introduction..... 81

II. Model for Vibrational Relaxation..... 85

III. Molecular Dynamics Simulations..... 92

IV. Autoregressive Modeling of $\gamma(t)$..... 97

V. Results..... 107

VI. Discussion..... 113

References..... 119

Appendix. PROGRAM LISTINGS 121

I. Program Listing "IATOM"..... 121

II. Program Listing "CALCAR"..... 136

III. Sample Output "CALCAR"..... 147

IV. Program Listing "CALCARC"..... 149

V. Program Listing "I2MEXP"..... 151

VI. Sample Output "I2MEXP"..... 166

VII. Program Listing "MD3DKE"..... 169

Chapter 1: INTRODUCTION

The study of the molecular nature of chemical reactions in solution is a rapidly emerging field in physical chemistry, having been facilitated tremendously in the past fifteen years by two major technological advancements. First, the development of ultrafast laser systems, with picosecond and more recently femtosecond time resolution, have made it possible to directly measure condensed phase dynamics on time scales comparable to the molecular motions in the system. This is a crucial development in the study of molecular dynamics in solution since high resolution spectroscopic techniques, which in the gas phase provide a wealth of both static and dynamical information, are considerably more difficult to apply or interpret in the rapidly changing high-density environment of a liquid. Second, and of equal importance, has been the development of readily available, high-performance computer and supercomputer technologies. The methods of computer simulation have come to play a vital role in scientific advancement, providing a crucial bridge between theory and experiment. Simulations provide a testing ground for theoretical predictions, furnishing a well characterized "experimental" system in which every detail of the structure and dynamics of the system are potentially available for analysis.

An important reaction system in the study of chemical reactions in solution is the iodine photodissociation and recombination reaction. The history of this reaction system is an interesting topic in itself, and illustrates well both the complexities of studying in detail the

reaction pathway of even simple systems in solution and also the progress that has occurred in this area. This history is summarized briefly in Cha. 2 and in more detail in a recent review article.[1] The work in this thesis consists of both experimental (Cha. 2) and theoretical (Cha. 4) studies of the iodine reaction system.

The experimental studies presented in Cha. 2 are directed at clarifying early-time predissociation and geminate recombination dynamics of iodine in liquid CCl_4. This is accomplished through monitoring the excitation wavelength dependence of the reaction pathway. A 675 nm excitation pulse is used to initially place a population on the directly dissociative A state, with picosecond time-resolved transient absorption spectra being used to follow the subsequent dynamics. These results are compared with a similar set of experimental observations which followed excitation to the bound B state. In this latter case the initial excited state dynamics consist of relaxation within the bound state and/or predissociation (curve-crossing to a dissociative state). It has been suggested from these experiments that the time scale for predissociation is around 10-15 ps. The experiments in Cha. 2 strongly indicate that predissociation is instead quite rapid ($\leq$ 1 ps). This conclusion is shown to be consistent with the available data at each excitation wavelength, though only if <u>both</u> ground state dynamics (of the population remaining in the ground state after excitation) and excited state dynamics are considered. The experiments suggest in addition that geminate recombination is also quite fast ($\leq$ 2 ps). These results are discussed in terms of our understanding of curve-crossing phenomena in solution.

Finally, the validity of the typically oversimplified approaches to curve-crossing in molecular dynamics simulations is considered in light of the apparent rapid nature of such processes.

Theoretical studies of the iodine reaction system are presented in Cha. 4. The focus here is on the vibrational relaxation process in the ground state following recombination. A model for diatomic relaxation in simple liquids is presented and applied to iodine in liquid xenon, although its usefulness is not limited to this particular system. The iodine vibrational relaxation process has been an object of intense theoretical and experimental interest ever since it was identified as the rate-limiting step in the photodissociation/recombination reaction. Recent experiments and molecular dynamics simulations have been directed at increasing our understanding of this "simple" energy transfer process in liquid xenon, a monatomic solvent with only translational degrees of freedom. The wealth of information on this system has made it an ideal test case for theories of vibrational relaxation.

Chapter 4 consists of the application of a generalized brownian dynamics (GBD) model for vibrational relaxation to the iodine relaxation problem. In GBD models for chemical reactions, the multiple solvent degrees of freedom in a reactant/solvent system are modeled as a set of dissipative and random driving forces through the use of coupled generalized Langevin equations (GLE's). In the process, important features of the solute-solvent interactions can be identified as a function of the GLE-based solvent model used. The resulting simulation methods are potentially quite computationally efficient when

compared with traditional molecular dynamics (MD) simulations. Input for the GBD simulations of Cha. 4 comes from molecular dynamics (MD) simulations of single iodine atoms in Lennard-Jones xenon, at temperature and densities identical to those in a set of full scale MD simulations of the same vibrational relaxation process.[2] Results are compared with these MD simulations as well, providing a direct test of the applicability of GBD models to problems of vibrational relaxation.

In the process of developing the GBD model for vibrational relaxation of Cha. 4 it became clear that there was a need for a general method for integrating the GLE. Chapter 3 consists of the development of autoregressive (AR) techniques for applications to memory function modeling. AR techniques are commonly used in electrical engineering applications, and are extremely well developed in the literature, though not for the purpose of memory function modeling. These techniques provide a powerful framework for integration of the GLE which is both computationally efficient and sufficiently general to model the memory function of the GLE to arbitrary accuracy. For those readers who are less interested in the details of the numerical techniques, Cha. 4 contains a summary of important results from Cha. 3 and can be read alone.

The work presented in chapters 2-4 of this thesis also appear in the literature.[3-5]

References

1. A. L. Harris, J. K. Brown, and C. B. Harris, Ann. Rev. Phys. Chem. 39, 341 (1988).

2. J. K. Brown, C. B. Harris, and J. C. Tully, J. Chem. Phys. 89, 6687 (1988).

3. D. E. Smith and C. B. Harris, J. Chem. Phys. 87, 2709 (1987).

4. D. E. Smith and C. B. Harris, J. Chem. Phys. (submitted for publication).

5. D. E. Smith and C. B. Harris, J. Chem. Phys. (submitted for publication).

Chapter 2: DIRECT MEASUREMENT OF PREDISSOCIATION AND GEMINATE RECOMBINATION TIMES FOR IODINE IN SOLUTION

I. INTRODUCTION

The iodine photodissociation/recombination reaction has been one of the most extensively studied condensed phase chemical reactions. Interest in this system initially centered around the cage effect, the process whereby solvent molecules hold the dissociating iodine atoms in close proximity and allow them to recombine geminately. This effect was first discussed in terms of I_2 over fifty years ago by Rabinowitch and Wood,[1] yet it was not until the development of picosecond laser techniques in the last fifteen years that we have been able to time-resolve dynamics of the geminate recombination process.[2] Since that time, numerous experiments have provided a wealth of information not only on recombination dynamics, but also on solvent induced predissociation, electronic relaxation, and vibrational relaxation.[3-18] (For a recent review, see Ref. 18.) In fact, most of the recent experiments have been directed toward understanding relaxation processes subsequent to recombination, while our knowledge of rapid early time dissociation and caging dynamics remains relatively undeveloped. Experiments will be described in this chapter which clarify the time scales of these early time dynamics. Results will be discussed in terms of how the rates for these processes affect our understanding of this reaction and of condensed phase reactions in general.

Photodissociation of iodine is usually accomplished through excitation to the bound B state (Fig. 1) followed by rapid collision-induced predissociation. This solvent induced curve-crossing process is not at all well understood theoretically, with considerable uncertainty even over which repulsive states are involved.[19] Development of theories for liquid-phase predissociation would benefit significantly from experimental measurements of the rate for this process, and in fact some such information has been obtained from several picosecond experiments. It has been generally accepted that the time scale for predissociation of iodine in CCl_4 is 10-15 ps. A value in this range was first suggested by Chuang, Hoffman, and Eisenthal[2] on the basis of transient bleach kinetics and was corroborated in a later experiment at a different probe wavelength by Harris, Berg, and Harris[3]. These later results are shown in Fig. 2. In this experiment, an I_2/CCl_4 sample is excited at 590nm and a 500nm probe pulse is used to monitor the absorbance change of the solution as a function of time following excitation. Since ground state iodine absorbs strongly at the probe wavelength, one observes a decrease in absorbance following excitation, a bleach. The delay in the rise to the bleach maximum has been attributed to an absorption from the B state, with the decay rate for this absorption interpreted as the predissociation rate. In addition to bleach data, transient absorption measurements in the near UV have been used to more accurately determine this time scale.[3] A 10-15 ps lifetime for the B state, though certainly not unreasonable given our lack of knowledge concerning liquid-phase curve-crossing, is somewhat disturbing in light of gas-

Figure 1. Electronic potential surfaces of I_2 involved in absorption and recombination dynamics. Six states which correlate with ground state atoms as well as numerous higher states are not shown (Ref. 23). Vertical lines indicate absorptions at 590 and 675nm.

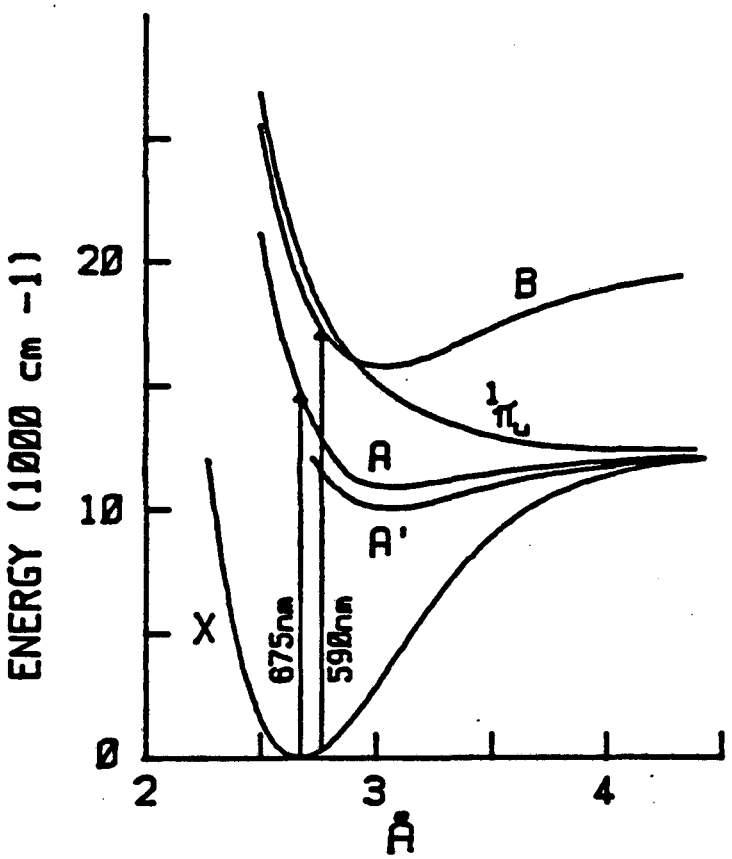

Figure 1.

Figure 2. Transient bleach at 500nm of I_2 in CCl_4 following 590nm excitation (from Ref. 3). 4 ps/pt.

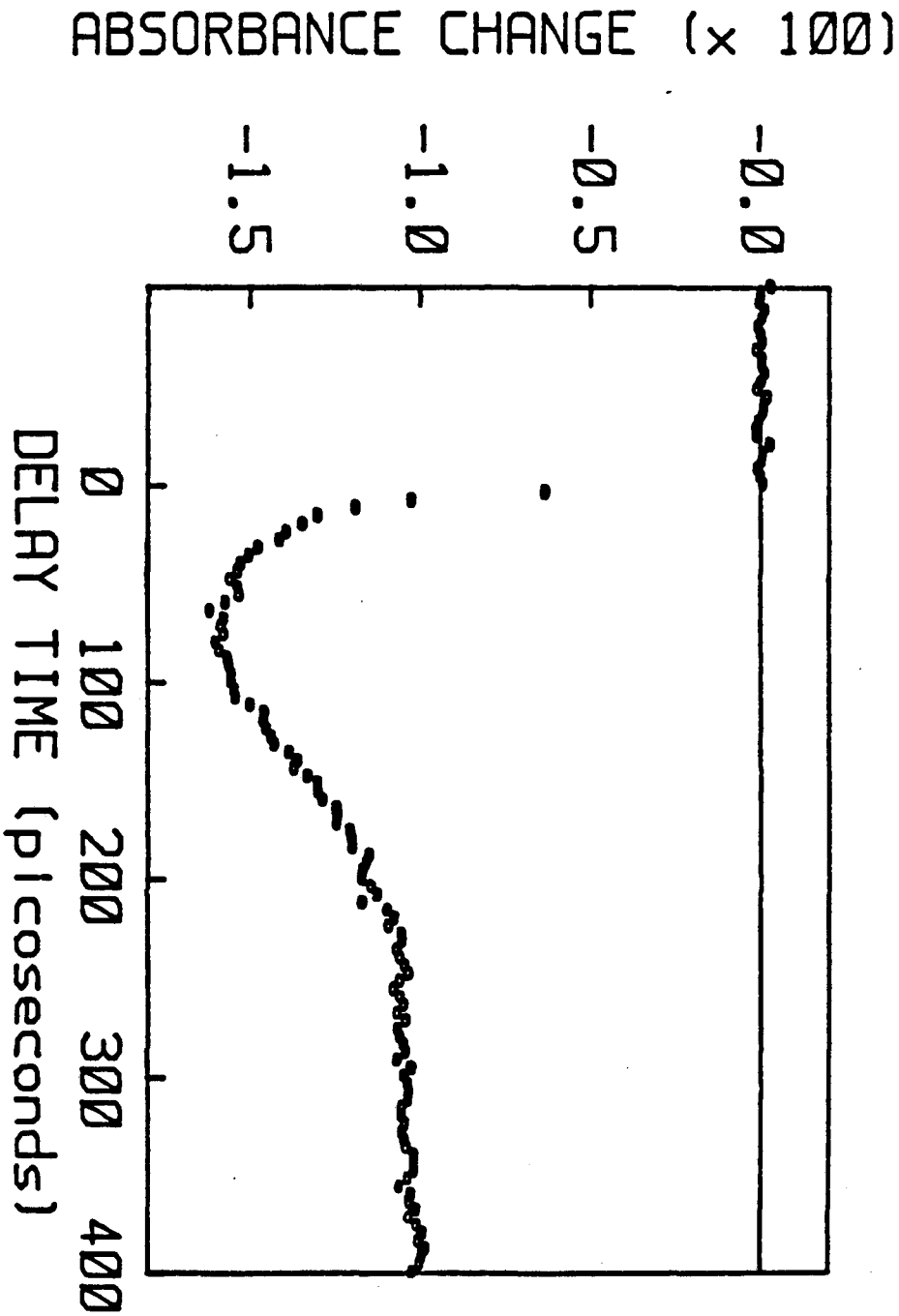
ABSORBANCE CHANGE (x 100)
-0.0
-0.5
-1.0
-1.5
0
100
200
300
400
DELAY TIME (picoseconds)

Figure 2.

phase measurements. In the gas phase, predissociation of iodine is found to occur within a few collisions with xenon,[20,21] a molecule similar in size and mass to CCl_4. In comparison, 10 ps corresponds to ~100 collisions in the liquid phase,[22] an order of magnitude difference. It would therefore be desirable to perform an experiment to confirm this apparent predissociation time scale. In this chapter, just such an experiment is described in which dissociation of iodine is accomplished directly, avoiding the predissociation process entirely. Visible excitation of molecular iodine can occur to three states; the bound B state, dissociative $^1\pi$ state, and the weakly bound A state (Fig. 1).[23,24] Direct dissociation is accomplished by exciting to a dissociative portion of the A state, well below the energy required to reach the B state. By comparing these results with previous experiments in which the molecules were excited to the B state, it becomes possible to isolate features which originate from the B state, and therefore clarify predissociation dynamics. A similar experiment with A state excitation was performed by Bado et al.[12], but did not give information about predissociation dynamics both because of the probe wavelengths used and because of complications resulting from high sample concentrations and repetition rates.[3,11] Additional experiments by Abul-Haj et al.[7,11] with A state excitation lacked the necessary time resolution to isolate rapid predissociation dynamics from subsequent relaxation processes. Results shown here are from experiments which combine ~1 ps time resolution and relevant probe wavelengths and which clearly imply that predissociation occurs in ≤ 1

ps, significantly faster than the 10-15 ps time scale previously reported.

In addition to information on predissociation these experiments also shed light on caging dynamics. There are several distinct steps in the recombination process. The dissociating atoms must transfer several thousand wave numbers of excess energy to the solvent, curve-cross to a bound state, and relax further to become trapped in that bound state. The solvent is responsible for holding the atoms in close proximity until the process is completed. Initially, it was thought that diffusive motion of the iodine atoms within a few solvent diameters of each other accounted for most of the 140 ps ground state recovery time in CCl_4.[2] Hydrodynamic models, which treat the solvent as a bulk rather than molecular liquid, were used to understand the results.[2,16,25-32] More recent experiments have indicated an upper bound of 15 ps for geminate recombination,[3] a time scale which suggests that the iodine atoms most likely remain in a common solvent cage until recombining. Such a rapid, isolated process should depend strongly on local solvent structure and dynamics, where hydrodynamic models are expected to break down. Molecular dynamics simulations, which explicitly treat the solvent as a molecular liquid, provide a more promising approach to understanding geminate recombination. However, our limited knowledge of curve-crossing in solution adds considerable uncertainty to these results. In particular, most simulations ignore the effect of multiple electronic surfaces in the recombination process, an effect which, it has been suggested, could slow recombination down by as much as a factor of 5.[33] The experiments

presented here will allow us to assign a time of ≤ 2 ps for geminate recombination of iodine in CCl_4, a result which agrees well with simulations in spite of their artificial treatment of curve-crossing. This suggests that curve-crossing in solution is extremely facile and that molecular dynamics simulations do not need to include "realistic" curve-crossing models in order to adequately predict recombination times.

The outline of the rest of this chapter is as follows. Section II contains a brief description of the experimental apparatus. Experimental results which have led to the assignment of a 10-15 ps predissociation time are discussed in more detail in Sec. III, followed by results of the present experiments in Sec. IV. In Sec. V the effect of dynamics within the ground state population following excitation are considered, as well as how these dynamics explain the early time features in the bleach region. In Sec. VI the implications of these experiments for our understanding of geminate recombination are discussed, followed by a summary and conclusions in Sec. VII.

II. EXPERIMENTAL

The experimental apparatus is diagramed in Fig. 3. The amplified synchronously-pumped dye laser system used here has been described previously.[3,34] It produces 1 ps, 590 nm pulses at a 10 Hz repetition rate with 0.8 mJ/pulse average energy. Each pulse is split by a 50% beamsplitter with one half being sent through a variable delay line and used to generate a probe continuum. The other 50% is focussed in a

Figure 3. Schematic diagram of the experimental apparatus. Excite and Probe paths are labeled. BS--beam splitter, DL--delay line, CG--continuum generation cell (H_2O), RG--Raman generation cell (D_6-acetone), BPF--band pass filter, SF--spatial filter, DC--dichroic (reflects 532nm, transmits 675nm), AC--amplification cell (DCM/DMSO ~5-10μM), F--filters, PD--photodiode, Ref.--reference sample, Sample--primary sample.

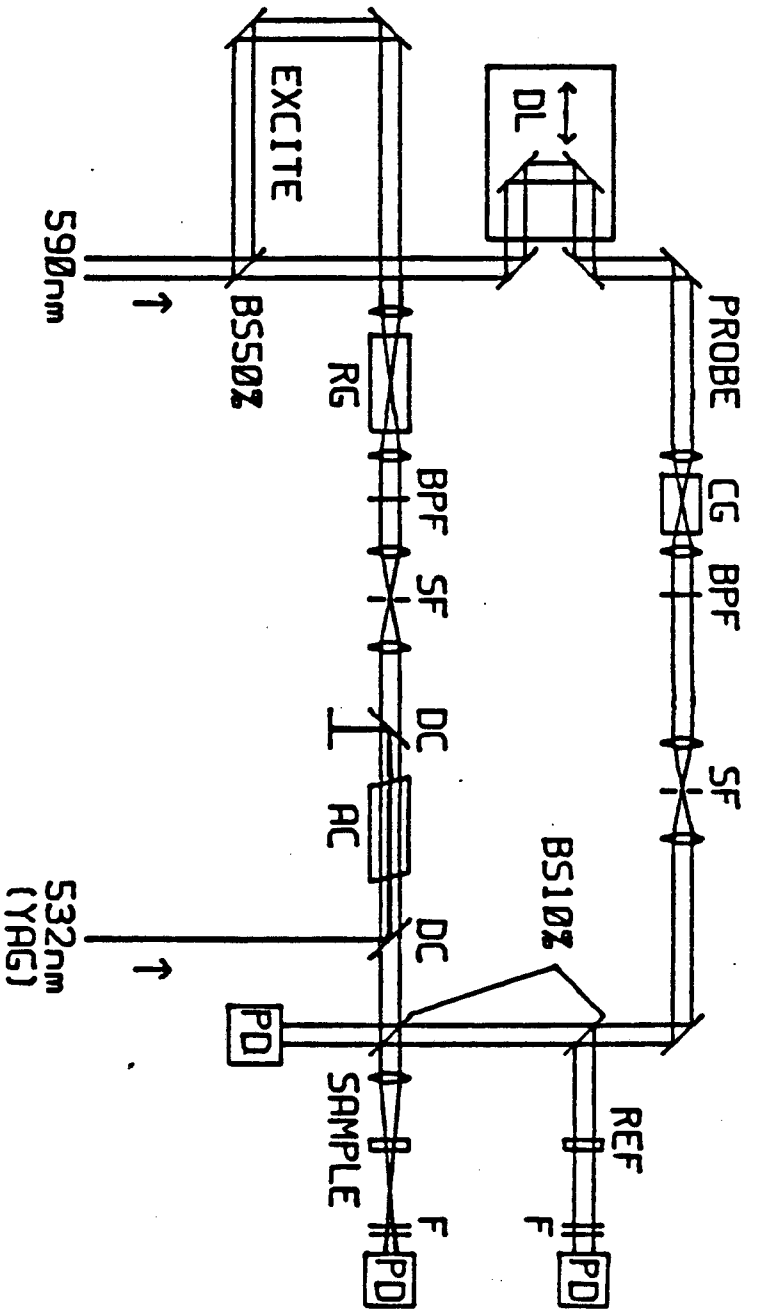
PROBE
CG
BPF
SF
BS10%
REF
F
PD
DL
EXCITE
590nm
BS50%
RG
BPF
SF
DC
AC
DC
532nm
(YAG)
PD
SAMPLE
F
PD

Figure 3.

cell of deuterated acetone, giving approximately 20% conversion to the first order Raman component at 675nm. This component is isolated with a band pass filter, spatially filtered to improve beam quality, and amplified in a 20cm dye cell containing DCM/DMSO (~5-10μM) using leftover 532nm light from the Nd:YAG laser. The resulting 675nm excite pulse is ~2ps long and has an average energy of .25 mJ/pulse. Excite and probe pulses are recombined and focussed in solutions of iodine in CCl_4 (~5mM). Excitation of iodine in the gas phase at 675nm results in excitation exclusively to the A state ~1000 wave numbers below the B state minimum and ~2000 wave numbers above the A state dissociation limit. Even given the 10nm width of the excite pulse and possible shift of the potentials in the liquid, excitation should be predominantly to the A state. The absorption strength of iodine at 675nm is ~1/5 of that at 590nm, which, along with the lower excitation energy, accounts for the smaller signal to noise ratio than in previous experiments. Absorbance as a function of delay time between excite and probe pulses was measured as reported previously.[3]

III. PREDISSOCIATION DYNAMICS

In this section previous experimental observations which led to the assignment of a 10-15 ps predissociation time are reviewed. All of these results are for I_2/CCl_4 excited with 590nm light, where ~85% of the excited state population is initially placed in the B state.

Two primary features in previous experiments are assigned to absorptions from the B state and are therefore thought to give

information on predissociation dynamics.[3] The transient bleach of I_2 at 500nm (Fig 2) rises in two components, a pulse width-limited rise characteristic of any bleach followed by a slow rise to its maximum ~50ps later. It has been suggested that the bleach maximum is delayed because of an absorption from the initially populated B state which would decay away as the B state predissociates. Second, there are strong absorptions in the 400-350nm region which rise and decay very rapidly (Fig 4). The rise at 400nm is pulse width-limited, or nearly so, and has therefore been assigned to the first state populated, the B state. The rise and decay times of the 400, 370, and 350nm absorptions are progressively longer, a feature which has been explained by vibrational relaxation within the B state. The decay time at 350nm gives a lower bound of 10 ps for predissociation since the B state lifetime must be at least as long as the longest observed absorption originating from the B state. Fig. 4 also shows a transient absorption measurement at 295nm which similarly consists of an early time absorption and a long time tail. The early time feature could conceivably be a continuation of the vibrational relaxation progression seen at 400-350nm. However, its decay time would require a predissociation time of > 20 ps which is in conflict with observed rise times of absorptions from states populated after recombination. These later rise times have been used to assign a 15 ps upper bound on predissociation, with the 295nm feature being left as yet unexplained. The experiments described in this chapter are designed to evaluate the assignment of the B state as the origin of the 500nm and near UV absorption features.

<u>Figure 4.</u> Transient UV absorptions of I_2 in CCl_4 (from Ref. 3). 400 and 350nm--0.4 ps/pt, 370nm--0.8 ps/pt, 295nm--1.1ps/pt.

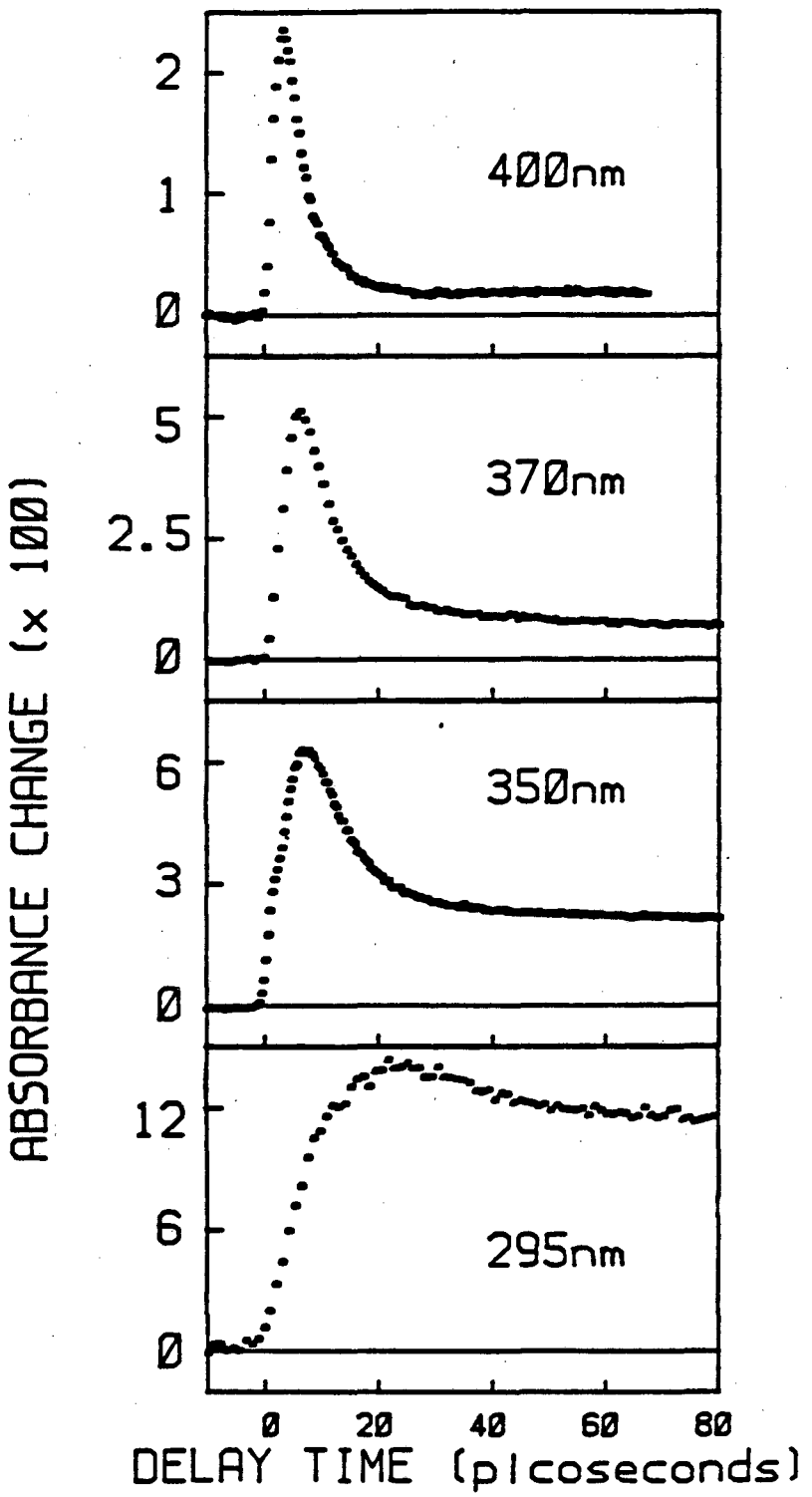
400nm
370nm
350nm
295nm
ABSORBANCE CHANGE (x 100)
DELAY TIME (picoseconds)
0
20
40
60
80
2
1
0
5
2.5
0
6
3
0
12
6
0

Figure 4.

IV. DIRECT DISSOCIATION DYNAMICS

In this section, results of the present experiments in which dissociation is accomplished directly are described. Excitation at 675nm is exclusively to a dissociative portion of the weakly bound A state. It would therefore be expected that any absorptions from the B state described in the previous section should disappear. There should be a pulse width-limited rise in the bleach at 500nm, and the early time absorptions in the near UV should not be observed.

Upon shifting of the excitation from the B state to the A state a significant change is observed in the 500nm transient bleach. Figure 5 shows results for 675nm excitation superimposed over the 590nm results from Fig. 2. The rise in the bleach is now entirely pulse width-limited, consistent with the previous assignment of a 500nm B state absorption, and apparently confirming predissociation dynamics discussed above. It is also interesting to note that beyond ~50 ps the bleach kinetics are independent of excite wavelength within the noise of the experiment. With excitation to the significantly bound A state (1600 cm-1 well depth) one might expect that recombination could occur preferentially to the A state if, for example, the dissociating atoms never reach a separation where curve-crossing is facile. This would be observed as an increase in the magnitude of the long time tail of the bleach feature, due to population trapped in the A/A' states, relative to the bleach maximum. The fact that no such increase is observed suggests that dissociation is indeed complete following 675nm excitation, and that subsequent partitioning dynamics are not strongly

Figure 5. Transient bleach at 500nm of I_2 in CCl_4, on two time scales. Results for 675nm excitation (points) are superimposed over results from Fig. 2 for 590nm excitation (solid lines). (a)--1.33 ps/pt, (b)--3.33 ps/pt.

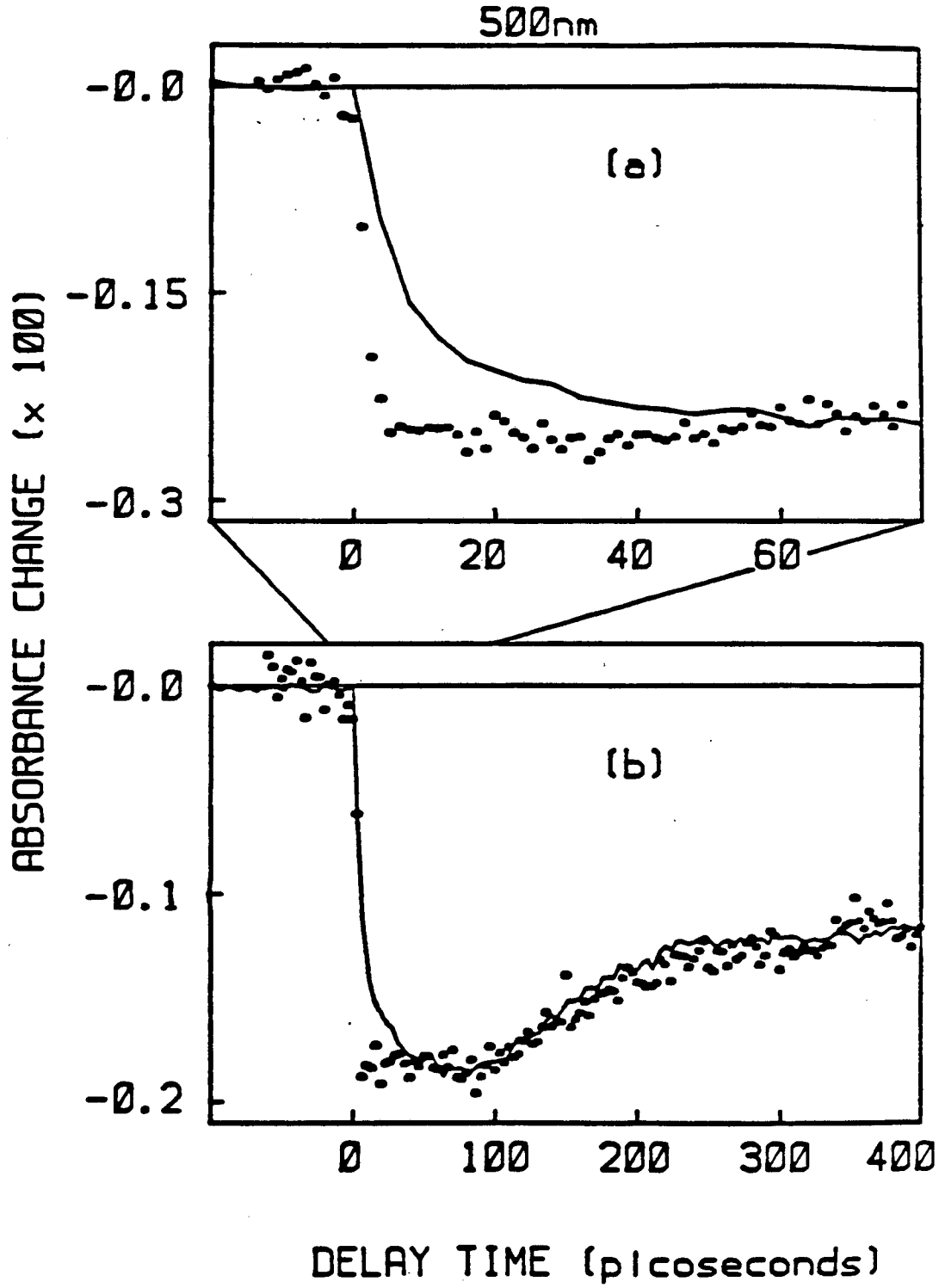
500nm
-0.0
-0.15
-0.3
(a)
0
20
40
60
ABSORBANCE CHANGE (x 100)
-0.0
-0.1
-0.2
(b)
0
100
200
300
400
DELAY TIME (picoseconds)

Figure 5.

affected by excitation pathway.

Although the bleach data seem consistent with previous expectations, surprising results in the near UV lead to a major reevaluation of these conclusions concerning predissociation. Figure 6 shows 400 and 350nm transient absorptions following 675nm excitation superimposed over results from Fig. 4. Note that all features are identical to within ~ 1 ps. The simplest explanation for such a marked similarity in these features is that predissociation is immediate (≤ 1 ps), and that after the first few picoseconds the excited state populations follow essentially the same trajectories independent of excitation pathway. In fact this will be the conclusion drawn following a brief consideration of two other possible explanations of this data, both of which allow for delayed predissociation.

If 590 and 675nm light actually accomplished 100% B state and A state excitation respectively one could conclude immediately that the early time transient absorptions in the near UV do not originate from the B state and that predissociation must be rapid. However, 590nm excitation is only ~85% to the B state, with the other 15% placed on directly dissociative A or $^1\pi$ states,[24] while 675nm excitation, though 100% to the A state in the gas phase, could conceivably place some small fraction of the iodine population in the B state due to potential shifts in solution. Therefore the possibility must be considered that 1) the UV absorptions are from the B state, from a small fraction of the total excited state population in the case of 675nm excitation, or 2) the UV absorptions are not from the B state but instead originate

Figure 6. Transient UV absorptions of I_2 in CCl_4. Results for 675nm excitation (points) are superimposed over results from Fig. 4 for 590nm excitation (solid lines). 400nm--0.4 ps/pt, 350nm--0.67 ps/pt.

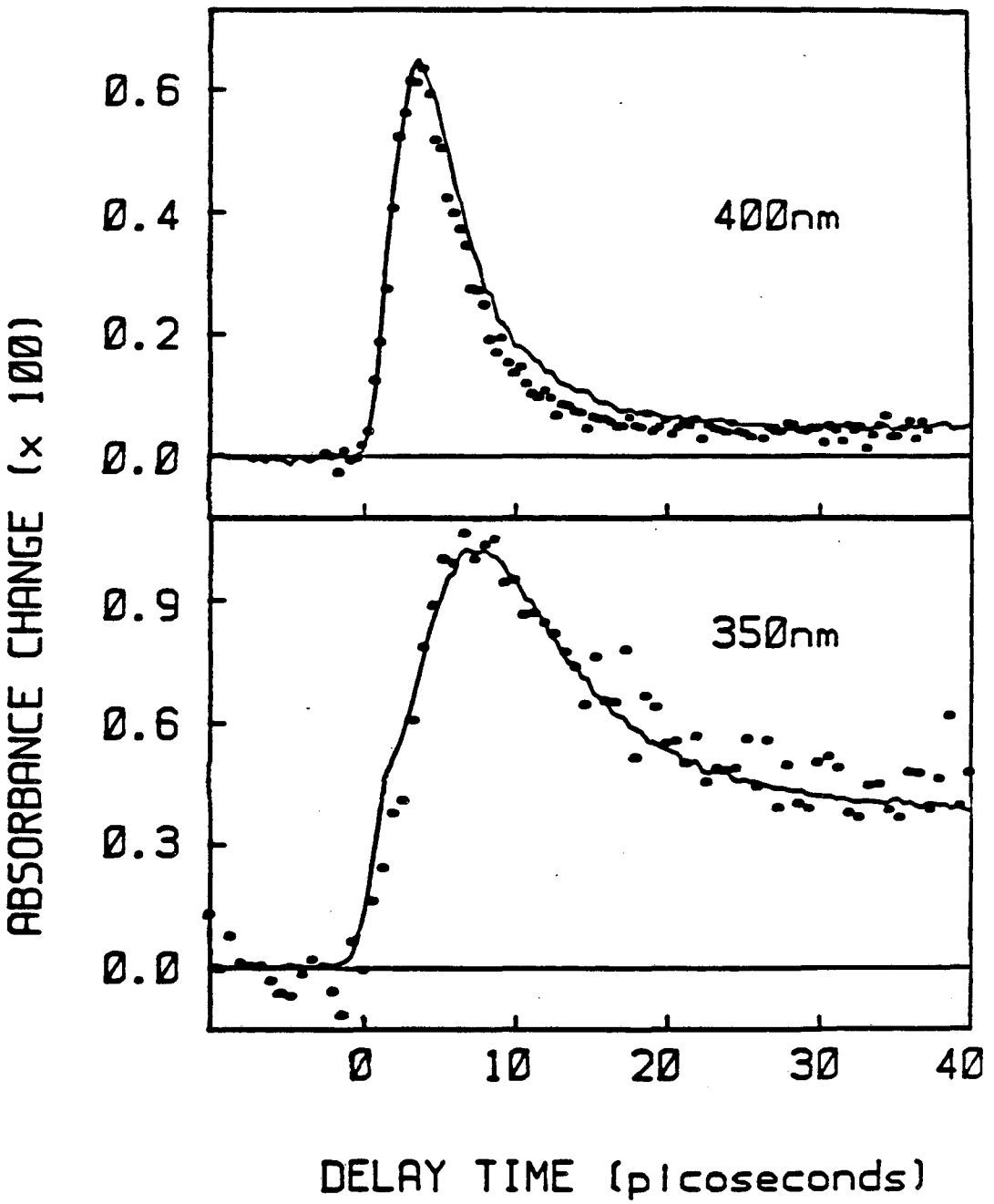
0.6
0.4
0.2
0.0
400nm
0.9
0.6
0.3
0.0
350nm
ABSORBANCE CHANGE (x 100)
0
10
20
30
40
DELAY TIME (picoseconds)

Figure 6.

from the molecules which dissociate directly, ~15% of the total excited state population in the 590nm case. Either of these possibilities would suggest order of magnitude differences in the strengths of the observed features depending on excitation pathway. After taking into account differences in ground state absorption strength and excitation intensities at the two wavelengths there is no apparent disparity in the observed strengths. If the first case were true the UV absorptions should disappear upon shifting the excitation further to the red, but in fact in another experiment not shown here the absorptions remain following 710nm excitation. If the second case were true, significant broadening should be observed in the early time features due to the predissociating B state population. The only adequate explanation for the excitation independent UV absorptions is that predissociation is actually very rapid ($\leq$ 1 ps), a result which, though reasonable given gas-phase predissociation data, is in direct conflict with the interpretation of the 500nm data presented above.

V. GROUND STATE DYNAMICS

Resolution of this conflict is accomplished by considering not only the excited states but also dynamics within the ground state population remaining after excitation. Absorption at 590nm is preferentially from v=1 and v=2 vibrational levels, which are significantly populated at room temperature. As a result, the ground state thermal distribution is significantly perturbed by the excite pulse. If reequilibration within this population occurs on a time scale slower than the

experimental resolution (~ 1 ps), this effect should be observed in the transient bleach kinetics. This section consists first of a qualitative discussion of the expected nature of the evidence for ground-state reequilibration, followed by a more quantitative determination of what should be observed at the excite and probe wavelengths used in various experiments.

Consider first the excite and probe wavelengths for the bleach experiment of Fig. 2. The 500nm probe light is absorbed primarily from v=0 in the ground state, while excitation at 590nm is preferentially from v=1 and v=2 (Fig. 7). Immediately following excitation but before any relaxation in the remaining ground state population a bleach will be observed at 500nm, both because some 500nm light is absorbed by v=1 and v=2 and because 590nm light excites some population from v=0. As the population reequilibrates there will be a net shift in population from v=0 to vibrational levels v=1 and v=2, resulting in an increase in the observed bleach. Therefore, from dynamics within the ground state population one would expect to see a pulse-limited rise in the bleach, followed by a delayed rise to the bleach maximum with ground state equilibration. This is exactly what is observed.

This effect can also account for a bleach which decays away in time. Take for example a hypothetical experiment in which excite and probe wavelengths are both 500nm. In this case, as the population reequilibrates there would be a net shift in population toward v=0, resulting in a net decrease in the bleach with time. A pulse-limited rise to the bleach maximum would be observed, followed by a decay to some equilibrium value.

Figure 7. Calculated I_2 room temperature absorption spectrum, including contributions from vibrational levels v=0 to v=3.

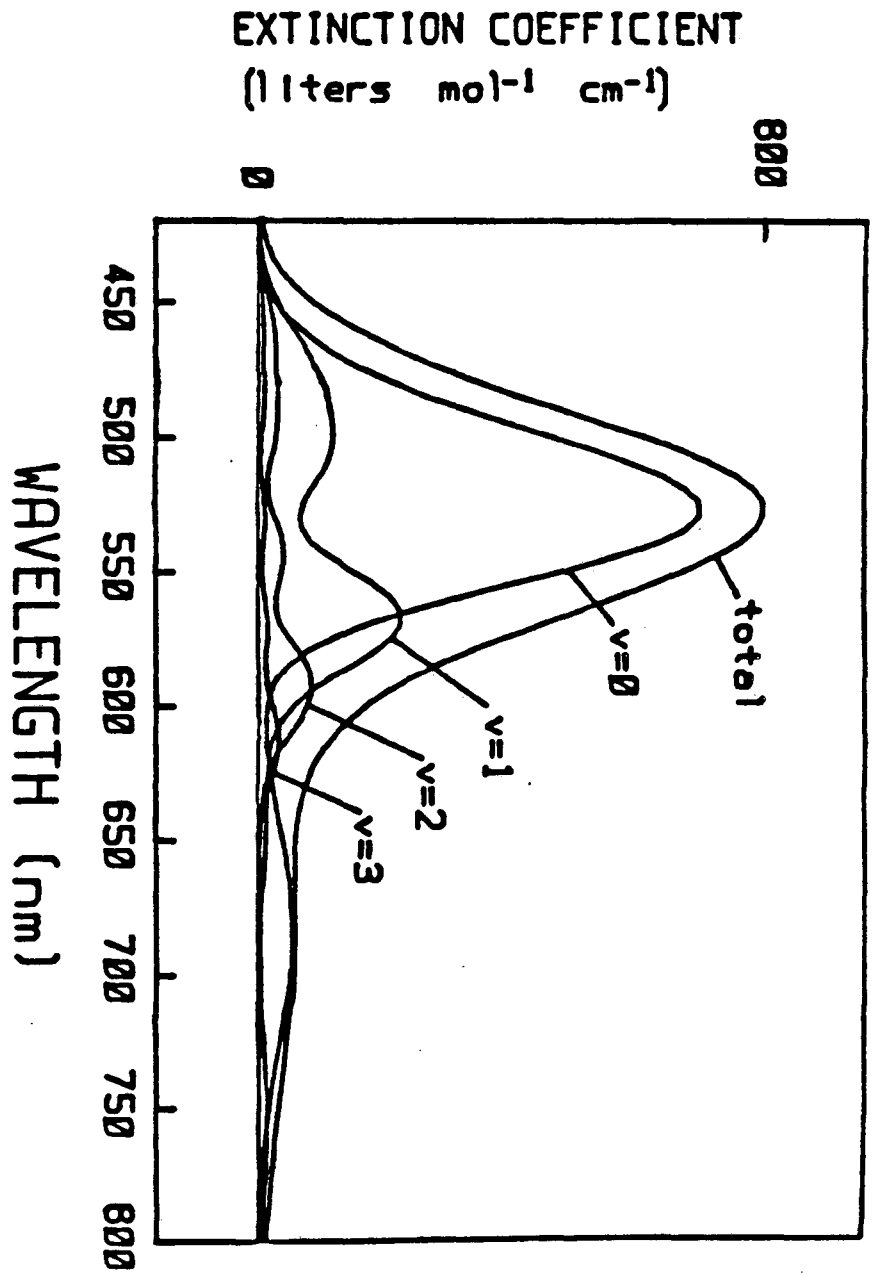
800
EXTINCTION COEFFICIENT
(liters mol-1 cm-1)
0
total
v=0
v=1
v=2
v=3
450
500
550
600
650
700
750
800
WAVELENGTH (nm)

Figure 7.

In order to predict the magnitude of this effect it is necessary to know transition strengths for each vibrational level at the excite and probe wavelengths of interest. Quantum mechanical transition strengths were calculated as described previously,[3] using approximate forms for potentials chosen to best fit known potentials in the 2.5-2.9 Å region where the ground state population is isolated. The calculated room temperature absorption spectrum, shown in Fig. 7 with contributions from vibrational levels v=0 to v=3, is in good agreement with the experimental gas-phase spectrum. The calculated transition strengths for v=0 through v=12 were used to determine the ratio of the bleach magnitude immediately following excitation to the bleach magnitude after ground state equilibration (ignoring the excited state population). While absolute bleach magnitudes are a function of excitation intensity and other experimental variables, this ratio is a function only of calculated transition strengths and can therefore be compared directly with experimentally determined ratios. Fig. 8 shows a plot of the calculated ratio versus excite wavelength for a 500nm probe. With 590nm excitation the ratio is 0.74, predicting an initial rise in the bleach with the laser pulse, followed by a slow rise to some final value, the ratio of the initial to final value being 0.74. Experimental comparison with this number is complicated by excited state dynamics which eventually cause a recovery in the absorption. A value of 0.66 ± 0.06 is found in CCl_4 if it is assumed that equilibration is complete when the bleach reaches its maximum. Similar results for other wavelengths are shown in Table I. As can be seen,

Figure 8. Calculated ratio of the initial bleach magnitude (immediately following excitation) to the bleach magnitude following ground state reequilibration. Ratio is plotted as a function of excite wavelength for a 500nm probe.

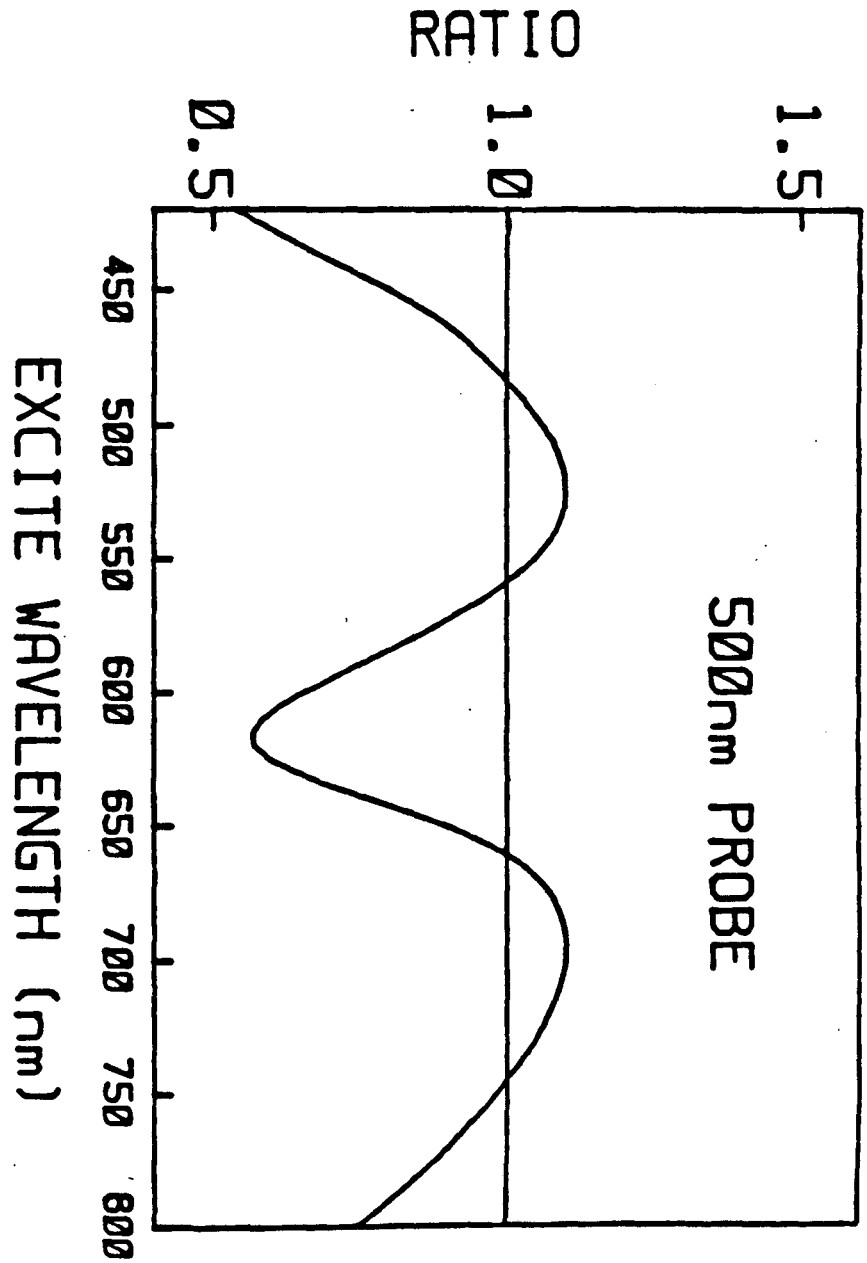
500nm PROBE
1.5
1.0
0.5
RATIO
450
500
550
600
650
700
750
800
EXCITE WAVELENGTH (nm)

Figure 8.

Table I. Ratio of initial bleach magnitude to equilibrium bleach magnitude: both experimental and calculated values for a 500nm probe and several excite wavelengths.

	Ratio	
Excite λ (nm)	Calculated	Experimental
570	0.92	1.00 ± .10[a]
590	0.74	0.66 ± .06
675	1.07	1.05 ± .10

[a] Unpublished

there is reasonable quantitative agreement between the experimental and calculated values, leading us to conclude that this effect can account for the observed early time bleach kinetics without any reference to excited state dynamics. The time scale for the rise in the bleach is equated with the relaxation time within the ground state. This suggests a value of ~20 ps for v=1 to v=0 relaxation, which is consistent with approximate vibrational relaxation data from previously observed decay curves.[3]

The significance of this calculation is in its prediction of a large excite wavelength dependence of the early time bleach dynamics, not in the quantitative predictions of actual ratios. This is because relatively small uncertainties in the potentials have a large effect on the predicted bleach kinetics. For example, with 590nm excitation, shifting the B state potential only 200 cm-1 to the red causes a change in the predicted ratio from 0.74 to 0.85. This magnitude of potential shift is consistent with observed spectra of iodine in various "non-interacting" solvents and therefore suggests that solvent effects should be important. In fact it has been observed experimentally that the ratio is different in different solvents in a way that is qualitatively consistent with the predicted trends. That is, the more blue shifted the absorption maximum is, the smaller the observed ratio of bleach magnitudes. This also explains why there was no delay in the rise to the bleach maximum in xenon, the only solvent which has not shown such a delay following 590nm excitation.[4] The iodine absorption is red shifted in xenon relative to the chloromethanes. In addition, the extremely slow ground state equilibration rate in xenon could allow

interference from the excited state population dynamics to obscure any effect from that equilibration which might otherwise be visible.

One piece of experimental data is in clear conflict with this interpretation of early time bleach kinetics. The initial picosecond iodine experiment by Chuang et al. showed a delayed rise in a 532nm bleach following 532nm excitation.[2] As discussed qualitatively above, ground state dynamics would be observed as a decay in the bleach in the case of identical excite and probe wavelengths, exactly the opposite of what they saw within the first 20 ps. The feature they observed peaks at ~20 ps while the bleach in Fig. 2 does not peak until > 50 ps after excitation. This could be an indication that these two features have entirely different origins. Perhaps their feature is an artifact resulting from a pulse width-limited time resolution of ~10 ps, with the observed rise time being only slightly slower than the expected experimental limit. Regardless, it has been clearly shown that ground state equilibration is an effect which should be observed and which adequately explains most observed early time bleach kinetics while allowing for a rapid predissociation.

VI. RECOMBINATION DYNAMICS

It seems evident now that all features in the UV and bleach regions of the iodine system can be best explained in terms of a rapid predissociation, 1 ps. Assignment of the UV absorption features in light of this conclusion leads directly to a reevaluation of the time scale for geminate recombination. The wavelength dependence of the

absorption kinetics is characteristic of a vibrational relaxation process, as discussed above. This suggests that the most reasonable assignment would be to absorptions from one of the three significantly bound states, the A, A', or ground states, yet distinguishing between these three states is difficult. Interpretation of the early time 295nm absorption (Fig. 4) as a continuation of the relaxation progression from 400 to 350nm requires assignment of the ground state as the origin of the UV absorptions since the time scale of the 295nm kinetics is too long for relaxation in the A or A' states. This feature does show a qualitatively similar solvent dependence when compared with ground state relaxation times determined previously from infrared absorption data (see table 3, Ref.3, comparing with kinetics at 860nm in particular). In addition, the rise times for the UV features are comparable to those for several ground state absorptions in the IR[3], while A' state features, when not obscured by other absorptions, always grow in in > 10 ps. It is therefore possible to make a somewhat tentative assignment of the UV features to absorptions from the upper portions of the ground state.

Regardless of which bound state is the origin of the UV absorptions, all of these states are populated only after recombination which, given the rapid rise of the 400nm feature, indicates that geminate recombination is an extremely rapid process (2 ps). This value is in reasonable agreement with several molecular dynamics simulations. Results of I_2/Xe and I_2/Kr simulations in general predict recombination times of between 1 and 5 ps.[35-38] Most of these simulations do not include the effects of multiple electronic states -

there are 10 states (16 including degeneracy) which correspond to two ground state iodine atoms - and as a result treat curve-crossing in a simplified or completely artificial manner. Inclusion of realistic surface-hopping models into simulations would be necessary in order to predict details of the recombination process, in particular partitioning dynamics between the ground and A/A' states. In addition, it has been suggested that inclusion of all possible states could slow recombination down by as much as a factor of 5.[33] The fact that such rapid recombination is observed, in agreement with simulations which ignore multiple electronic states, indicates that curve-crossing between electronic surfaces near-resonant in energy is extremely facile in solution and not the rate limiting step in recombination. Molecular dynamics simulations effectively model energy transfer processes and structural effects (caging) of the solvent and should therefore give reasonable predictions of the time scale for geminate recombination in spite of their simplified treatment of curve-crossing.

VII. CONCLUSIONS

Results have been presented which clarify the time scales for predissociation and geminate recombination in the iodine photodissociation/recombination reaction. The similarity of UV absorption features following A state and B state excitation can be best explained by a rapid predissociation (1 ps). This is significantly faster than previously thought. In addition geminate recombination is also seen to be rapid (2 ps). Both of these results

lead to the conclusion that curve-crossing in solution is extremely facile. Molecular dynamics simulations therefore are valuable in predicting recombination rates in this system in spite of treating curve-crossing in an artificial manner. In addition, it has been shown that in order to adequately interpret bleach kinetics it is necessary to consider dynamics within the ground state population remaining after excitation. This effect should be generally important in any system with significant thermal population in excited vibrational levels and with a ground state reequilibration rate slower than the experimental time resolution.

References

1. E. Rabinowitch and W. C. Wood, Trans. Faraday Soc. 32, 547,1381 (1936).

2. T. J. Chuang, G. W. Hoffman, and K. B. Eisenthal, Chem. Phys. Lett. 25, 201 (1974).

3. A. L. Harris, M. Berg, and C. B. Harris, J. Chem. Phys. 84, 788 (1986).

4. M. E. Paige, D. J. Russell, and C. B. Harris, J. Chem. Phys. 85, 3699 (1986).

5. M. Berg, A. L. Harris, and C. B. Harris, Phys. Rev. Lett. 54, 951 (1985).

6. M. Berg, A. L. Harris, J. K. Brown, and C. B. Harris, in Ultrafast Phenomena IV, edited by D. H. Auston and K. B. Eisenthal, Springer Series in Chemical Physics (Springer, New York, 1984), Vol. 38, p. 300.

7. N. A. Abul-Haj and D. F. Kelley, J. Chem. Phys. 84, 1335 (1986).

8. D. F. Kelley, N. A. Abul-Haj, and D.-J. Jang, J. Chem. Phys. 80, 4105 (1984).

9. D. F. Kelley and N. A. Abul-Haj, in Ultrafast Phenomena IV, edited by D. H. Auston and K. B. Eisenthal, Springer Series in Chemical Physics (Springer, New York, 1984), Vol. 38, p. 292.

10. D. F. Kelley and P. M. Rentzepis, Chem. Phys. Lett. 85, 85 (1982).

11. N. A. Abul-Haj and D. F. Kelley, Proc. SPIE-Int. Soc. Photo-Opt. Eng. 533, 20 (1985).

12. P. Bado, C. Dupuy, D. Magde, K. R. Wilson, and M. M. Malley, J. Chem. Phys. 80, 5531 (1984).

13. P. Bado and K. R. Wilson, J. Phys. Chem. 88, 655 (1984).

14. P. Bado, P. H. Berens, J. P. Bergsma, S. B. Wilson, K. R. Wilson, and E. J. Heller, in Picosecond Phenomena III, edited by K. B. Eisenthal, R. M. Hochstrasser, W. Kaiser, and A. Laubereau, Springer Series in Chemical Physics (Springer, New York, 1982), Vol. 23, p. 260.

15. P. Bado, C. G. Dupuy, J. P. Bergsma, and K. R. Wilson, in Ultrafast Phenomena IV, edited by D. H. Auston and K. B. Eisenthal, Springer Series in Chemical Physics (Springer, New York, 1984), Vol. 38, p. 296.

16. C. A. Langhoff, B. Moore, and M. DeMeuse, J. Am. Chem. Soc. 104, 3576 (1982).

17. C. A. Langhoff, B. Moore, and M. DeMeuse, J. Chem. Phys. 78, 1191 (1983).

18. A. L. Harris, J. K. Brown, and C. B. Harris, Ann. Rev. Phys. Chem., 39, 341 (1988).

19. J. Tellinghuisen, J. Chem. Phys. 82, 4012 (1985).

20. J. I. Steinfeld, J. Chem. Phys. 44, 2740 (1966).

21. G. A. Capelle and H. P. Broida, J. Chem. Phys. 58, 4212 (1973).

22. D. J. Nesbitt and J. T. Hynes, J. Chem. Phys. 77, 2130 (1982).

23. R. S. Mulliken, J. Chem. Phys. 55, 288 (1971).

24. J. Tellinghuisen, J. Chem. Phys. 76, 4736 (1982).

25. G. T. Evans and M. Fixman, J. Phys. Chem. 80, 1544 (1976).

26. B. Martire and R. G. Gilbert, Chem. Phys. 56, 241 (1981).

27. J. T. Hynes, R. Kapral, and G. M. Torrie, J. Chem. Phys. 72, 177 (1980).

28. M. Schell and R. Kapral, Chem. Phys. Lett. 81, 83 (1981).

29. K. J. Shin and R. Kapral, J. Chem. Phys. 69, 3685 (1978).

30. M. W. Balk, C. L. Brooks III, and S. A. Adelman, J. Chem. Phys. 79, 804 (1983).

31. C. L. Brooks III and S. A. Adelman, J. Chem. Phys. 80, 5598 (1984).

32. J. M. Dawes and M. G. Sceats, Chem. Phys. 96, 315 (1985).

33. D. P. Ali and W. H. Miller, J. Chem. Phys. 78, 6640 (1983); Chem. Phys. Lett. 105, 501 (1984).

34. M. Berg, A. L. Harris, J. K. Brown, and C. B. Harris, Opt. Lett. 9, 50 (1984).

35. J. K. Brown, C. B. Harris, and J. C. Tully, J. Chem. Phys. 89, 6687 (1988).

36. D. L. Bunker and B. S. Jacobson, J. Am. Chem. Soc. 94, 1843 (1972).

37. J. N. Murrell, A. J. Stace, and R. Dammell, J. Chem. Soc. Faraday Trans. 2 74, 1532 (1978).

38. A. H. Lipkus, F. P. Buff, and M. G. Sceats, J. Chem. Phys. 79, 4830 (1983).

Chapter 3: NUMERICAL INTEGRATION OF THE GENERALIZED LANGEVIN EQUATION THROUGH AUTOREGRESSIVE MODELING OF THE MEMORY FUNCTION

I. INTRODUCTION

Computer simulations have recently come to occupy a unique place in scientific investigation, providing an important bridge between experiment and theory. The simulation method of molecular dynamics (MD) provides a particularly powerful tool for the study of dynamical properties of molecular systems. The power of MD lies in the fact that dynamics of complicated systems are simulated explicitly through integration of Newton's equations with some assumed potential between particles. Yet the power of MD is also its shortcoming -- explicit simulation of large molecular systems or of slow processes (slow relative to the molecular motions) can be extremely computationally intensive. If the system is large enough or the process being studied is slow enough it can become impossible to use MD at all. One approach to overcoming these computational burdens lies in the development of faster and more numerous supercomputers -- and in fact this area has seen rapid growth and will certainly open up vast new areas to the tools of computer simulations in the years to come.[1] An alternative approach considered here is based on the elimination of non-relevant degrees of freedom from the MD system.

In many cases, while effective use of MD requires simulation of a large system, the process of interest may be isolated to a few atoms or molecules. This is typical of simulations of chemical reactions in

solution, where the reaction may include as few as two atoms, yet effective simulation of the reactant/solvent system may require consideration of hundreds of solvent atoms as well. Thus an enormous amount of computer time, in fact almost all of the simulation time in some cases, is devoted to explicit simulation of solvent degrees of freedom which are, at most, of secondary interest. This leads to the obvious question of whether these non-relevant solvent degrees of freedom can be replaced by some simple model solvent. The particular approaches most commonly used to this effect are brownian dynamics (BD)[2], based on the Langevin equation[3], or generalized brownian dynamics (GBD)[4], based on the generalized Langevin equation (GLE). BD techniques are much more easily implemented, yet in cases where the fundamental assumption of the Langevin equation breaks down, that is, when the solute particle is <u>not</u> much heavier than the solvent particles, it is necessary to consider GBD.

In GBD, the full many-body interactions of the solvent are decomposed into random driving forces and systematic dissipative forces through use of sets of coupled GLE's. The generalized Langevin equation can be written in scalar form for a single particle of mass m as[5-7]

$$m\dot{v}(t) = -\int_{t_0}^{t} dt' m\gamma(t')v(t-t') + R(t) \qquad (1)$$

where R(t) is the "random" force and the integral, a convolution of the velocity v(t) with the memory function $\gamma(t)$, is the dissipative force.

It can be further shown that R(t) and $\gamma(t)$ are related by the second fluctuation dissipation theorem:[7]

$$\langle R(t')R(t'+t)\rangle = k_b T m \gamma(t) \tag{2}$$

where the brackets indicate an ensemble average, k_b is Boltzmann's constant and T is the temperature. The GLE is particularly useful in that it easily lends itself to approximation. In particular, R(t) can be represented as a gaussian random force and the memory function, which in essence contains all relevant information about the full many-body dynamics of the solvent, can be approximated by a variety of simple functional forms, thus making evaluation of the GLE a relatively simple task.

There have been a variety of approaches to efficient numerical evaluation of the GLE. Most of these have involved modeling $\gamma(t)$ with one of a few very simple models.[8-14] Yet often it would be desirable to model $\gamma(t)$ to arbitrary closeness, especially when the purpose is not in applying meaning to the parameters as much as in using the modeled memory function in a simulation. There have been several approaches to this[4,15-18], two of which will be specifically discussed in this chapter; one developed by Ciccotti and coworkers,[4,17] and another by Adelman and coworkers.[18] In this chapter a general method for memory function modeling is presented which is on some level formally equivalent to that of Ciccotti and Adelman, yet quite different in its realization. It is based on autoregressive (AR) techniques and provides for both computationally efficient evaluation

of the friction integral in the GLE and generation of a process R(t) with the desired correlation function.

The random force R(t) is modeled as an AR process with the following generating function

$$x_n = -\sum_{j=1}^{p} a_j x_{n-j} + b_0 w_n \quad , \tag{3}$$

where p is the model order, x_n is a discrete process to be equated with R(t) and the a_j's are the AR parameters. w_n is a normalized gaussian distributed white noise source,

$$\langle w_n \rangle = 0.$$
$$\langle w_n w_{n+\tau} \rangle = \delta_\tau \quad ,$$

and b_0 is a factor which scales the magnitude of the correlation sequence associated with the process x_n. The techniques presented here are derived from an extensive literature in the field of digital signal processing and spectrum estimation[19-22], where AR and related techniques are used widely in a variety of applications ranging from radar processing and radio astronomy to economic forecasting and geophysics, where the AR method was originally developed and termed the maximum entropy method. (For a more complete list with references see Ref. 22. For literature showing the equivalence of autoregressive and maximum entropy methods see Ref. 21.) The power of the AR technique, from an engineering perspective, lies in its computational efficiency in providing reasonable spectrum estimations of digital signals even when those signals are corrupted badly with noise. For the present

applications, the AR procedure provides a convenient and powerful method for modeling correlation functions, in particular the memory function of the GLE, and in the process provides for generation of the random force process associated with the memory function.

In Cha. 4, this technique is applied to the vibrational relaxation of I_2 in Lennard-Jones (L-J) xenon at the temperature and densities used in the corresponding MD simulations of Brown and coworkers.[23] The memory functions for these GBD simulations are determined from force autocorrelation functions (FAF's) for single iodine atoms in L-J xenon as determined from equilibrium MD simulations. Results are encouraging as to the potential applicability of GBD techniques to problems of chemical interest.

The outline of this chapter is as follows. In Sec. II the method for determination of the AR parameters is described and the functional form for the resulting approximation of $\gamma(t)$ is derived. In Sec. III the AR technique is applied to a test case: $\gamma(t)$ for an iodine atom in liquid xenon. In Sec. IV some important details of the numerical implementation of this technique are described and results of a GLE simulation of an iodine atom in liquid xenon are presented. Discussion of the results and conclusions are presented in Sec. V. Finally, in the Appendix, the connection between this work and the work of Ciccotti and coworkers[4,17] as well as that of Adelman and coworkers[18] is explicitly derived.

II. AUTOREGRESSIVE MODELING OF $\gamma(t)$

This section consists of a description of the procedure used to determine the AR parameters from a given input memory function and a derivation of the functional form for the resulting AR approximation of $\gamma(t)$.

A. Determination of AR Parameters

Given a random force autocorrelation sequence γ_τ, where $(\gamma_\tau;\ \tau = 0,1,2,\ \ldots)$ is a discrete representation of $\gamma(t)$, determination of the AR parameters which give

$$\gamma_\tau \approx r_\tau = \langle x_n x_{n+\tau} \rangle$$

is straightforward. The brackets refer to the ensemble or time average. Multiplying Eq. (3) by $x_{n-\tau}$ and taking the ensemble average gives

$$\langle x_{n-\tau} x_n \rangle = -\sum_{j=1}^{p} a_j \langle x_{n-\tau} x_{n-j} \rangle \qquad \tau = 1,2,\cdots,N \tag{2.1}$$

since $\langle x_{n-\tau} w_n \rangle = 0$; $\tau > 0$. With $\gamma_\tau \approx \langle x_n x_{n+\tau} \rangle$, Eq. (2.1) becomes

$$\gamma_\tau \approx -\sum_{j=1}^{p} a_j \gamma_{\tau-j} \qquad \tau = 1,2,\cdots,N \quad , \tag{2.2}$$

where N corresponds to the number of points in γ_τ used to determine the set $\{a_j\}$. This set of equations is often referred to as the Yule-

Walker equations.[24,25] Writing Eq. (2.2) in matrix notation, and noting that $\gamma_\tau = \gamma_{-\tau}$ gives

$$\begin{bmatrix} \gamma_0 & \gamma_1 & \cdots\cdots & \gamma_{p-1} \\ \gamma_1 & \gamma_0 & \cdots\cdots & \gamma_{p-2} \\ \gamma_2 & \gamma_1 & \cdots\cdots & \gamma_{p-3} \\ \cdot & \cdot & \cdots\cdots & \cdot \\ \cdot & \cdot & \cdots\cdots & \cdot \\ \gamma_{N-1} & \gamma_{N-2} & \cdots\cdots & \gamma_{N-p} \end{bmatrix} \times \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \cdot \\ a_p \end{bmatrix} \approx - \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \cdot \\ \cdot \\ \gamma_N \end{bmatrix} , \tag{2.3}$$

which can be written as

$$\underset{\approx}{M}\underset{\sim}{a} \approx -\underset{\sim}{\gamma} \quad . \tag{2.4}$$

For N = p, Eq. (2.4) is solved by

$$\underset{\sim}{a} = -\mathrm{inv}(\underset{\approx}{M})\underset{\sim}{\gamma} \quad . \tag{2.5}$$

If N > p there is in general no unique solution for $\underset{\sim}{a}$, and it is necessary to solve for the vector $\underset{\sim}{a}$ through some fitting procedure, for example by minimizing

$$\sum_{\tau=1}^{N} (\epsilon_\tau)^2 \quad , \tag{2.6}$$

where $\epsilon_\tau = \gamma_\tau + a_1\gamma_{\tau-1} + \cdots + a_p\gamma_{\tau-p}$.

This minimization is accomplished using standard matrix techniques by

$$\underset{\sim}{a} = -\mathrm{inv}(\underset{\approx}{M}^T\underset{\approx}{M})\underset{\approx}{M}^T\underset{\sim}{\gamma} \quad , \tag{2.7}$$

where T refers to the matrix transpose. In practice the matrix $\underset{\approx}{M}^T\underset{\approx}{M}$ is often near singular, and in the examples presented in this chapter quadruple precision has been used for the matrix inversion. Note also that this technique does not provide a best fit of the correlation sequence γ_τ but rather minimizes the sum in Eq. (2.6). This is desirable given the linear nature of the Yule-Walker equations (2.2) - a direct fit to γ_τ is nonlinear and not at all straightforward to perform. The minimization of (2.6) unfortunately does <u>not</u> always provide a near best fit of γ_τ, particularly for large model orders or when N >> p. This will be discussed further in Sec. III.

B. Functional form for r_τ

Given the discrete process x_n of order p, the resulting correlation sequence $r_\tau = \langle x_n x_{n+\tau}\rangle$ is determined by first Fourier transforming x_n.

$$\hat{x}(\omega) = \frac{b_0}{1 + a_1 e^{-i\omega} + \cdots + a_p e^{-ip\omega}} \tag{2.8}$$

where $0 \le \omega < 2\pi$. The carat denotes the Fourier transform. The denominator in (2.8) is a polynomial in $e^{-i\omega}$. Taking the roots of this polynomial gives

$$\hat{x}(\omega) = \frac{b_0}{\prod_{j=1}^{p}(1-\exp(\omega_j)\exp(-i\omega))} \quad , \tag{2.9}$$

where the ω_j's are in general complex and occur in complex conjugate pairs since $\{a_j\}$ are real. In addition, stability requires that the magnitude of each root, $\exp(\omega_j)$, be less than unity. This is accomplished without changing the spectrum by reflecting each unstable root into the unit circle[26], i.e. by making the real part of each ω_j negative,

$$\mathrm{Re}(\omega_j) = -\mathrm{abs}[\mathrm{Re}(\omega_j)] \quad ,$$

where "abs" is the absolute value. By definition, the Fourier transform of the correlation sequence $r_\tau = \langle x_n x_{n+\tau}\rangle$ is given by

$$\hat{r}_\omega(p) = \langle\hat{x}(\omega)\hat{x}^*(\omega)\rangle$$

$$= \frac{b_0^2}{(1+a_1e^{-i\omega}+\cdots+a_pe^{-ip\omega})(1+a_1e^{i\omega}+\cdots+a_pe^{ip\omega})}$$

$$= \frac{b_0^2}{\prod_{j=1}^{p}(1-\exp(\omega_j)\exp(-i\omega))(1-\exp(\omega_j)\exp(i\omega))} \quad . \tag{2.10}$$

Inverse transforming (2.10) gives

$$r_\tau(p) = \sum_{j=1}^{p} A_j \exp(\omega_j \tau) \qquad \tau = 0,1,2,\cdots \quad . \tag{2.11}$$

with

$$A_j = \frac{b_0^2}{(1-\exp(2\omega_j)\prod_{\substack{k=1\\k\neq j}}^{p}(1-\exp(\omega_k)\exp(-\omega_j))(1-\exp(\omega_k)\exp(\omega_j))} . \tag{2.12}$$

Thus it is seen from Eq. (2.11) that the resulting correlation sequence $r_\tau(p)$, which is the AR model of γ_τ, is a sum of exponentials. This multiexponential form will be seen in Sec. IV to provide the basis for both a computationally efficient algorithm for solving the friction integral in the GLE and for a method for generalizing the time step implicit in the discrete AR process.

III. EXAMPLE: I/Xe

In Cha. 4 the vibrational relaxation of iodine in L-J xenon is investigated with GBD simulations. Input parameters for these simulations are derived from equilibrium force autocorrelation functions (FAF's) for single iodine atoms in liquid xenon at various densities as calculated from molecular dynamics. The memory function $\gamma(t)$ for each density is derived from the corresponding FAF as will be described below, and is modeled using the techniques presented in this chapter. One of these iodine atom FAF's is considered in this section

as an illustration of the effectiveness of the AR technique for memory function modeling.

Begin with the FAF for an iodine atom in L-J liquid xenon at temperature 300 K and reduced density $\rho^* = 0.57$ (corresponding to a density of 1.8 g/cm^3), shown in Fig. 1. $\rho^* = \rho\sigma^3$, where σ is the L-J radius and ρ is the number density. Calculation of this FAF from MD is described in Cha. 4. The memory function $\gamma(t)$ can be determined from MD in a variety of ways[27-29], and here use is made of a relationship derived by Kubo in which $\gamma(t)$ is determined directly from the total FAF for the atom in solution[7],

$$\frac{1}{\gamma[\omega]} = \frac{1}{\gamma_T[\omega]} - \frac{1}{i\omega} \tag{3.1}$$

where $$\gamma[\omega] = \int_0^\infty \gamma(t)\exp(-i\omega t)dt$$

and $$\gamma_T[\omega] = \int_0^\infty \frac{\langle f(t')f(t'+t)\rangle}{k_b Tm}\exp(-i\omega t)dt \quad ,$$

with $\langle f(t')f(t'+t)\rangle$ being the total FAF, a quantity which is easily generated from a molecular dynamics simulation. Equation (3.1) can be solved using Fourier transform techniques since the correlation functions are all even functions in time. Figure 1 shows the random and total FAF's for the iodine atom. Recall (Cf. Eq. (2)) that the random FAF equals $k_b Tm\gamma(t)$.

Figure 1. Random and total force autocorrelation functions for an iodine atom in Lennard-Jones xenon, $\rho^* = 0.57$, and $T = 300$ K. Magnitudes are in ps^{-2} corresponding to force squared divided by $k_b Tm$.

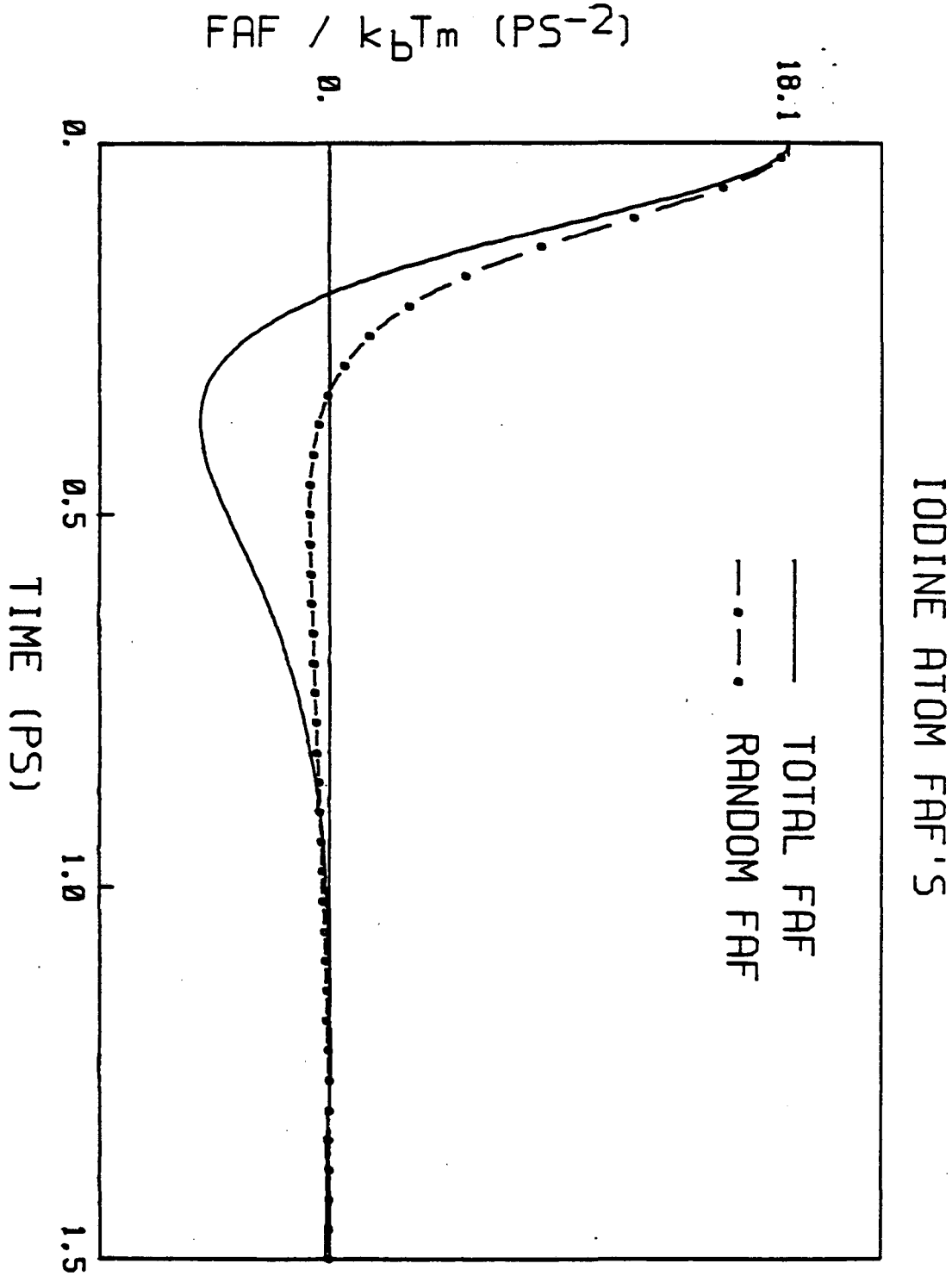
IODINE ATOM FAF'S
TOTAL FAF
RANDOM FAF
FAF / k_bTm (PS^{-2})
18.1
0.
0.
0.5
1.0
1.5
TIME (PS)

Figure 1.

Figure 2 shows results of several AR fits to $\gamma(t)$ for model orders ranging from 4 to 49. N = p in each case so fitting proceeds as in Eq. (2.5), and the fits cover the first 10*N femtoseconds of $\gamma(t)$. (The time interval between points in the sequence γ_τ is 10 femtoseconds) Agreement is, as expected, essentially exact out to τ = N and deviates thereafter. Two comments are in order here. First, in practice it is typically <u>not</u> necessary to use model orders of anywhere near 50 to effectively model the memory function. For example, in Cha. 4 a model order of four will be shown to be adequate for simulations of vibrational relaxation. This is due both to the fact that it is only the early time portion of $\gamma(t)$ which contributes to vibrational relaxation and that it is effective to use an overdetermined set of Eqs. (2.3) to determine the AR parameters in this case. Use of the overdetermined set of equations has the obvious advantage of keeping model orders as small as desired, yet as mentioned above it does not always achieve a reasonable fit to $\gamma(t)$. It has been found that it is usually reasonable to use an overdetermined set of equations if the model order is relatively small, less than about eight, and N is not extremely large, $\lesssim$ 50. In brief, reasonably small model orders can usually be achieved through consideration of which region of $\gamma(t)$ is most critical and through careful use of overdetermined sets of equations in the AR parameter determination. Second, integration of the GLE, as will be shown below, is computationally quite efficient and thus it is not at all unreasonable to perform simulations with large model orders if it is found to be necessary. The AR procedure provides

Figure 2. AR fits to the memory function $\gamma(t)$ calculated from Eq. (2.5) for model orders 4, 17, 30, and 49 shown in comparison with $\gamma(t)$ out to a time of 1.0 ps.

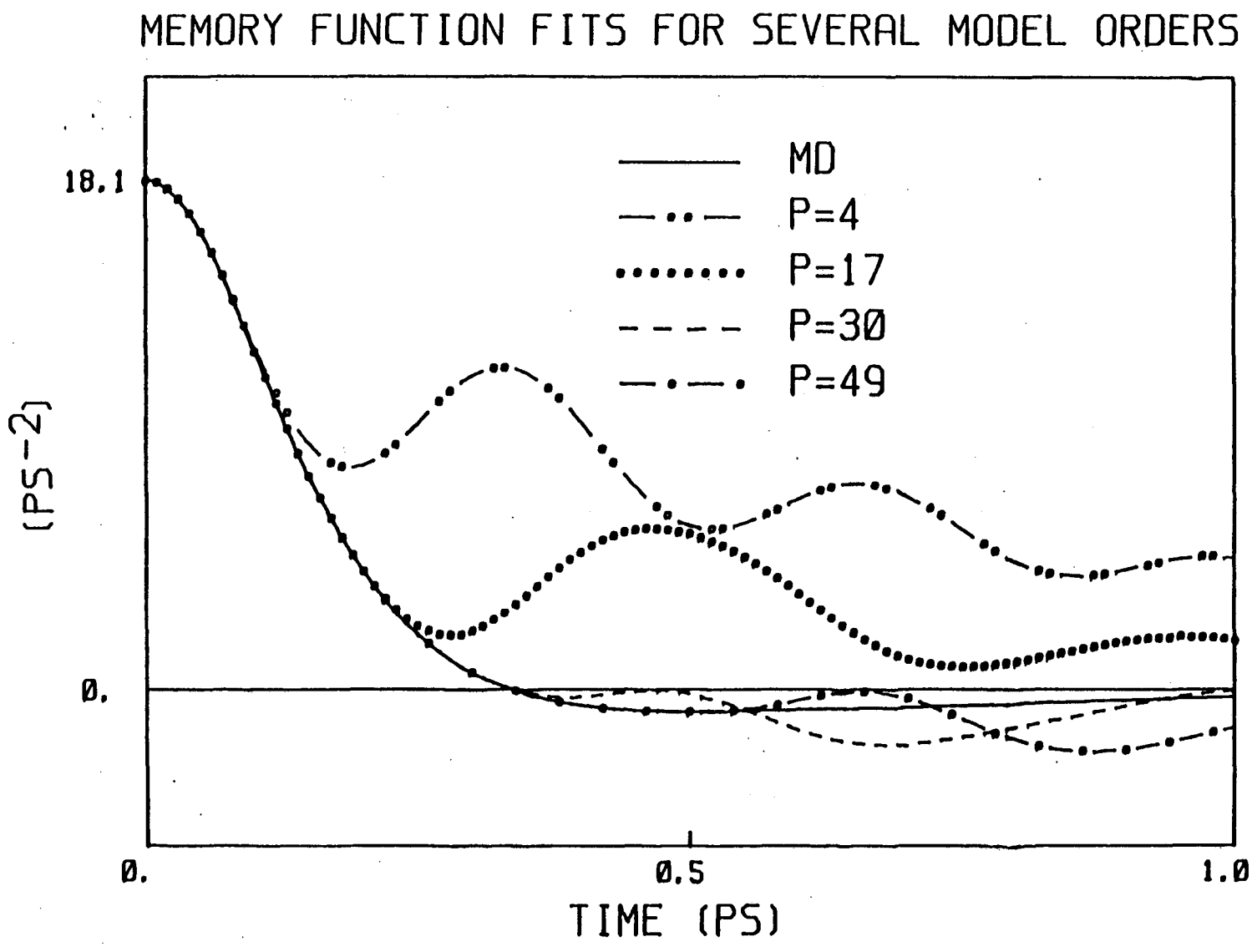
MEMORY FUNCTION FITS FOR SEVERAL MODEL ORDERS
MD
P=4
P=17
P=30
P=49
18.1
0.
(PS-2)
0.
0.5
1.0
TIME (PS)

Figure 2.

a general and effective method for modeling memory functions to arbitrary closeness.

Figure 3 shows results of a numerical calculation of the correlation sequence $r_\tau = \langle x_n, x_{n+\tau} \rangle$ for a model order of 49 in comparison with the expected correlation sequence as seen in Fig. 2. x_n is calculated from Eq. (3) for n = 1 to 500,000, after an initial 5000 steps to allow the sequence to converge, with the correlation sequence r_τ calculated at every time step. It can be seen that agreement is good, confirming the analysis of Sec. II(B).

IV. NUMERICAL APPLICATIONS

Efficient numerical integration of the GLE using an AR model for the memory function requires both an efficient evaluation of the friction integral in the GLE and a generalization of the AR model to arbitrary time step. Both of these requirements are met through taking advantage of the multiexponential form for r_τ (Eq. (2.11)). This section deals first with the generalization to arbitrary time step followed by a presentation of a numerically efficient algorithm for evaluation of the friction integral. Finally, these numerical methods are illustrated in Sec. C through a simulation of an iodine atom in liquid xenon.

A. Time Step Generalization

The difficulty with using an inherently discrete process to

Figure 3. Results of a numerical calculation of $r_\tau(49)$. x_n is calculated from Eq. (3) with $p = 49$ and the AR parameters corresponding to the $p = 49$ case of Fig. 2.

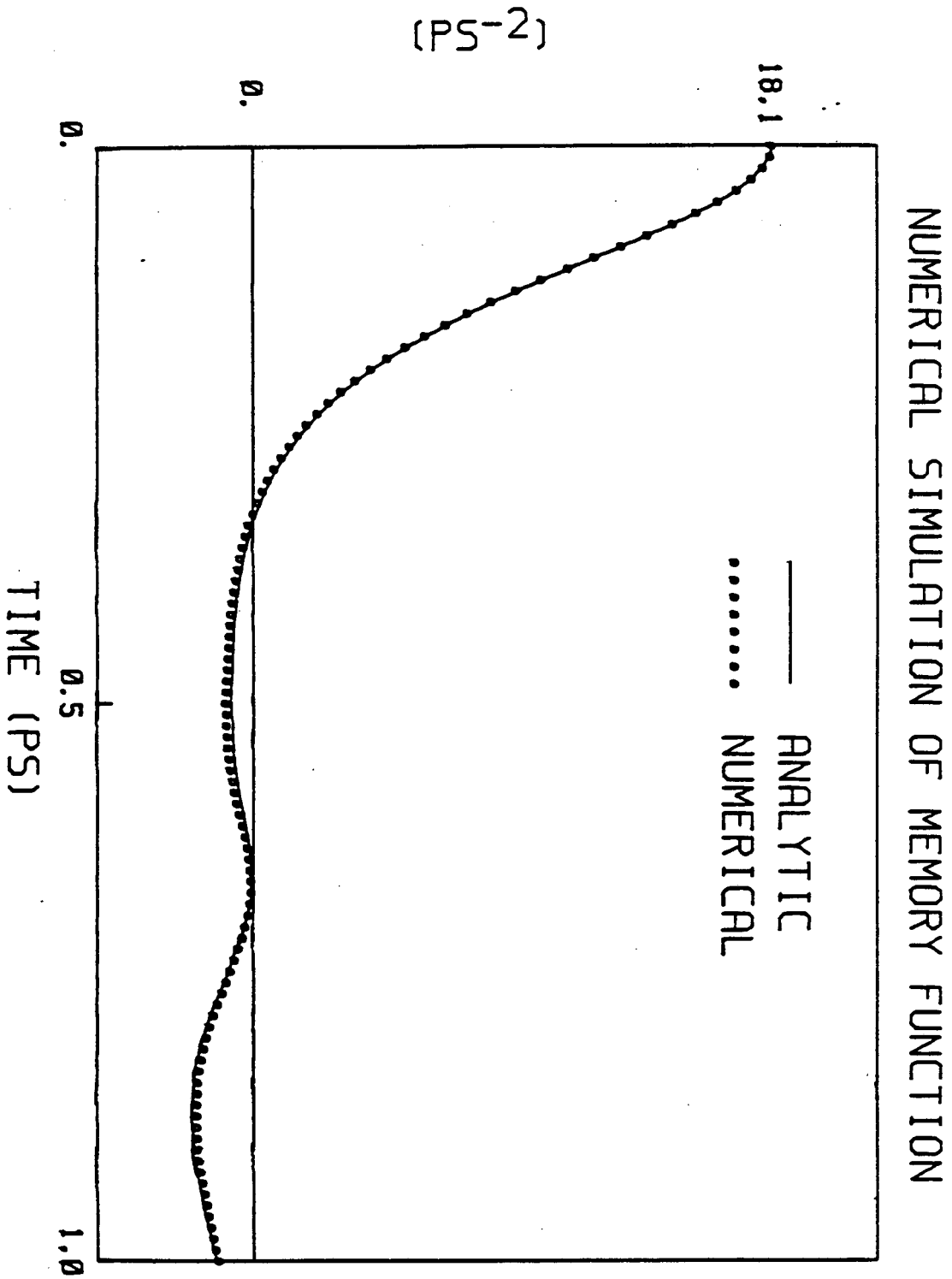
NUMERICAL SIMULATION OF MEMORY FUNCTION
ANALYTIC
NUMERICAL
18.1
0.
(PS-2)
0.
0.5
1.0
TIME (PS)

Figure 3.

describe the random force and memory function of the GLE is that the time step δ between points in the correlation sequence γ_τ is implicit in the resulting parameterization. In general δ is not the same as the time step used in GBD or MD simulations -- in fact it is frequently much larger. There are several ways to accomplish a time step generalization, including generating γ_τ with the desired time step or extrapolating between points in γ_τ and using the new sequence to determine the AR parameters. Either of these methods would require fitting to a larger number of correlation sequence points to achieve agreement with $\gamma(t)$ out to the same time, since the new time step is shorter than the original. This requires either an undesirable increase in model order or overcoming the numerical difficulties involved in fitting to an overdetermined set of equations as described above. A sometimes more useful method for generalizing the time step makes use of the multiexponential form of the correlation sequence of the discrete process x_n. Both r_τ and the discrete random force x_τ must be generalized.

Generalization of r_τ to arbitrary time step is accomplished trivially by scaling the frequencies in Eq. (2.11) by a factor of h/δ, where h is the GLE simulation time step and δ, as before, is the time between points in the correlation sequence γ_τ. The resulting sequence is

$$r_\tau(p) = \sum_{j=1}^{p} A_j \exp(\omega_j' \tau) \qquad \tau = 0,1,2,\cdots \quad , \qquad (4.1)$$

with $\omega_j' = (h/\delta)\omega_j$.

Generalization of x_τ to arbitrary time step is more difficult. Beginning with the new set of frequencies $\{\omega_j'\}$ it is easy to calculate the associated AR parameters $\{a_j\}$ from the relationship

$$1 + a_1' e^{-i\omega} + \cdots + a_p' e^{-ip\omega} = \prod_{j=1}^{p} (1-\exp(\omega_j')\exp(-i\omega)) \quad . \qquad (4.2)$$

However, the correlation sequence associated with the new set of AR parameters

$$r_\tau'(p) = \sum_{j=1}^{p} A_j' \exp(\omega_j' \tau) \qquad \tau = 0,1,2,\cdots \qquad (4.3)$$

is not equivalent to Eq. (4.1) since the scaling factors A_j' are functions of the frequencies $\{\omega_j\}$ (see Eq. (2.12)). For low model orders the primary differences between the resulting correlation sequence (4.3) and the correlation sequence of Eq. (4.1) is in their magnitudes at t=0. This magnitude can be trivially adjusted by changing the parameter b_0 in Eq. (3), and in practice this adjustment makes the difference between the correlation sequences rather small, at least for relatively low model orders. This is the method for time step generalization used in Cha. 4, and it is illustrated below in Sec. C.

B. Evaluation of the Friction Integral

The multiexponential functional form for $\gamma(t)$ is also the basis for a computationally efficient algorithm for solving the friction integral. The friction integral in Eq. (1) is written in discrete form as

$$F_\tau = -\sum_{j=0}^{\tau} hm\gamma_j v_{\tau-j} \quad , \tag{4.4}$$

where h is the time step in the simulation. Although this algorithm is derived for a first order discrete approximation to the friction integral, it can be easily generalized to arbitrary order. For the case where $\gamma(t)$ is modeled as a single exponential,

$$m\gamma_\tau = Ce^{-hd\tau} \quad ,$$

(4.4) becomes

$$F_\tau = -\sum_{j=0}^{\tau} hC\exp(-hdj)v_{\tau-j} \tag{4.5}$$

and

$$\begin{aligned} F_{\tau+1} &= -\sum_{j=0}^{\tau+1} hC\exp(-hdj)v_{\tau+1-j} \\ &= \exp(-hd)F_\tau - hCv_{\tau+1} \quad . \end{aligned} \tag{4.6}$$

Thus $F_{\tau+1}$ is calculated from F_τ and the discrete sum in (4.4) is not

calculated explicitly. For the case where $\gamma(t)$ is modeled as a multiexponential,

$$m\gamma_\tau = \sum_{k=1}^{p} C_k \exp(-hd_k \tau) \quad ,$$

(4.4) becomes

$$F_\tau = -\sum_{j=0}^{\tau} h\left(\sum_{k=1}^{p} C_k \exp(-hd_k j)\right) v_{\tau-j} \quad ,$$

which can be written

$$F_\tau = \sum_{k=1}^{p} \left(-\sum_{j=0}^{\tau} hC_k \exp(-hd_k j))v_{\tau-j}\right)$$

$$= \sum_{k=1}^{p} F'_{k,\tau} \quad . \tag{4.7}$$

Each element in (4.7) is calculated as in (4.6), and thus the friction integral in Eq. (1) is calculated in a computationally efficient way for a multiexponential form for $\gamma(t)$. Note that each element in Eq. (4.7), $F'_{k,\tau}$, is in general complex given that the frequencies in the AR correlation sequence are complex, yet the total frictional force F_τ is always real since the complex frequencies occur in complex conjugate pairs.

C. Simulation of I/Xe

As an illustration of the numerical methods described in this section a simulation of an iodine particle in liquid xenon has been performed. Parameter selection for this simulation is motivated by the system studied in Cha. 4, the vibrational relaxation of I_2 in liquid xenon. For this case it is primarily important to effectively model the early time portion of the memory function, as this region contains most of the high frequency components of $\gamma(t)$ which couple effectively with the diatomic vibrational motion. A value of $N = 30$, corresponding to the first 0.3 ps of $\gamma(t)$, will be shown in Cha. 4 to be sufficiently large to model the relaxation process. Model order is chosen to be as small as possible while still providing for reasonable relaxation results.

AR parameter determination proceeds for several model orders as in Eq. (2.7) with $N = 30$. $\gamma(t)$ is the same as used in Sec. III for an iodine atom in L-J liquid xenon at temperature of 300 K and reduced density 0.57. Analysis of goodness of fit for model orders ranging from $p = 2$ to $p = 6$ is shown in Table I and illustrates the steady improvement of fitting to $\gamma(t)$ as order is increased. Above $p = 6$ fitting no longer improves as model order increases, presumably since Eq. (2.7) does not fit directly to the correlation sequence, as has already been discussed. The time step used in the simulations in Cha. 4 is 1 femtosecond, a factor of 10 smaller than the time step δ between points in γ_τ. Calculation of parameters for the new time step proceeds as in Sec. IV(A), with the frequencies in Eq. (2.11) being scaled by a

Table I. Sum of squared residuals $(r_\tau(p) - \gamma_\tau)^2$ for several model order fits to the random FAF calculated for two different time steps h. N = 30 in all cases, and the residuals are summed over $\tau = 0,1, \ldots ,N$. AR parameters for the h = 10 fs case are calculated as in Eq. (2.7), while for the h = 1 fs case they are calculated using the method described in Sec. IV.

	Sum of squared residuals (ps^{-4})	
Model order	h = 10 fs	h = 1 fs
2	9.794 x10-1	1.062
3	5.381 x10-2	3.967 x10-3
4	2.233 x10-2	5.212 x10-4
6	1.135 x10-4	6.180 x10-2

factor of 10 and the new AR parameters being calculated from these frequencies. The parameter b_0 is scaled to adjust the magnitude of the resulting correlation sequence to equal γ_τ at $\tau = 0$. Analysis of goodness of fit for the resulting sequences is tabulated in Table I in comparison with the original fits and is seen to be quite good, actually improving for model orders 3 and 4. For model orders higher than about 4 this method for generalizing the time step for the AR parameters is only moderately effective, as in the $p = 6$ case. For applications which consider non-equilibrium dynamics, it may be much more critical to properly model $\gamma(t)$ than $R(t)$ since the magnitude of the velocity in the friction integral is typically large enough that the friction term dominates the random force term in the GLE. Thus for the purpose of simulations it may be perfectly adequate to use this method generalizing R_τ even when it gives a less than ideal fit to $R(t)$, i.e., as is the case with higher model orders (≥ 6). Typically, then, the best model for $\gamma(t)$ may not exactly correspond to the correlation function of the modeled random force, in violation of the fluctuation-dissipation theorem (Cf. Eq. (2)), yet in most cases this discrepancy should be quite small.

For the simulations in Cha. 4 it will be shown that a model order of four is sufficiently large to effectively represent $\gamma(t)$, and therefore that case is considered here. Simulation of the iodine atom in liquid xenon proceeds via numerical integration of the GLE, Eq. (1). $R(t)$ is generated from Eq. (3) using the fourth order AR parameters just calculated for the $h = 1$ fs case. The friction integral in Eq. (1) is calculated as described in Sec. IV(B), using the

multiexponential $r'_\tau(4)$ just calculated for the h = 1 fs case, i.e. from Eq. (4.3) since $r'_\tau(4)$ gives closer agreement with $\gamma(t)$ than $r_\tau(4)$ from Eq. (4.1). Integration proceeds using a second order integrator as described by Beeman[30] with a time step of 1 fs. After simulation of an initial 2 ps to eliminate any dependence on initial conditions, a single trajectory was run for 500 ps with the FAF for the iodine atom calculated every 10 fs and averaged over the trajectory. Figure 4 shows this FAF along with the total FAF of Fig. 1. In addition, the figure shows the expected FAF for the fourth order AR model as calculated analytically from Eq. (3.1) using a continuous representation of $r'_\tau(4)$ for $\gamma(t)$ in order to calculate the resulting total FAF. As can be seen, agreement between the simulated and expected FAF's is nearly exact, and both are in good agreement with the FAF of Fig. 1.

V. CONCLUSIONS

In this chapter, a general and efficient procedure for numerically integrating the generalized Langevin equation through autoregressive modeling of the memory function has been presented. The AR technique provides a powerful framework for modeling the memory function to arbitrary order which allows both for efficient evaluation of the friction integral in the GLE and for generation of a random force process with the desired spectrum. The discrete nature of the autoregressive method is quite natural for numerical simulations but can cause some complications in generalizing the method to arbitrary

Figure 4. FAF for an iodine atom in liquid xenon from a GLE simulation, with AR parameters calculated from Eq. (2.7) with N = 30 and a model order of four, in comparison with $\gamma(t)$ and the expected FAF as calculated from Eq. (3.1).

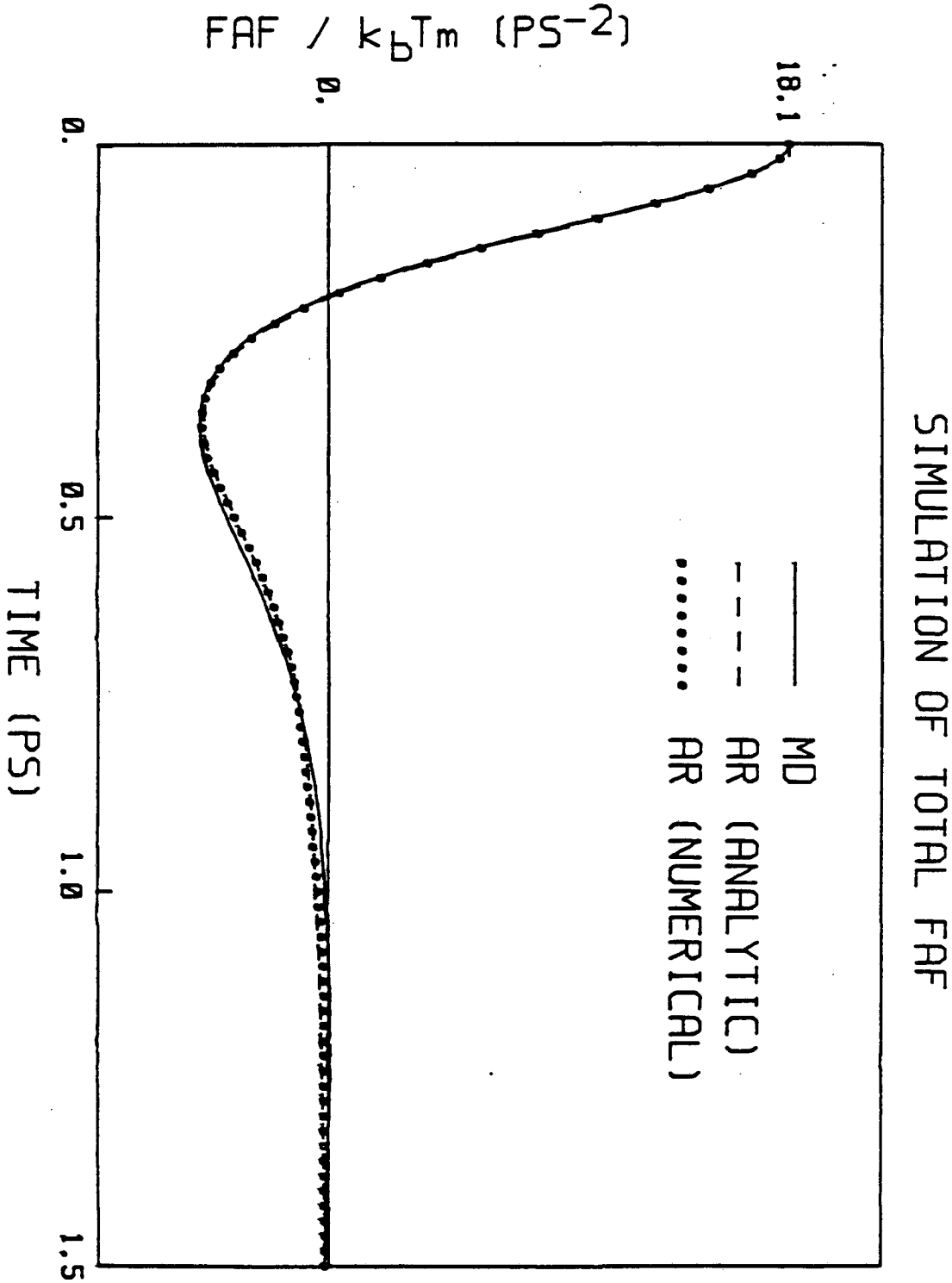
SIMULATION OF TOTAL FAF
MD
AR (ANALYTIC)
AR (NUMERICAL)
FAF / k_bTm (PS^{-2})
18.1
0.
0.
0.5
1.0
1.5
TIME (PS)

Figure 4.

simulation time step. This generalization has been shown to be trivial for the memory function given its multiexponential form, and more difficult, yet achievable, for the sequence associated with the random force. The multiexponential form for the memory function model is also responsible for an efficient procedure for numerical integration of the GLE, a feature which allows for computationally reasonable simulations with large model orders, ~50 or greater. This technique has been tested on the system of an iodine atom in liquid xenon, and found to be effective in modeling $\gamma(t)$ and in simulating iodine atomic motion.

In the following chapter this technique is applied to a problem of chemical interest -- diatomic vibrational relaxation in solution, and in particular to the relaxation of iodine in model liquid xenon. This is a classic case of a system which lends itself well to the techniques of generalized brownian dynamics. The system of interest is small, two atoms, yet effective MD simulation requires consideration of hundreds of solvent atoms as well.[23] The problem of iodine vibrational relaxation has been studied extensively both experimentally as well as with molecular dynamics (for an extensive review see Ref. 31), and it therefore provides an ideal testing ground for the effectiveness of GBD. The numerical methods presented in this chapter allow us to pursue that goal effectively.

APPENDIX

The work presented in this chapter consists of a procedure for modeling the memory function $\gamma(t)$ in the generalized Langevin equation to arbitrary order, which allows for efficient numerical evaluation of the GLE. Previous work in this field along similar lines has been conducted, and it is the purpose of this appendix to make a connection between the autoregressive method presented here and two other procedures described in the literature. Ciccotti and coworkers[4,17] have developed a method for numerical integration of the GLE based on a Mori continued fraction expansion[32] of the velocity autocorrelation function (VAF) (or of any numerically known autocorrelation function of a dynamical variable). They have shown that truncation of the continued fraction expansion after a finite number of terms leads to an efficient numerical procedure for integration of the GLE to arbitrary accuracy as determined from the number of terms remaining in the truncated expansion. Adelman and coworkers[18] have developed a different procedure for integration of the GLE based on a reformulation of the GLE into an equivalent set of "chain" equations which govern the motion of an infinite harmonic chain of fictitious atoms. Truncation of the chain after a finite number of atoms provides for efficient numerical integration, in the same spirit as the continued fraction method. Each of these techniques, including both reformulation of the GLE and subsequent truncation, are formally equivalent to the autoregressive method of this chapter, though all three methods are quite different in their realization.

The equivalence of the Ciccotti formalism and the AR formulation is detailed first. Ciccotti and Ryckaert have shown[17] that the GLE can be

rewritten as an infinite set of linear first order differential equations,

$$\dot{A}_0(t) = A_1(t)$$
$$\dot{A}_i(t) = -\gamma_i(0)A_{i-1}(t) + A_{i+1}(t), \quad (i=1,\infty) \quad , \qquad (A.1)$$

where $A_0(t)$ is the velocity and the coefficients $\gamma_i(0)$ correspond to the coefficients in the continued fraction expansion of the Laplace transform of the VAF, $C_v(t)$,

$$\tilde{C}_v(s) = \cfrac{C_v(0)}{s + \cfrac{\gamma_1}{s + \cfrac{\gamma_2}{\ddots}}} \quad , \qquad (A.2)$$

where ~ refers to the Laplace transform and $\gamma_i = \gamma_i(0)$. Truncation of the continued fraction after a finite number of terms,

$$\tilde{C}_v(s) = \cfrac{C_v(0)}{s + \cfrac{\gamma_1}{\ddots \; \cfrac{}{s + \cfrac{\gamma_{n-1}}{s + \beta}}}} \quad , \qquad (A.3)$$

gives rise to a finite set of differential equations to govern the motion of the brownian particle,

$$\dot{A}_0(t) = A_1(t)$$
$$\dot{A}_i(t) = -\gamma_i(0)A_{i-1}(t) + A_{i+1}(t), \quad (i=1,n-2)$$
$$\dot{A}_{n-1}(t) = -\gamma_{n-1}(0)A_{n-2}(t) - \beta A_{n-1} + \xi(t) \quad , \qquad (A.4)$$

where $\xi(t)$ is a gaussian white noise source. This finite set of differential equations provides for efficient integration of the GLE. This set of n first order equations can be rewritten as an nth order differential equation

$$A_0^{(n)}(t) + \alpha_1 A_0^{(n-1)}(t) + \cdots + \alpha_n A_0(t) = \xi(t) \quad , \qquad (A.5)$$

where the coefficients of the nth order differential equation are functions of β and (γ_i; i=1,n-1). Equation (A.5) is valid not only for $A_0(t) = v(t)$, but also for $A_0(t) = R(t)$, the random force in the GLE. This is easily seen noting that $\gamma_1(t)$, corresponding to $\gamma_1(0)$ in the continued fraction (A.2), is the memory function $\gamma(t)$ of Eq. (1) and therefore proportional to the random FAF and that there is a corresponding continued fraction expansion of $\gamma_1(t)$ analogous to Eq. (A.2)

$$\tilde{\gamma}_1(s) = \cfrac{\gamma_1(0)}{s + \cfrac{\gamma_2}{s + \cfrac{\gamma_3}{\ddots}}} \qquad (A.6)$$

Taking Eq. (A.5) as an nth order differential equation in the random force it is easy to see the connection with the AR procedure, since the AR generating function of Eq. (3) is equivalent to a discrete

approximation to equation (A.5). Thus the AR procedure is equivalent to the continued fraction-based method of Ciccotti and Ryckaert in the limit when the discrete approximation of the differential equation is valid, that is, in the limit of small time step. In any realistic simulation, such as those presented in Cha. 4, the time step is small enough that Eq. (3) and Eq. (A.5) are essentially equivalent.

The harmonic chain representation of the GLE developed by Adelman and coworkers is similarly related to the autoregressive procedure. This is most easily shown by deriving the equivalence between the harmonic chain representation and the continued fraction representation of Ciccotti and coworkers. Adelman has shown that the reformulation of the GLE into a set of "chain equations" can be represented as a continued fraction[18]

$$\tilde{\chi}(s) = \cfrac{1}{s^2 + \omega_{e0}^2 - \cfrac{\omega_{c1}^4}{s^2 + \omega_{e1}^2 - \cfrac{\omega_{c2}^4}{\cdots}}} , \tag{A.7}$$

where $\dot{\chi}(t) = C_v(t)/C_v(0)$, the normalized VAF. See Ref. 18 for meanings placed on the other parameters. Truncation of the harmonic chain after n atoms gives

$$\tilde{\chi}(s) = \cfrac{1}{s^2 + \omega_{e0}^2 - \cfrac{\omega_{c1}^4}{\ddots \; \cfrac{}{s^2 + \omega_{e(n-1)}^2 - \omega_{cn}^4 \beta}}} . \tag{A.8}$$

To connect with the continued fraction form of Ciccotti and Ryckaert, note that

$$\begin{aligned}\tilde{C}_v(s)/C_v(0) &= \tilde{\dot{\chi}}(s) \\ &= s\tilde{\chi}(s) - \chi(0) \\ &= s\tilde{\chi}(s) \qquad . \end{aligned} \tag{A.9}$$

Substituting Eq. (A.9) into the 2nth order truncation of the Ciccotti continued fraction, Eq. (A.3), gives

$$s\tilde{\chi}(s) = \cfrac{1}{s + \cfrac{\gamma_1}{\ddots \; \cfrac{}{s + \cfrac{\gamma_{2n-1}}{s + \beta}}}} \tag{A.10}$$

which, upon rearrangement, leads to

$$\tilde{\chi}(s) = \cfrac{1}{s^2 + \gamma_1 - \cfrac{\gamma_1\gamma_2}{s_2 + \gamma_2 + \gamma_3 - \cfrac{\gamma_3\gamma_4}{\ddots \; \cfrac{}{s^2 + \gamma_{2n-2} + \gamma_{2n-1} - \beta\gamma_{2n-1}}}}} \qquad . \tag{A.11}$$

This continued fraction is equivalent to the nth order truncation of Adelman's continued fraction, Eq. (A.8), thus proving the equivalence between an nth order truncation of Adelman's harmonic chain and a 2nth order truncation of Ciccotti's continued fraction, and therefore also

the equivalence of Adelman's method and the AR method, once again in the appropriate limit of small time step.

References

1. F. F. Abraham, Adv. Phys. 35, 1 (1986).

2. P. Turq, F. Lantelme, and H. L. Friedman, J. Chem. Phys. 66, 3039 (1977).

3. S. Chandrasekhar, Rev. Mod. Phys., 15, 1 (1943).

4. G. Ciccotti, M. Ferrario, and J.-P. Ryckaert, Mol. Phys. 46, 875 (1982).

5. H. Mori, Progr. Theor. Phys. 33, 423 (1965).

6. R. Zwanzig, Ann. Rev. Phys. Chem. 16, 67 (1965).

7. R. Kubo, Rep. Prog. Theor. Phys. 29, 255 (1966).

8. J. D. Doll and D. R. Dion, J. Chem. Phys. 65, 3762 (1976).

9. M. Shugard, J. C. Tully, and A. Nitzan, J. Chem. Phys. 66, 2534 (1977); 69, 336 (1978).

10. J. C. Tully, G. H. Gilmer, and M. Shugard, J. Chem. Phys. 71, 1630 (1979).

11. J. C. Tully, J. Chem. Phys. 73, 1975 (1980).

12. J. E. Straub, M. Borkovec, and B. J. Berne, J. Chem. Phys. 84, 1788 (1986).

13. J. K. Brown, D. J. Russell, D. E. Smith, and C. B. Harris, Rev. Phys. Appl. 22, 1787 (1987).

14. C. B. Harris, J. K. Brown, M. E. Paige, D. E. Smith, and D. J. Russell, in Ultrafast Phenomena V, edited by G. R. Fleming and A. E. Siegman, Springer Ser. Chem. Phys. 46, (Springer, New York, 1986), p. 326.

15. A. Nitzan, M. Shugard, and J. C. Tully, J. Chem. Phys. 69, 2525 (1978).

16. F. Marchesoni and P. Grigolini, J. Chem. Phys. 78, 6287 (1983).

17. G. Ciccotti and J.-P. Ryckaert, Mol. Phys. 40, 141 (1980).

18. S. A. Adelman, Adv. Chem. Phys. 53, 61 (1983), and references therein.

19. J. A. Cadzow, Proc. IEEE 70, 907 (1982), and references therein.

20. L. R. Rabiner and B. Gold, Theory and Application of Digital Signal Processing, (Prentice-Hall, New Jersey, 1975).

21. Modern Spectrum Analysis, edited by D. G. Childers (Wiley, New York, 1978).

22. S. M. Kay, and S. L. Marple, Jr., Proc. IEEE 69, 1380 (1981).

23. J. K. Brown, C. B. Harris, and J. C. Tully, J. Chem. Phys. 89, 6687 (1988).

24. G. U. Yule, Phil. Trans. R. Soc. London, Ser. A 226, 267 (1927).

25. G. Walker, Proc. R. Soc. London, Ser. A 13, 518 (1931).

26. B. Friedlander and B. Porat, IEEE Trans. Aerospace and Electronic Sys. AES-20, 158 (1984).

27. M. Berkowitz, J. D. Morgan, D. J. Kouri, and J. A. McCammon, J. Chem. Phys. 75, 2462 (1981).

28. D. Levesque and L. Verlet, Phys. Rev. A 2, 2514 (1970).

29. B. Berne and G. D. Harp, Adv. Chem. Phys. 17, 63 (1970).

30. D. Beeman, J. Comp. Phys. 20, 130 (1976).

31. A. L. Harris, J. K. Brown, and C. B. Harris, Ann. Rev. Phys. Chem. 39, 341 (1988) and references therein.

32. H. Mori, Prog. Theor. Phys. 34, 399 (1965).

Chapter 4: VIBRATIONAL RELAXATION OF DIATOMIC MOLECULES IN SOLUTION

I. INTRODUCTION

In this chapter a simple model for vibrational relaxation in solution based on the generalized Langevin equation (GLE) is developed. Input parameters for the model are derived directly from molecular dynamics (MD) simulations of single solute atoms dissolved in the solution of interest, with the only adjustable parameters in the model being the potentials used in the MD calculations. This model is applied to the system of I_2 in liquid xenon and thereby shown to be useful for the calculation of diatomic relaxation in simple solvents. Although it is developed specifically for the problem of diatomic relaxation, this model is potentially applicable to a variety of polyatomic systems as well. The purpose in developing this model is two-fold. The first purpose is to develop a simple and computationally efficient method for simulating diatomic vibrational relaxation in solution. This is desirable given the typically slow nature of diatomic relaxation in liquids relative to the time scale of the molecular motions. MD methods of computer simulation, which include explicit simulation of large numbers of solvent degrees of freedom in addition to consideration of the solute molecule, are quite computationally burdensome. The methods of generalized brownian dynamics (GBD), based on the generalized Langevin equation, provide an interesting and efficient alternative to MD. The second purpose is to

test the effectiveness of GLE-based methods of computer simulations when applied to systems far from equilibrium (i.e., when is the assumption of linear response, implicit in the GLE, valid?) and in the process increase our understanding of condensed phase energy transfer processes. This method is particularly well designed for this purpose given its close connection with MD, with the only adjustable parameters being the potentials used to calculate the input correlation functions. This close connection allows for both a direct and a meaningful comparison of GBD and MD simulation results.

In GBD, the full many-body interactions of the solvent are decomposed into random driving forces and systematic dissipative forces through use of sets of coupled GLE's. The generalized Langevin equation can be written in scalar form for a single particle of mass m as[1-3]

$$m\dot{v}(t) = -\int_{t_0}^{t} dt' m\gamma(t')v(t-t') + R(t) \quad , \qquad (1)$$

where R(t) is the "random" force and the convolution integral is the dissipative force. It can be further shown that R(t) and the memory function $\gamma(t)$ are related by the second fluctuation dissipation theorem,[3]

$$\langle R(t')R(t'+t)\rangle = k_b T m\gamma(t) \quad , \qquad (2)$$

where the brackets indicate an ensemble average, k_b is Boltzmann's constant and T is the temperature. The GLE is particularly useful in

that it easily lends itself to approximation. R(t) can be represented as a gaussian random force, and the memory function $\gamma(t)$, which in essence contains all relevant information about the full many-body dynamics of the solvent, can be approximated by a variety of simple functional forms, thus making evaluation of the GLE a relatively simple task. In Cha. 3, a general method for integration of the GLE was developed which 1) allows for modeling of the memory function to arbitrary order, 2) provides for generation of a gaussian "random force" process with the desired spectrum, and 3) enables computationally efficient evaluation of the friction integral in Eq. (1). In this chapter the numerical methods of Cha. 3 are applied to the problem of vibrational population relaxation in solution.

The problem of vibrational relaxation in solution has been approached from several directions (for recent reviews, see Refs. 4 and 5), the most common being based on perturbation theory[6-9] or independent binary collision (IBC) models.[10-17] More recent work has included a variety of classical[18-21] and semiclassical methods.[22-24] Hydrodynamic models for vibrational relaxation, of which this is one, include a variety of approaches[25-33], the most thoroughly developed of which is the MTGLE work of Adelman and coworkers[31-33]. Their approach consists of reformulating the generalized Langevin equation into an equivalent set of linear coupled equations which govern the motion of an infinite chain of harmonically bound "atoms". In the appendix of Cha. 3, it was shown that the MTGLE reformulation and the AR technique for integration of the GLE used here are equivalent in the appropriate

limit. The similarities and differences between the MTGLE model and this GLE-based model will be discussed further at the end of Sec. II.

The general technique used here to simulate diatomic vibrational relaxation is to model the solute-solvent interactions with GLE's while treating the intramolecular interaction explicitly. Molecular dynamics simulations of an equilibrium system are used to determine the appropriate form for the memory function, which is then modeled with a variety of multiple exponential functions. Vibrational relaxation rates are computed for each memory function model via integration of the GLE's and intramolecular equations of motion.

The specific system chosen for simulation here is the vibrational relaxation of iodine in liquid xenon. The iodine photodissociation and recombination reaction, including vibrational relaxation following recombination, has been studied extensively both experimentally and theoretically.[34] Since the advent of picosecond laser techniques, which made the dynamics of this simple chemical reaction experimentally accessible, much has been learned from it concerning solvent induced predissociation, recombination, and electronic and vibrational relaxation. Recent experiments by Paige et al.[35,36] have been directed at elucidating vibrational relaxation dynamics, specifically by isolating translational degrees of freedom of the solvent through studying the iodine reaction in a monatomic solvent, liquid xenon. Concurrently with these experiments, Brown, Harris, and Tully performed extensive molecular dynamics simulations of the same reaction system at identical densities and temperature.[18] This simple GBD model for vibrational relaxation will be evaluated through extensive comparison

with the molecular dynamics simulations of Brown et al. Results are used to evaluate the importance of various assumptions and approximations which go into this model, as well as the overall usefulness of these methods for modeling vibrational relaxation in solution.

The outline of the chapter is as follows. In Sec. II. a GBD model for vibrational relaxation is developed and the various assumptions which make this a simple model for relaxation are discussed. Section III. contains a description of molecular dynamics simulations used in the calculation of memory functions for the relaxation simulations, and also a description of an MD simulation of a rigid I_2 molecule used to test various approximations in the GBD model. Section IV. summarizes the autoregressive technique for memory function modeling which was developed in Cha. 3. Results of vibrational relaxation simulations of iodine in liquid xenon are presented in Sec. V and discussed in Sec. VI.

II. MODEL FOR VIBRATIONAL RELAXATION

A simple stochastic model for diatomic motion which depends on an intuitive extension of the generalized Langevin equation to a system of interacting particles is developed in this section. Implicit in this extension of the GLE are several assumptions which will be discussed with regard to diatomic systems. Discussion of these assumptions is followed by a description of the determination of the memory function $\gamma(t)$ from molecular dynamics simulations.

The generalized Langevin equation, although originally developed for a single spherical particle in a fluid, can be extended to the treatment of interacting particles in solution. Several attempts have been made to rigorously derive a GLE (and the associated fluctuation-dissipation theorem) for a system of interacting particles[37-39], and the resulting expressions are quite complex. An alternative, intuitive extension of the GLE to a system of two interacting particles takes the form of six coupled GLE's,

$$m\dot{v}_{i\alpha}(t) = -\int_{t_0}^{t} dt' m\gamma_{i\alpha}(t')v_{i\alpha}(t-t') + R_{i\alpha}(t) \qquad (2.1)$$

$$+ F_{i\alpha}(|\underset{\sim}{r}_2(t)-\underset{\sim}{r}_1(t)|) \quad , \qquad i=1,2;\ \alpha=1,3 \quad ,$$

where $F_{i\alpha}(|\underset{\sim}{r}_2(t)-\underset{\sim}{r}_1(t)|)$ is the force due to the internuclear interaction. The index i refers to the two atoms and the index α corresponds to the three cartesian coordinates. It is this intuitive extension which will be used here.

Writing the GLE in this simple form depends on two major assumptions about the solute-solvent interactions. First, it is assumed that the solvent interacts with each atom independently, that is, that the friction on each coordinate is a function only of the motion of that coordinate and that the random forces $R_{i\alpha}(t)$ are independent. In particular, cross correlations between the solvent interactions with each solute atom are ignored, including the compression of the diatomic due to the pressure of the system. This assumption will later be shown to be valid for the calculation of

vibrational relaxation of iodine in Lennard-Jones (L-J) xenon through a molecular dynamics simulation of a rigid I_2 molecule in solution. This simulation will show that the cross correlations are low frequency in nature and do not couple efficiently to the vibrational motion. As a result of this assumption, a fluctuation-dissipation theorem can be derived for Eq. (2.1) which is identical in form to the second fluctuation-dissipation theorem of Eq. (2),

$$\langle R_{i\alpha}(t')R_{i\alpha}(t'+t)\rangle = k_b T m \gamma_{i\alpha}(t) \tag{2.2}$$

The second assumption required to write the GLE for interacting particles as in Eq. (2.1) is that $\underset{\sim}{\gamma}(t)$ and $\underset{\sim}{R}(t)$ are independent of the internuclear separation $\underline{R}$ ($\underline{R} = |\underset{\sim}{r}_2(t)-\underset{\sim}{r}_1(t)|$). On one level this assumption is clearly not true for two interacting particles. The solvent interaction with a free atom is significantly different from the solvent interaction with an atom in close proximity to another solute atom, and thus is clearly a function of internuclear separation. For the case of a true diatomic molecule, however, it is expected that $\underline{R}$ will vary rather slightly even for a vibrationally excited molecule, and the approximation that $\gamma(\underset{\sim}{t})$ and $R(\underset{\sim}{t})$ are independent of internuclear separation should not be a bad one. For the case of a homonuclear diatomic, $\gamma_{i\alpha}(t)$ is identical for each atom and it is subsequently referred to as $\gamma(t)$.

Finally, it should be noted that implicit in the generalized Langevin equation is the assumption of linear response, that the relaxation of a system from a state far from equilibrium has the same character as the relaxation from fluctuations which naturally occur

around equilibrium. Thus, under this assumption, only information about the equilibrium system is needed to model the solvent even when the solute molecule is far from equilibrium. However, the relaxation process simulated covers ~12,000 wave numbers of energy and includes vibrational amplitudes of up to ~1Å, and thus the assumption of solvent linear response may not be valid. The validity of this approximation will be evaluated for the system of iodine in liquid xenon later in this chapter.

Implementation of Eq. (2.1) for the calculation of vibrational relaxation requires a knowledge of the correct form for the memory function $\gamma(t)$. As was noted above, $\gamma(t)$ for a single particle is proportional to the random force autocorrelation function (FAF) (Eq. (2)). As will be shown below, the random FAF is directly related to the total FAF, a quantity which is easily calculated for a single particle from a molecular dynamics simulation. Therefore, it might be desirable to equate $\gamma(t)$ in Eq. (2.1) to the equilibrium memory function for a single atom. Consider for a moment an intuitive picture of a diatomic molecule immersed in a simple solvent. Each atom in the molecule interacts with the solvent with some potential, usually assumed to be independent of the position of the other solute atom. Thus the FAF, from which $\gamma(t)$ is derived, should be closely related to the FAF of a single atom immersed in the same solvent, with one major difference. The presence of the second atom provides a screening effect, essentially eliminating interaction with the solvent in one direction along the bond axis. This anisotropy is what leads to the pressure along the bond axis. It has already been discussed how these

cross-correlations can be ignored in this model, so instead of introducing an anisotropy into the solute-solvent interactions the screening effect is accounted for through modifying the magnitude of the single atom FAF by a factor of 1/2. The effect of this scaling factor on the vibrational relaxation rate should be equivalent to the effect of the anisotropy as long as the assumption that cross-correlations can be ignored is valid.

Recall from Eq. (2) that $\gamma(t)$ is proportional to the random (FAF) $\langle R(t')R(t'+t)\rangle$. Kubo has shown that the random FAF is related to the total FAF by[3]

$$\frac{1}{\gamma[\omega]} = \frac{1}{\gamma_T[\omega]} - \frac{1}{i\omega} \quad , \tag{2.3}$$

where $$\gamma[\omega] = \int_0^\infty \gamma(t)\exp(-i\omega t)dt$$

and $$\gamma_T[\omega] = \int_0^\infty \frac{\langle f(t')f(t'+t)\rangle}{k_b Tm}\exp(-i\omega t)dt \quad ,$$

with $\langle f(t')f(t'+t)\rangle$ being the total FAF. Note the high and low frequency limits of Eq. (2.3). At high frequencies, $\gamma[\omega] \approx \gamma_T[\omega]$, while at low frequencies $\gamma[\omega]$ is finite and $\gamma_T[\omega] \approx i\omega$. The low frequency limit implies that the total FAF $\langle f(t')f(t'+t)\rangle$ must integrate to zero. The total FAF for a single atom in solution is easily generated from a molecular dynamics simulation. Therefore, determination of $\gamma(t)$ for Eq. (2.1) proceeds as follows. A molecular

dynamics simulation of a single atom in solution is performed and the total FAF for the atom is calculated. Equation (2.3) is then used to extract the random FAF from the total FAF. Solution of Eq. (2.3) can be achieved through Fourier transform techniques since the correlation functions are all even in time. Equation (2) gives the memory function for the single atom, which is then scaled by a factor of 1/2 to obtain the desired $\gamma(t)$. The MD simulation is clearly a computationally intensive step in these calculations, especially since relevant statistics are collected on only a single atom in the entire simulation system. In fact, the calculation of single atom FAF's is the most time consuming step in the simulations of vibrational relaxation presented in this chapter.

The MTGLE work of Adelman and coworkers[31], consisting of a GLE-based model for chemical dynamics in solution including (but not limited to) vibrational relaxation[32,33], is quite similar to this work in both its methodology and applications, yet differs from this work both mathematically and conceptually on several points. Both models are based on the GLE and are formally equivalent in their treatment of dynamics of a <u>single</u> atoms, as was shown in the appendix to Cha. 3. In application to diatomic systems the two models differ in the treatment of **R**-dependence (internuclear separation) and cross-correlations between atom-solvent interactions. **R**-dependence is treated quite explicitly in the MTGLE model, with parameters for the chain equations being dependent on internuclear separation. The MTGLE model is designed to treat a variety of condensed phase reactions including "cage recombination", which requires realistic treatment of the solvent

over a wide range of internuclear separation. The relatively small variation in internuclear separation during vibrational relaxation warrants the simplified treatment of the model used here. Cross-correlations between atom-solvent interactions are also treated generally in the MTGLE model and ignored in this model. The present treatment is once again justified for the specific purpose of modeling vibrational relaxation, as will be shown in Sec. III. Conceptually, the two models differ significantly. The MTGLE model is appealing because of the close connection between chain "atoms" and solvation shell dynamics in solution. This model, on the other hand, retains the form of the GLE of Eq. (1) in which dynamical information about the solvent is contained in the memory function $\gamma(t)$. This leads conceptually to a more "collisional" view of the solvent dynamics, as will be discussed in Sec. III. Parameters for the MTGLE chain equations are determined from equilibrium pair distribution functions (PDF's) for the diatomic in solution. These PDF's can be either calculated explicitly through simulations of rigid diatomic molecules held fixed at <u>several</u> values of $\underline{R}$ (to give $\underline{R}$-dependent chain parameters), or approximated analytically, although the computational burden of explicitly calculating multiple PDF's has generally discouraged the use of simulation procedures. Parameters for this model are determined directly from equilibrium molecular dynamics FAF's for single atoms dissolved in the solvent of interest, with no adjustable parameters other than the MD potentials. Comparison with molecular dynamics results is therefore straightforward and leads to a clear indication of the validity of various approximations, including

linear response of the solvent, used in this model for vibrational relaxation. The model developed in this chapter is applied to the system of iodine in L-J xenon and compared quantitatively with molecular dynamics simulations of vibrational relaxation at identical densities and out to long times (500 ps). To my knowledge this the first such detailed quantitative comparison of MD and GBD vibrational dynamics which allows for effective evaluation of the validity of GBD models for vibrational relaxation.

III. MOLECULAR DYNAMICS SIMULATIONS

Molecular dynamics simulations of single atom FAF's for the system of iodine in L-J xenon follow closely the simulations of Brown et al.[18] In particular, the Xe-Xe and I-Xe potential parameters are identical. The densities considered are equivalent to the high and low densities simulated previously, 1.8 and 3.0 g/cm^3. These correspond to Lennard-Jones reduced densities ($\rho^* = \rho\sigma^3$) of .57 and .95, and are therefore considered moderate and high density liquids respectively. The system size is smaller in these simulations, 108 atoms instead of 257, because the larger system size is unnecessary in the calculation of equilibrium properties such as the total FAF. The solvent temperature in the simulations of Brown et al. was initially 280 K but was not constant throughout the relaxation process. The energy added to the iodine molecule to initiate the reaction was eventually distributed among the solvent atoms as the iodine relaxed, resulting in a bulk temperature rise of ~30 K upon completion of the relaxation process. The solvent

temperature used here is 300 K, corresponding to the solvent temperature of Brown et al. approximately half way through the vibrational relaxation process. For comparison sake, the time for this calculation is on the order of 1/100th the time required for the full molecular dynamics simulations (~ 40 trajectories, each calculated for 500 ps) of iodine vibrational relaxation at each density. Figure 1 shows a plot of both the random and total FAF's for an iodine atom in L-J xenon at the two densities simulated, $\rho^* = 0.57$ and 0.95.

The force autocorrelation function contains a variety of information about solvent structure and dynamics. The early time portion of the total FAF provides a picture of collisional dynamics of the solution, the form being roughly a function of the potential surfaces and the magnitude giving an indication of the collision frequency. Comparing early time portions of the total FAF's for the low and high density cases in Fig. 1 confirms this. The frequency of collisions is significantly higher in the high density case, accounting for the observed differences in magnitude. On the other hand, the nature of the collisions are expected to be only slightly modified at different densities, any differences being due to a change in the region of the potential surfaces involved in the collisions, and thus the shape of the early time regions is found to depend only slightly on density. The random FAF's have the same early time collision-based structure, giving a collisional basis to the friction integral in the GLE since the friction is calculated by convoluting the velocity of the particle with the random FAF. The longer time features of the total FAF give information on the collective collisions and structure of the

Figure 1. a) Random and total force autocorrelation functions for an iodine atom in Lennard-Jones xenon, $\rho^* = 0.57$ and T = 300K. Magnitudes are in units of ps^{-2}, corresponding to force squared divided by $k_b Tm$.

b) same as a) except $\rho^* = 0.95$.

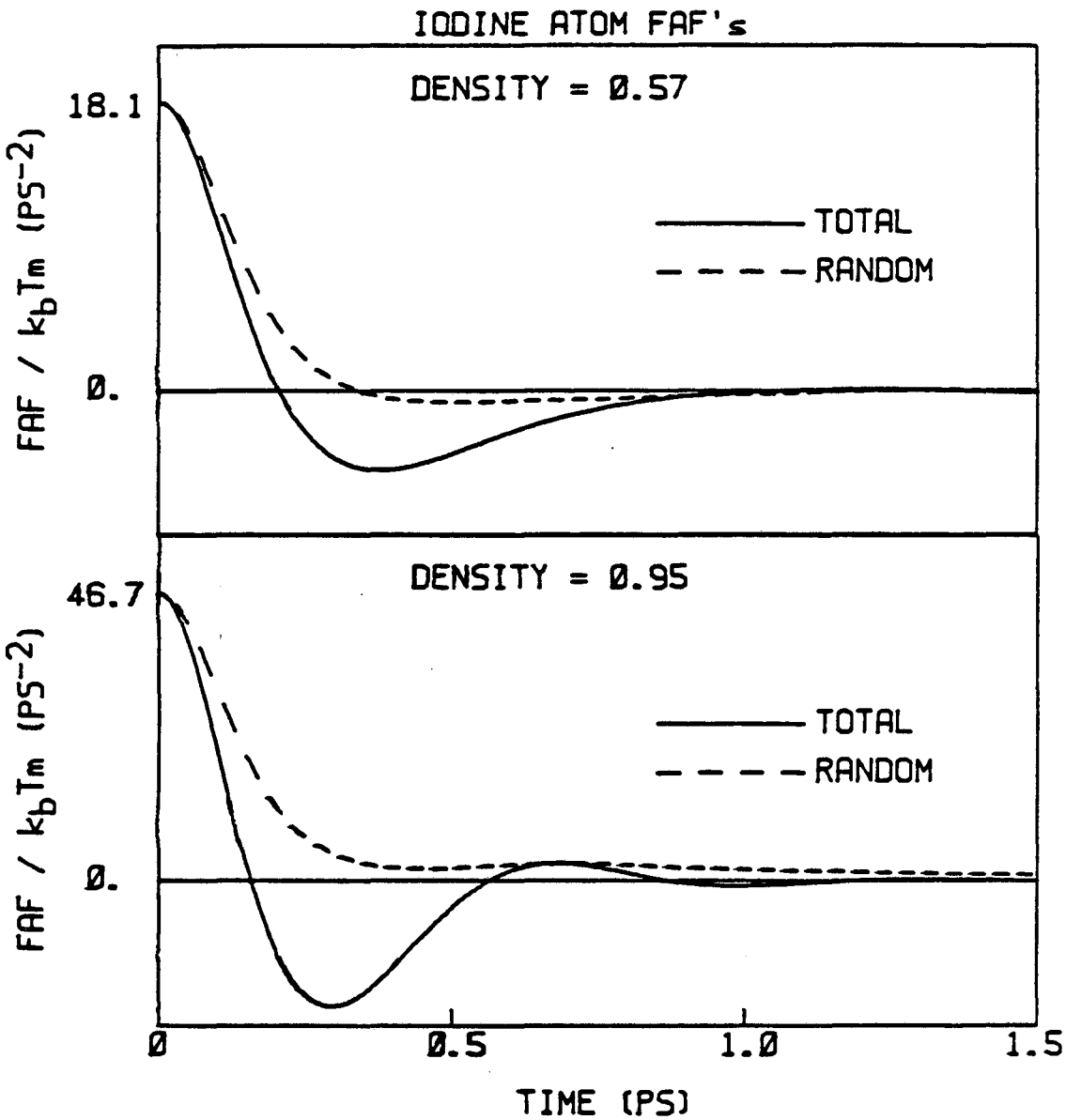
IODINE ATOM FAF's
DENSITY = 0.57
18.1
0.
FAF / $k_b T_m$ (PS^{-2})
TOTAL
RANDOM
DENSITY = 0.95
46.7
0.
FAF / $k_b T_m$ (PS^{-2})
TOTAL
RANDOM
0
0.5
1.0
1.5
TIME (PS)

Figure 1.

solvent. The negative region of the FAF indicates the occurrence of collisions on average from an opposite direction of the "initial" collision of the early time features. The negative feature of the high density FAF is deeper and shifted to earlier time than in the low density case, indicating a more compact and solid solvent cage structure. Extended oscillations, hinted at in the high density case, correspond to the atom oscillating repeatedly within the solvent cage. Thus this single function contains within its shape and magnitude a wealth of information about potential surfaces, density, temperature, local solvent structure, and the nature of collisional dynamics. Therefore, a vibrational relaxation model based on FAF's is well grounded intuitively as well as formally.

A molecular dynamics simulation of a rigid I_2 molecule in L-J xenon is used to evaluate the validity of the assumption that the solvent interacts with each atom independently. This simulation proceeds as above except that the iodine atom is replaced with an I_2 molecule held rigid at its equilibrium bond distance. System size is 109 atoms, with an I_2 molecule replacing the single iodine atom, and the density ($\rho^* = 0.57$) and temperature (300K) are identical. Let F_a be the force exerted by the solvent on one atom along the bond axis and F_b be the equivalent force on the second atom. Two self correlation functions and two cross correlations are calculated

$$
\begin{aligned}
C_{aa}(t) &= \langle F_a(t')F_a(t'+t)\rangle \\
C_{bb}(t) &= \langle F_b(t')F_b(t'+t)\rangle \\
C_{ab}(t) &= \langle F_a(t')F_b(t'+t)\rangle \\
C_{ba}(t) &= \langle F_b(t')F_a(t'+t)\rangle \quad . \qquad (3.1)
\end{aligned}
$$

By symmetry $C_{aa}(t) = C_{bb}(t)$ and $C_{ab}(t) = C_{ba}(t)$ for a homonuclear diatomic. Vibrational and center-of-mass FAF's for the rigid I_2 molecule are calculated from the relationships

$$C_{vib}(t) = C_{aa}(t) + C_{bb}(t) - C_{ab}(t) - C_{ba}(t)$$
$$C_{cm}(t) = C_{aa}(t) + C_{bb}(t) + C_{ab}(t) + C_{ba}(t) \qquad (3.2)$$

and shown in Fig. 2. Note that the center-of-mass FAF looks very much like the single atom FAF's including the long time negative tail. The integral of the center-of-mass FAF integrates to zero as in the single atom correlation functions. The vibrational correlation function looks much like the center-of-mass FAF in form except for an offset. The long time offset is due to the pressure in the system, a zero frequency force compressing the atoms together. The Fourier transform of these two correlation functions reveals that at frequencies greater than about 50 cm^{-1} or greater the vibrational and center-of-mass power spectra are identical. This indicates that the cross correlations, which account for the differences between the two spectra, are low frequency in nature. Since the iodine vibrational frequency is greater than 150 cm^{-1} throughout the relaxation process to be simulated, it is expected that the low frequency cross correlations would have little effect on the relaxation rates. This lends validity to the assumption that the atom-solvent interactions can be treated independently.

IV. AUTOREGRESSIVE MODELING OF $\gamma(t)$

Figure 2. Vibrational and center-of-mass FAF's for a rigid I_2 molecule in L-J xenon, $\rho^* = 0.57$ and T = 300K, calculated as in Eq. (3.2).

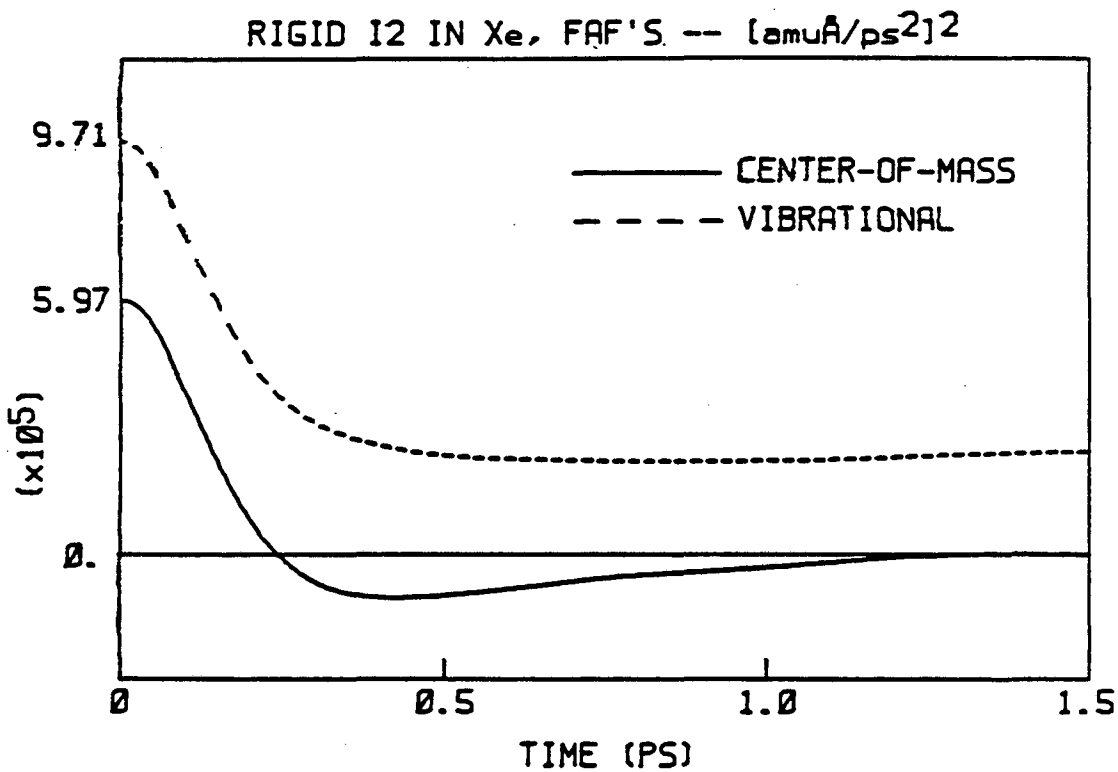
RIGID I2 IN Xe, FAF'S -- [amuÅ/ps2]2
9.71
5.97
Ø.
(x1Ø5)
CENTER-OF-MASS
VIBRATIONAL
Ø
Ø.5
1.Ø
1.5
TIME (PS)

Figure 2.

The general method for modeling $\gamma(t)$ which was developed in Cha. 3 is summarized in this section. This method, based on autoregressive techniques, provides both for computationally efficient evaluation of the friction integral in the GLE and for generation of a process R(t) with the desired correlation function. These techniques are derived from an extensive literature in the field of digital signal processing and spectrum estimation.[40,41]

The random force R(t) is modeled as an autoregressive (AR) process with the following generating function

$$x_n = -\sum_{j=1}^{p} a_j x_{n-j} + b_0 w_n \quad , \tag{4.1}$$

where p is the model order, x_n is a discrete process to be equated with R(t) and the a_j's are the AR parameters. w_n is a normalized gaussian distributed white noise source,

$$\langle w_n \rangle = 0.$$
$$\langle w_n w_{n+\tau} \rangle = \delta_\tau \quad ,$$

and b_0 is a factor which scales the magnitude of the correlation sequence associated with the process x_n.

Given a random force autocorrelation sequence γ_τ, where (γ_τ; τ = 0,1,2, ...) is a discrete representation of $\gamma(t)$, determination of the AR parameters which give

$$\gamma_\tau \approx r_\tau = \langle x_n x_{n+\tau} \rangle \quad ,$$

where the brackets refer to the time or ensemble average, is accomplished through a matrix solution of the Yule-Walker equations

$$\gamma_\tau \approx -\sum_{j=1}^{p} a_j \gamma_{\tau-j} \qquad \tau = 1,2,\cdots,N \quad . \tag{4.2}$$

Noting that $\gamma_\tau = \gamma_{-\tau}$, Eq. (4.2) is written in matrix notation as

$$\begin{bmatrix} \gamma_0 & \gamma_1 & \cdots\cdots & \gamma_{p-1} \\ \gamma_1 & \gamma_0 & \cdots\cdots & \gamma_{p-2} \\ \gamma_2 & \gamma_1 & \cdots\cdots & \gamma_{p-3} \\ \cdot & \cdot & \cdots\cdots & \cdot \\ \cdot & \cdot & \cdots\cdots & \cdot \\ \gamma_{N-1} & \gamma_{N-2} & \cdots\cdots & \gamma_{N-p} \end{bmatrix} \times \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \cdot \\ a_p \end{bmatrix} \approx - \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \cdot \\ \cdot \\ \gamma_N \end{bmatrix} , \tag{4.3}$$

or more briefly as

$$\underset{\approx}{M}\underset{\sim}{a} \approx -\underset{\sim}{\gamma} \quad . \tag{4.4}$$

N corresponds to the number of points in γ_τ used to determine the set $\{a_j\}$. For N = p, (4.4) is solved by

$$\underset{\sim}{a} = -\mathrm{inv}(\underset{\approx}{M})\underset{\sim}{\gamma} \quad . \tag{4.5}$$

If N > p there is in general no unique solution for $\underset{\sim}{a}$, and it is necessary to solve for the vector $\underset{\sim}{a}$ through some fitting procedure, for

example by minimizing

$$\sum_{\tau=1}^{N}(\epsilon_\tau)^2 \quad , \tag{4.6}$$

where $\epsilon_\tau = \gamma_\tau + a_1\gamma_{\tau-1} + \cdots + a_p\gamma_{\tau-p}$.

This minimization is accomplished using standard matrix techniques by

$$\underset{\sim}{a} = -\mathrm{inv}(\underset{\approx}{M}^T\underset{\approx}{M})\underset{\approx}{M}^T\underset{\sim}{\gamma} \quad . \tag{4.7}$$

In practice the matrix $\underset{\approx}{M}^T\underset{\approx}{M}$ is often near singular, and in the examples presented in this chapter quadruple precision has been used for the matrix inversion. Note also that this technique does not provide a best fit of the correlation sequence γ_τ but rather minimizes the sum in (4.6). This is desirable given the linear nature of the Yule-Walker equations (Eq. (4.2)) -- a direct fit to γ_τ is nonlinear and not at all straightforward to perform. The minimization of Eq. (4.6) unfortunately does <u>not</u> always provide a near best fit of γ_τ, particularly for large model orders or when N >> p. This has been discussed further in Sec. III of Cha. 3.

The resulting correlation sequence $r_\tau = \langle x_n x_{n+\tau}\rangle$ is found to be multiexponential,

$$r_\tau(p) = \sum_{j=1}^{p} A_j \exp(\omega_j \tau) \qquad \tau = 0,1,2,\cdots \quad , \tag{4.8}$$

with

$$A_j = \frac{b_0^2}{(1-\exp(2\omega_j)\prod_{\substack{k=1\\k\neq j}}^{p}(1-\exp(\omega_k)\exp(-\omega_j))(1-\exp(\omega_k)\exp(\omega_j))} \quad . \tag{4.9}$$

The frequencies ω_j are the stable roots of the polynomial

$$p(\omega) = 1 + a_1 e^{-i\omega} + \cdots + a_p e^{-ip\omega} \quad .$$

This multiexponential form for $r_\tau(p)$ provides the basis both for a computationally efficient algorithm for solving the friction integral in the GLE and for the generalization of the time step implicit in the discrete AR process.

Figure 3 shows the results of several AR fits of the random FAF for iodine in L-J xenon, $\rho^* = 0.57$, as calculated from Eq. (4.7). The results are plotted as a difference plot, $(r_\tau(p) - \gamma_\tau)$, since all of the fits reproduce the random FAF fairly well. Choice of the parameter N in the fitting procedure is actually quite important. For the problem of vibrational relaxation of a diatomic, where frequency of motion in the reaction is relatively high, it is much more critical to effectively model the short time features of the random FAF than the longer time features associated with low frequency motions in the solvent. Therefore the value of N should be kept relatively small, especially for lower order models. N is typically 30 in these simulations, corresponding to the first .3 picoseconds of the random FAF since the time between points δ in the autocorrelation sequence is 10 femtoseconds. The only exception is for the second order case, where simulations are run for both N = 30 and N = 10. Analysis of

Figure 3. AR fits of the random FAF, $\rho^* = 0.57$, calculated from Eq. (4.7) with N = 10 for 2nd order (o); and with N = 30 for 2nd order (x), 3rd order (dot-dashed), 4th order (dashed) and 6th order (solid).

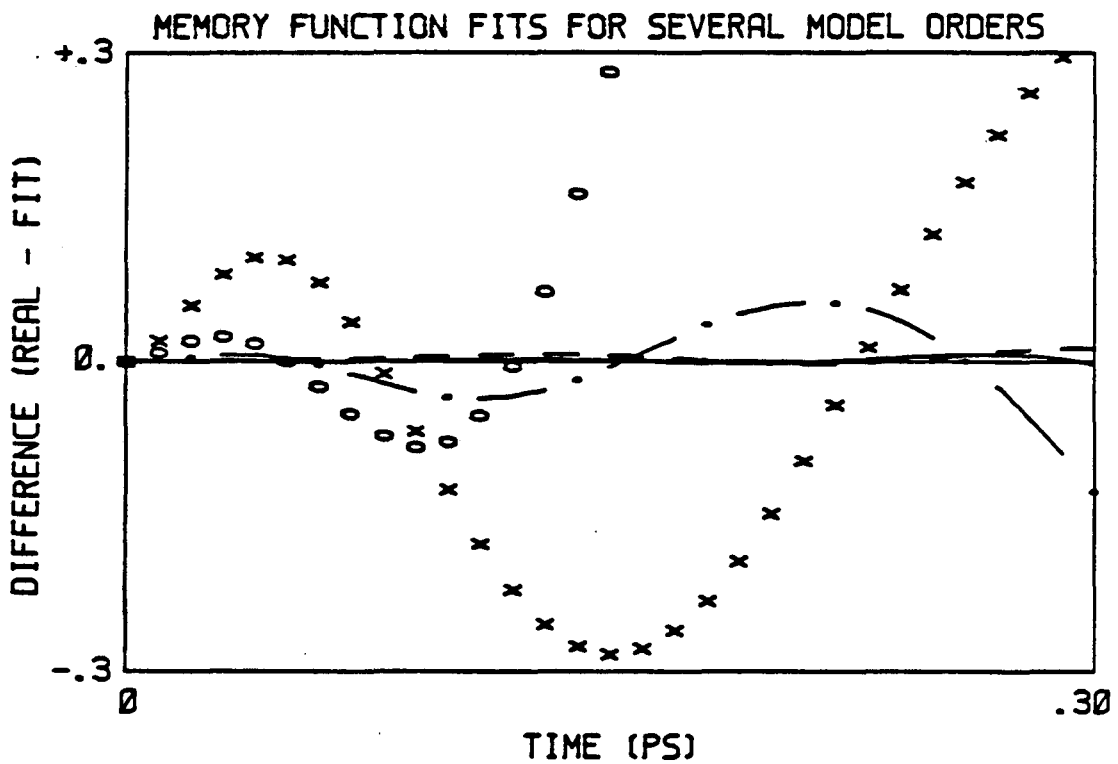
MEMORY FUNCTION FITS FOR SEVERAL MODEL ORDERS
+.3
0.
-.3
DIFFERENCE (REAL - FIT)
0
.30
TIME (PS)

Figure 3.

least squares fits for each order, calculated for the first 0.3 ps of the random FAF, is summarized in table I and shows clearly the improvement in fitting γ_τ as order is increased from p = 2 to p = 6. Above p = 6 the fits no longer improve, presumably since Eq. (4.7) does not directly fit $r_\tau(p)$ to γ_τ.

The AR procedure is discrete in nature and therefore, in order to proceed with simulations of vibrational relaxation, it is necessary to calculate AR parameters for the simulation time step h (h = 1 fs in all cases considered here). AR parameters corresponding to the plots in Fig. 3 are for the δ = 10 fs case. Calculation of the new parameters proceeds as described in Sec. IV(A) of Cha. 3. $r_\tau(p)$ for the h = 1 fs case is calculated by scaling the frequencies in Eq. (4.8) by a factor of h/δ

$$r_\tau(p) = \sum_{j=1}^{p} A_j \exp(\omega_j' \tau) \qquad \tau = 0,1,2,\cdots \quad , \qquad (4.10)$$

where

$$\omega_j' = (h/\delta)\omega_j \quad .$$

The new AR parameters are calculated from the relationship

$$1 + a_1' e^{-i\omega} + \cdots + a_p' e^{-ip\omega} = \prod_{j=1}^{p} (1-\exp(\omega_j')\exp(-i\omega)) \quad . \qquad (4.11)$$

The corresponding sequence

$$r'_\tau(p) = \sum_{j=1}^{p} A'_j \exp(\omega'_j \tau) \qquad \tau = 0,1,2,\cdots \quad , \qquad (4.12)$$

is not equal to $r_\tau(p)$ of Eq. (4.10) since the magnitudes A'_j are a function of $\{\omega'_j\}$, while the set $\{A_j\}$ are unchanged from the $\delta = 10$ fs case. The magnitude of $r'_\tau(p)$ in Eq. (4.12) is adjusted with the parameter b_0 (see Eq. (4.9)) such that

$$r'_0(p) = r_0(p) = \gamma_0$$

Analysis of fits for the resulting sequences $r'_\tau(p)$ is also tabulated in table I and, as can be seen, agreement has improved for model orders three and four, and is significantly worse for the $p = 6$ case. The relatively poor results for the $p = 6$ case are not particularly disturbing. For simulations of vibrational relaxation far from equilibrium the results are much more sensitive to the form of the memory function in the friction integral than to the form of the random force. This is because the typically large velocity of the excited oscillator causes the friction term in the GLE to dominate the random force term. The memory function used in these simulations can be taken equally well from Eq. (4.10) or Eq. (4.12), and in fact whichever is better is used, while the random force will in every case have the correlation sequence of Eq. (4.12).

V. RESULTS

TABLE I. Sum of squared residuals $(r_\tau(p) - \gamma_\tau)^2$ for several model order fits to the random FAF calculated for two different time steps h. N = 30 in all cases, and the residuals are summed over $\tau = 0,1, \ldots ,N$. AR parameters for the h = 10 fs case are calculated as in Eq. (4.7), while for the h = 1 fs case they are calculated as in Eq. (4.12).

	Sum of squared residuals (ps^{-4})	
Model order	h = 10 fs	h = 1 fs
2	9.794 x10-1	1.062
3	5.381 x10-2	3.967 x10-3
4	2.233 x10-2	5.212 x10-4
6	1.135 x10-4	6.180 x10-2

Results of vibrational relaxation simulations of iodine in liquid xenon are described in this section. The proper choice of AR model order is first determined from several preliminary simulations of the $\rho^* = 0.57$ case with no random forces included in the GLE's. Results of full simulations for both densities are then presented in comparison with molecular dynamics results from Brown et al.[18]

Simulations of vibrational relaxation proceed as follows. The iodine molecule is initially placed on the I_2 RKR ground state potential with ~11,500 wave numbers of energy (approximately 1000 wave numbers below the dissociation energy) and with random phase. The equations of motion in (2.1) are then integrated numerically using a second order integrator described by Beeman[42] with a time step of 1 femtosecond. The friction integrals are evaluated by the technique described in Sec. IV(B) of Cha. 3 (with a second order integrator) for a multi-exponential memory function, with the particular memory function determined from Eqs. (4.10) or (4.12), whichever provides the better representation of $\gamma(t)$. $F_{i\alpha}(|\underset{\sim}{r}_2(t)-\underset{\sim}{r}_1(t)|)$ is calculated from the slope of the RKR potential at the I-I Internuclear separation. The random forces $R_{i\alpha}(t)$ are calculated independently for each coordinate as in Eq. (4.1). Initialization of the memory terms in Eq. (2.1) is essentially ignored, with the systematic damping force being calculated directly from (2.1) with velocities before t=0 being set to zero. This should only affect dynamics for approximately one picosecond since the memory function is essentially zero by that time (Fig. 1), and thus has a negligible effect on the total relaxation dynamics which are calculated for 500 picoseconds. The equations of motion are

calculated independently for each iodine atom, except for the I-I interaction. Trajectories were integrated to 500 ps for both densities and were used to calculate time dependent average vibrational energies.

The appropriate AR model order for these simulations is chosen by performing preliminary simulations of vibrational relaxation without including the random forces in Eq. (2.1). Exclusion of the random forces eliminates noise in the relaxation results and therefore allows detailed comparison of single trajectories, thus saving considerable simulation time. Inclusion of the random forces, as expected, considerably affects the distribution of vibrational relaxation rates, yet has a very small effect on the average relaxation. Results for the $\rho^* = 0.57$ case, plotted as the average vibrational energy as a function of time, are shown in Fig. 4 for several models of $\gamma(t)$. Two of the relaxation curves shown are for second order memory models, the difference being in the value of N used in the fitting procedure. The value of N determines the region of the random FAF used in the calculation of the AR parameters, with a value of 30 for N corresponding to the first 0.3 picoseconds of the FAF. On comparison with results for higher order memory models it can be seen that neither of the two parameter models lead to an appropriate form for the relaxation, taking the higher order results as "correct". Results for model orders of three and four are very close yet still distinguishable. Relaxation for model orders higher than four are indistinguishable from fourth order and are not shown. On the basis of this convergence, it appears that a model order of four is adequate to describe the dynamics of this system, and all subsequent simulations

Figure 4. Results of generalized Langevin simulations with no random force, $\rho^* = 0.57$, calculated for 2nd order (N = 10 and N = 30), 3rd order (N = 30), and 4th order (N = 30) AR memory function models.

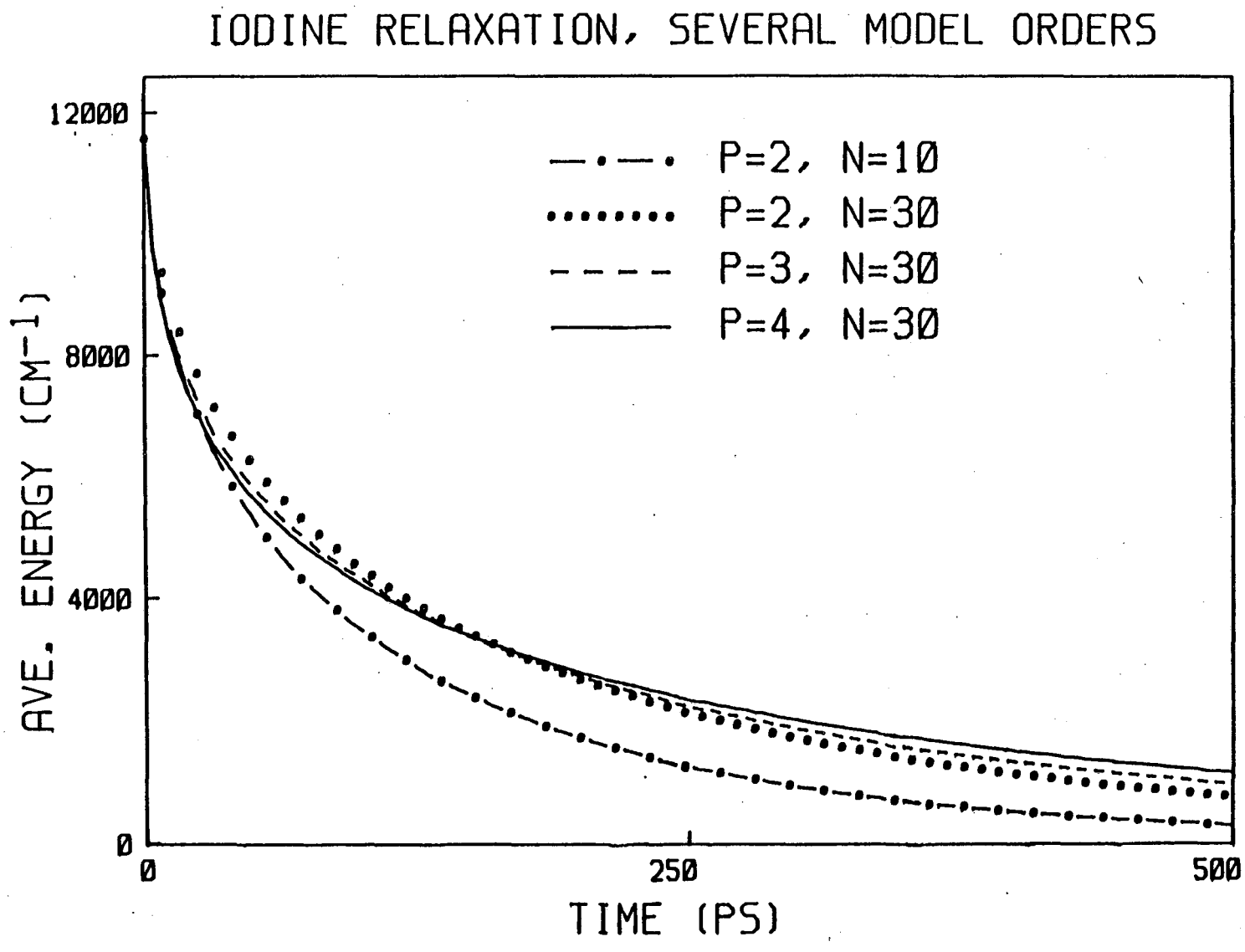
IODINE RELAXATION, SEVERAL MODEL ORDERS
P=2, N=10
P=2, N=30
P=3, N=30
P=4, N=30
AVE. ENERGY (CM-1)
12000
8000
4000
0
0
250
500
TIME (PS)

Figure 4.

will therefore use fourth order AR models for the memory functions. Failure of lower order models is probably due primarily to the anharmonicity of the I_2 potential, where a wide range of frequencies (150 to 214 wave numbers) are found in the energy regions simulated.

Full simulations of vibrational relaxation using fourth order AR memory function models and including random forces have been performed for iodine in liquid xenon at reduced densities of 0.57 and 0.95. 62 trajectories are calculated for each density and compared in each case with ~40 molecular dynamics trajectories. Results are shown in Fig. 5. As can be seen, agreement is reasonably good in both cases, though not exact. Differences from MD results are small enough, given the noise in the simulations, to discourage attempts to specifically identify the source of the deviations. For example, a breakdown in the validity of the linear response approximation would be expected to cause greater deviation at higher vibrational energies than at energies near equilibrium, yet it is impossible to identify any such trend from the data presented. The conclusion that can be drawn from the data is that the major approximation of the model, that relevant solvent information for vibrational relaxation is contained in <u>equilibrium</u> <u>atomic</u> FAF's, appears to be reasonable, at least for the specific case of iodine in L-J xenon.

VI. DISCUSSION

The results presented in Sec. V show very good agreement between the simple model for vibrational relaxation presented in this chapter

Figure 5. Comparison of GBD results with molecular dynamics from Brown et al. for low ($\rho^* = 0.57$) and high ($\rho^* = 0.95$) density cases. Autoregressive parameters are calculated for a model order of four and N = 30 for both densities.

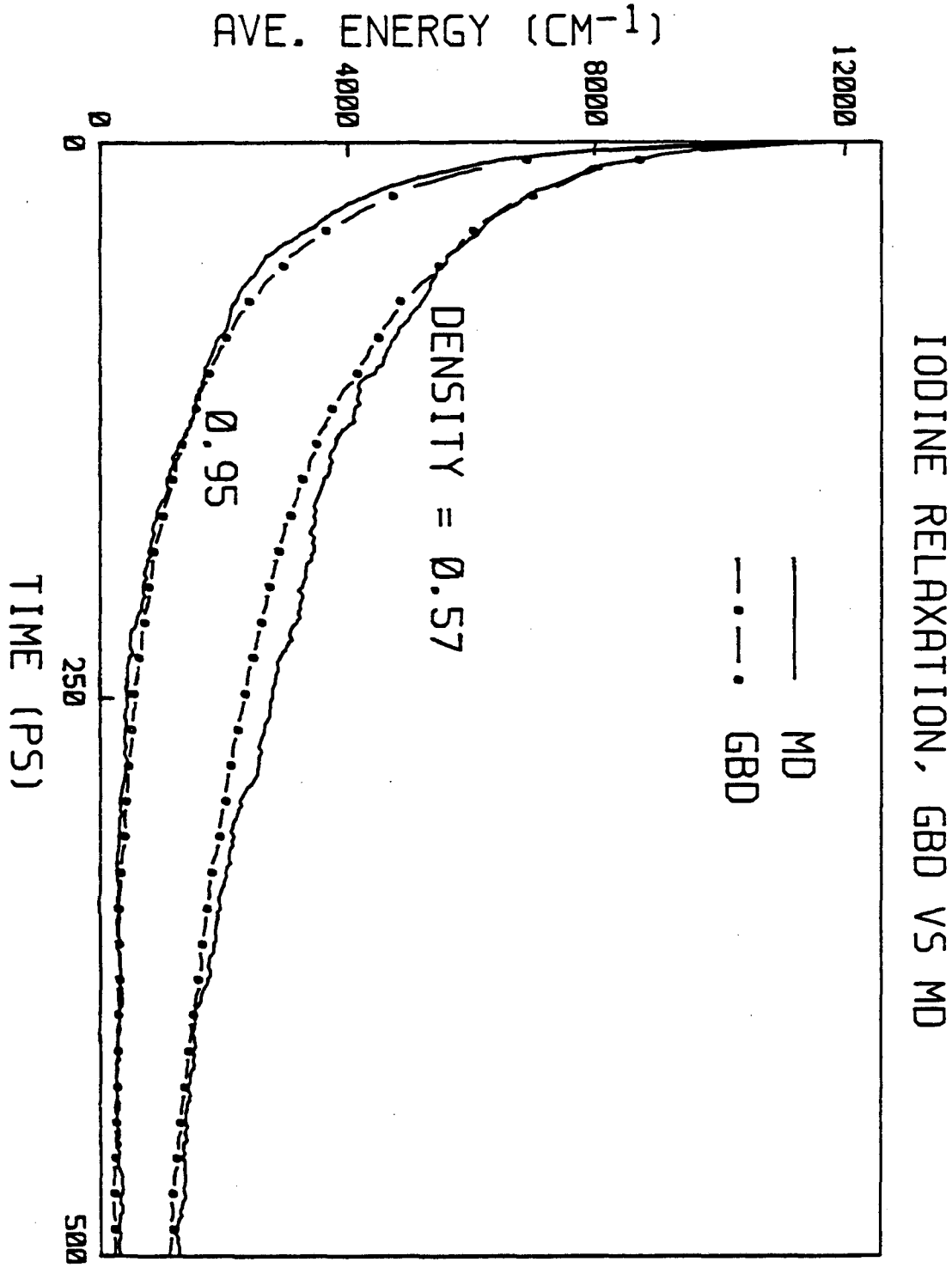
IODINE RELAXATION, GBD VS MD
MD
GBD
DENSITY = 0.57
0.95
AVE. ENERGY (CM-1)
12000
8000
4000
0
0
250
500
TIME (PS)

Figure 5.

and molecular dynamics simulations. To my knowledge, this is the first such detailed comparison made between a GLE-based model for vibrational relaxation and molecular dynamics. It should be stressed that there are no adjustable parameters in this model, all results being directly derived from the MD potential parameters. Such good agreement therefore lends credence to the techniques of GBD which are grounded in a linear response assumption. The iodine vibrational motion, especially high up in the well, is extremely rapid and large in amplitude (~1 angstrom), yet an effective representation of the solvent interaction with this motion still may be derived from equilibrium solute/solvent properties. The computational burden of this simple model lies primarily in the calculation of the equilibrium atomic FAF's from MD simulations, which takes on the order of 1/100th of the time for full MD simulations of the vibrational relaxation process. The computational efficiency of the GBD technique is therefore clearly demonstrated.

It is also quite interesting that relevant information about diatomic/solvent interactions appears to be contained within single atom FAF's. This suggests that the nature of solvent interactions with a single atom of a diatomic is not strongly affected by the second atom, the presence of which can therefore be accounted for with a trivial screening effect. This is not terribly surprising considering that atomic FAF's contain relevant information about both solvent structure and collisional dynamics which perhaps should not be significantly modified for a diatomic. The flexibility of this model for vibrational relaxation rests on this ability to use single atom

FAF's, since the input correlation functions can be either efficiently calculated or analytically approximated.

Finally, it would be interesting to extend the scope of this model to include more complicated polyatomic systems. The primary difficulty in this extension comes in the determination and application of appropriate screening factors. Consider the next simplest case of a linear triatomic, which has several (4) vibrational modes. Simple geometric considerations suggest that, although solvent interaction with the end atoms in the direction of the bond axis should be mediated by the presence of other solute atoms as in the diatomic case, yet interaction perpendicular to the bond axis, which couples to the vibrational bending modes, should effectively encounter no screening at all. Thus it would become necessary to use a molecule-based reference frame and to apply the solvent forces differently to each axis on each atom. Application to more complicated polyatomics would require additional development of a more rigorous method for the determination of screening factors given the resulting complicated molecular geometries.

In summary, a simple classical stochastic model for diatomic vibrational relaxation based on the generalized Langevin equation has been presented. The memory function in the generalized Langevin equation is determined directly from equilibrium force autocorrelation functions for the individual atoms of the diatomic dissolved in the solvent of interest. Simulations of vibrational relaxation have made use of a simple autoregressive procedure for modeling the memory functions to arbitrary order, developed in Cha. 3, and shown that for

the case at hand a model order of four is sufficient for applications to vibrational relaxation. This model has been tested on the system of iodine in Lennard-Jones xenon and found to be very effective in reproducing data from molecular dynamics simulations at two very different densities. It would be interesting to extend these simulations to a system with significantly different potential surfaces to test the generality of these techniques, for example the relaxation of iodine in liquid argon, as well as to polyatomic systems.

References

1. H. Mori, Progr. Theor. Phys. 33, 423 (1965).
2. R. Zwanzig, Ann. Rev. Phys. Chem. 16, 67 (1965).
3. R. Kubo, Rep. Prog. Theor. Phys. 29, 255 (1966).
4. D. W. Oxtoby, Adv. Chem. Phys. 47, 487 (1981).
5. J. Chesnoy and G. M. Gale, Ann. Phys. Fr. 9, 893 (1984).
6. D. J. Diestler and R. S. Wilson, J. Chem. Phys. 62, 1572 (1975).
7. A. Nitzan and R. S. Silbey, J. Chem. Phys. 60, 4070 (1974).
8. D. W. Oxtoby and S. A. Rice, Chem. Phys. Lett. 42, 1 (1976).
9. D. J. Diestler, Chem. Phys. Lett. 39, 39 (1976).
10. R. N. Schwartz, Z. I. Slawsky, and K. F. Herzfeld, J. Chem. Phys. 20, 1951 (1952).
11. M. Fixman, J. Chem. Phys. 34, 369 (1961).
12. R. Zwanzig, J. Chem. Phys. 34, 1931 (1961); 36, 2227 (1962).
13. K. F. Herzfeld, J. Chem. Phys. 36, 3305 (1962).
14. P. K. Davis and I. Oppenheim, J. Chem. Phys. 57, 505 (1972).
15. H. K. Shin and J. Keizer, Chem. Phys. Lett. 27, 611 (1974).
16. D. W. Oxtoby, Mol. Phys. 34, 987 (1977).
17. R. Granek, A. Nitzan, and E. Weitz, J. Chem. Phys. 89, 5589 (1988).
18. J. K. Brown, C. B. Harris, and J. C. Tully, J. Chem. Phys. 89, 6687 (1988).
19. B. L. Holian, J. Chem. Phys. 84, 3138 (1986).
20. P. B. Visscher and B. L. Holian, J. Chem. Phys. 89, 5128 (1988).
21. J. Chesnoy and J. J. Weis, J. Chem. Phys. 84, 5378 (1986).
22. M. F. Herman, J. Chem. Phys. 87, 4779, 4794 (1986).
23. R. P. Parson and E. J. Heller, J. Chem. Phys. 85, 2569, 2581 (1986).

24. P. S. Dardi and R. I. Cukier, J. Chem. Phys. 86, 2264 (1987).

25. S. A. Adelman and R. H. Stote, J. Chem. Phys. 88, 4397 (1988).

26. R. H. Stote and S. A. Adelman, J. Chem. Phys. 88, 4415 (1988).

27. H. Metiu, D. W. Oxtoby, and K. F. Freed, Phys. Rev. A 15, 361 (1977).

28. S. Velsko and D. W. Oxtoby, J. Chem. Phys. 72, 2260 (1980).

29. J. K. Brown, D. J. Russell, D. E. Smith, and C. B. Harris, Rev. Phys. Appl. 22, 1787 (1987).

30. C. B. Harris, J. K. Brown, M. E. Paige, D. E. Smith, and D. J. Russell, in Ultrafast Phenomena, V, edited by G. R. Fleming and A. E. Siegman, Springer Ser. Chem. Phys. 46, (Springer, New York, 1986), p. 326.

31. S. A. Adelman, Adv. Chem. Phys. 53, 61 (1983), and references therein.

32. C. L. Brooks, M. W. Balk, and S. A. Adelman, J. Chem. Phys. 79, 784 (1983).

33. C. L. Brooks and S. A. Adelman, J. Chem. Phys. 80, 5598 (1984).

34. A. L. Harris, J. K. Brown, and C. B. Harris, Ann. Rev. Phys. Chem., 39, 341 (1988) and references therein.

35. M. E. Paige, D. J. Russell, and C. B. Harris, J. Chem. Phys. 85, 3699 (1986).

36. M. E. Paige and C. B. Harris, (in preparation).

37. G. Ciccotti and J.-P. Ryckaert, J. Stat. Phys. 26, 73 (1981).

38. G. Bossis, B. Quentrec, and J. P. Boon, Mol. Phys. 45, 191 (1982).

39. W. F. van Gunsteren and H. J. C. Berendsen, Mol. Phys. 47, 721 (1982).

40. J. A. Cadzow, Proc. IEEE 70, 907 (1982), and references therein.

41. L. R. Rabiner and B. Gold, Theory and Applications of Digital Signal Processing, (Prentice-Hall, New Jersey, 1975).

42. D. Beeman, J. Comp. Phys. 20, 130 (1976).

APPENDIX

I. PROGRAM LISTING "IATOM"

```
C   program "IATOM"
C   David Smith
C   Aug. 16, 1989
C
C   This program calculates the total acceleration autocorrelation
C   function (FAF) for a single iodine atom dissolved in Lennard-Jones
C   xenon. The subroutines of this program are taken in their
C   entirety, with the exceptions of fcorr(), wfcor() and tempc(), from
C   the program "I2XENON" by J. Keenan Brown, modified to calculate
C   the motion of only a single atom rather than a diatomic. Since
C   this program is derived directly from "I2XENON", the comments are
C   rather brief. The reader is referred to J. K. Brown's PhD thesis
C   for more details on any given portion of the program. This program
C   is designed to run on the Cray X-MP/48 at the San Diego
C   Supercomputer Center with the CFT Fortran compiler. The input
C   file 'p108' contains the positions and velocities of 108
C   particles at a given density. Some input files are already
C   equilibrated for a temperature of 300 Kelvin, yet the program is
C   written in such a way that equilibration is accomplished for any
C   reasonable input file.

      parameter(ntm=108,ntm1=ntm-1,ntm3=3*ntm,nn3=ntm*ntm1/2)
      parameter(nn4=3*nn3)
      parameter(ncr=256)

      common /corp/kn,ncor,iflag,nstep
      common /bk1/r2t(nn3),rxt(nn3),ryt(nn3),rzt(nn3),rtxyz(nn4)
      common /blk1/alxyz,c1,c2,smass
      common /blk2/h,h2,h26,hi
      common /blk4/v(ntm3)
      common /blk5/r(ntm3),r2(ntm3),a(ntm3),a2(ntm3)
      common /blk6/c1d,c2d,inum(ntm1),icord(nn3),r2cut,r2max
      common /blk7/rmin
      common /blk8/vzero
      common /blk9/natom,natom1,natom3
      common /i21/c1i2,c2i2,c3i2,c4i2,c1i2d,c2i2d,c3i2d,c4i2d
      common /i22/vzix,smassi,r2maxi,r2cuti,rnewix
      common /rand/xseed
      common /stin/rzero,rmax,elj,sigma
      common /fcor/fsave(3,ncr),corf(ncr)
C
C   The following command is specific to the CFT compiler on the Cray
C   X-MP/48. It must be the first executable line in the program, as
C   it is responsible for defining all of the I/O unit numbers and for
C   opening the necessary files. The file names can be changed when
C   the program is executed. 'out6' contains the FAF output, 'o9'
```

```
C   contains energy information output during the trajectory, and
C   'p108' is the input file.

      call link("unit6=(out6,create,text),
     $ unit2=(p108,open,text),
     $ unit9=(o9,create,text)//")

      call g05ccf
      xseed=0.
C
C   'ncor' is the number of points in the correlation function.
C   Typically the correlation functions are calculated out to 2.56
C   picoseconds; 256 points at 10 fs per point.  'iflag' is a flag used
C   in the initialization of the FAF's.  'corf' contains the
C   correlation function, and 'fsave' contains the past values of the
C   accelerations used in the calculation of the FAF.
C
      ncor=256
      kn=ncor
      iflag=0

      do 871 i=1,256
 871  corf(i)=0.

      call stinfo

      ktm=0
      akt=300.*.831*1.5*natom

      call sort(r)
      call energy(ek,ep,etot,v,r)
      write(6,99) ktm,ek,ep,etot
 99   format(i6,3e15.6)
C
C   initial temperature equilibration
C
      do 200 i=1,100

        call tempc(ek,akt,v)
        call sort(r)

        do 210 j=1,10
        call integ(ktm)
 210    continue

        call energy(ek,ep,etot,v,r)
        write(6,99) ktm,ek,ep,etot
 200  continue

      call empty(6)
C
```

```
C   loops to equilibrate the temperature carefully.  The total kinetic
C   energy of the system is averaged over 20,000 time steps, and this
C   average is used to determine the temperature.  The kinetic energies
C   of the particles are then scaled to adjust the temperature toward
C   the desired value.  After several iterations the temperature was
C   always found to be within one degree of the desired value.  The
C   temperature is written to the output file at the end of each large
C   iteration.
      do 301 ll=1,5

      eks = 0.
      do 300 i=1,1000

        call sort(r)

        do 310 j=1,10
        call integ(ktm)
 310    continue

        call energy(ek,ep,etot,v,r)
        write(6,99) ktm,ek,ep,etot

        eks = eks + ek

 300  continue

      eks = eks/1000.
      call tempc(eks,akt,v)
      write(9,*) eks, eks*2./108./3./.831

 301  continue

      call empty(9)
C
C   Writing out the positions and velocities of each particle allows
C   for future trajectories to be run, with this file as input, without
C   the extensive temperature equilibration performed above.
C
      do 335 i=1,natom3,3
      write(9,98) (r(i+j), j=0,2)
      write(9,98) (v(i+j), j=0,2)
 335  continue
 98   format(3e15.7)

C   section to initialize fsave, the array containing the past history
C   of the accelerations.  Requires at least 'ncor' calls to fcorr().

      do 600 i=1,26

        call sort(r)

        do 610 j=1,10
```

```
        call integ(ktm)
        call fcorr
 610    continue
 600  continue
C
C   'iflag' is reset so that the correlation function is calculated
C   when fcorr() is called.

      iflag=1

      call empty(6)
C
C   'nstep'*10 equals the number of calls to fcorr.  The temperature is
C   calculated throughout the trajectory by summing the total kinetic
C   energies as above.
C
      nstep = 2000
      eks = 0.
      do 700 i=1,nstep/10

        do 705 i1=1,10

        call sort(r)

        do 710 j=1,10
        call integ(ktm)
        call vibcor
 710    continue

 705    continue

      call energy(ek,ep,etot,v,r)
      write(6,99) ktm,ek,ep,etot
      eks = eks + ek

 700  continue
C
C   write out the system temperature.
C
      eks = eks * 10. / float(nstep)
      write(9,*) eks, eks*2./108./3./.831
C
C   write out the correlation functions.
      call wfcorr()
C
C   exit the program and close all I/O connections
      call exit

      end

C  ****************************
      subroutine integ(ktm)
```

```
C  ****************************

      parameter(ntm=108,ntml=ntm-1,ntm3=3*ntm,nn3=ntm*ntml/2)
      parameter(nn4=3*nn3)

      common /blk2/h,h2,h26,hi
      common /blk4/v(ntm3)
      common /blk5/r(ntm3),r2(ntm3),a(ntm3),a2(ntm3)
      common /blk9/natom,natoml,natom3

      ktm=ktm+2

      do 100 i=1,natom3
 100  r2(i)=r(i)+h*v(i)+h26*(4.*a(i)-a2(i))

      call accel(a2,r2)

      do 110 i=1,natom3
      r(i)=(r2(i)-r(i)+h26*(2.*a2(i)+a(i)))*hi
 110  r(i)=r2(i)+h*r(i)+h26*(4.*a2(i)-a(i))

      call accel(a,r)

      do 120 i=1,natom3
 120  v(i)=(r(i)-r2(i)+h26*(2.*a(i)+a2(i)))*hi

      return
      end

C  **************************
      subroutine accel(a,r)
C  **************************

      parameter(ntm=108,ntml=ntm-1,ntm3=3*ntm,nn3=ntm*ntml/2)
      parameter(nn4=3*nn3)

      common /bkl/r2t(nn3),rxt(nn3),ryt(nn3),rzt(nn3),rtxyz(nn4)
      common /blkl/alxyz,cl,c2,smass
      common /blk2/h,h2,h26,hi
      common /blk4/v(ntm3)
      common /blk6/cld,c2d,inum(ntml),icord(nn3),r2cut,r2max
      common /blk7/rmin
      common /blk8/vzero
      common /blk9/natom,natoml,natom3
      common /i21/cli2,c2i2,c3i2,c4i2,cli2d,c2i2d,c3i2d,c4i2d
      common /i22/vzix,smassi,r2maxi,r2cuti,rnewix
      common /stin/rzero,rmax,elj,sigma

      dimension a(ntm3),r(ntm3)

      call ssort(r,rmin,rnewix,knum)
```

```
      k=0
      do 100 i=1,l
      do 110 j=1,inum(i)
      k=k+1
      temp=1./r2t(k)**4
      r2t(k)=temp*(c3i2d*r2t(k)*temp-c4i2d)
      rxt(k)=rxt(k)*r2t(k)
      ryt(k)=ryt(k)*r2t(k)
110   rzt(k)=rzt(k)*r2t(k)
100   continue

      do 120 i=k+1,knum
      temp=1./r2t(i)**4
      r2t(i)=temp*(c1d*r2t(i)*temp-c2d)
      rxt(i)=rxt(i)*r2t(i)
      ryt(i)=ryt(i)*r2t(i)
120   rzt(i)=rzt(i)*r2t(i)

      do 121 i=1,natom3
121   a(i)=0.
      k=0
      do 131 i=1,l
      i3=3*i
      i2=i3-1
      i1=i2-1
      do 141 j=1,inum(i)
      k=k+1
      j3=3*icord(k)
      j2=j3-1
      j1=j2-1
      a(i1)=a(i1)+rxt(k)
      a(i2)=a(i2)+ryt(k)
      a(i3)=a(i3)+rzt(k)
      a(j1)=a(j1)-rxt(k)
      a(j2)=a(j2)-ryt(k)
      a(j3)=a(j3)-rzt(k)
141   continue
131   continue

      do 130 i=2,natoml
      i3=3*i
      i2=i3-1
      i1=i2-1
      do 140 j=1,inum(i)
      k=k+1
      j3=3*icord(k)
      j2=j3-1
      j1=j2-1
      a(i1)=a(i1)+rxt(k)
      a(i2)=a(i2)+ryt(k)
      a(i3)=a(i3)+rzt(k)
      a(j1)=a(j1)-rxt(k)
```

```
      a(j2)=a(j2)-ryt(k)
      a(j3)=a(j3)-rzt(k)
 140  continue
 130  continue

      do 150 i=1,3
 150  a(i)=a(i)/smassi
      do 160 i=4,natom3
 160  a(i)=a(i)/smass

      return
      end

C  ************************
      subroutine sort(r)
C  ************************

      parameter(ntm=108,ntm1=ntm-1,ntm3=3*ntm,nn3=ntm*ntm1/2)
      parameter(nn4=3*nn3)

      common /bk1/r2t(nn3),rxt(nn3),ryt(nn3),rzt(nn3),rtxyz(nn4)
      common /blk1/alxyz,c1,c2,smass
      common /blk2/h,h2,h26,hi
      common /blk4/v(ntm3)
      common /blk6/c1d,c2d,inum(ntm1),icord(nn3),r2cut,r2max
      common /blk7/rmin
      common /blk8/vzero
      common /blk9/natom,natom1,natom3
      common /i21/c1i2,c2i2,c3i2,c4i2,c1i2d,c2i2d,c3i2d,c4i2d
      common /i22/vzix,smassi,r2maxi,r2cuti,rnewix
      common /stin/rzero,rmax,elj,sigma

      dimension r(ntm3)

      k=0
      do 30 i=1,natom3-3,3
      i2=i+1
      i3=i2+1
      do 40 j=i+3,natom3,3
      j2=j+1
      j3=j+2
      rx=r(i)-r(j)
      ry=r(i2)-r(j2)
      rz=r(i3)-r(j3)
      rx=rx-alxyz*anint(rx/alxyz)
      ry=ry-alxyz*anint(ry/alxyz)
      rz=rz-alxyz*anint(rz/alxyz)
      k=k+1
 40   r2t(k)=rx**2+ry**2+rz**2
 30   continue

      k=0
```

```
      kk=0
      inum(1)=0

      do 50 i=1,1
      do 60 j=2,natom
      k=k+1
      if (r2t(k).gt.r2maxi) goto 60
      inum(i)=inum(i)+1
      kk=kk+1
      icord(kk)=j
 60   continue
 50   continue

      do 70 i=2,natom1
      inum(i)=0
      do 80 j=i+1,natom
      k=k+1
      if (r2t(k).gt.r2max) goto 80
      inum(i)=inum(i)+1
      kk=kk+1
      icord(kk)=j
 80   continue
 70   continue

      return
      end

C  ****************************************
      subroutine ssort(r,rnew,rneix,knum)
C  ****************************************

      parameter(ntm=108,ntm1=ntm-1,ntm3=3*ntm,nn3=ntm*ntm1/2)
      parameter(nn4=3*nn3)

      common /bk1/r2t(nn3),rxt(nn3),ryt(nn3),rzt(nn3),rtxyz(nn4)
      common /blk1/alxyz,c1,c2,smass
      common /blk2/h,h2,h26,hi
      common /blk4/v(ntm3)
      common /blk6/c1d,c2d,inum(ntm1),icord(nn3),r2cut,r2max
      common /blk7/rmin
      common /blk8/vzero
      common /blk9/natom,natom1,natom3
      common /i21/c1i2,c2i2,c3i2,c4i2,c1i2d,c2i2d,c3i2d,c4i2d
      common /i22/vzix,smassi,r2maxi,r2cuti,rnewix
      common /stin/rzero,rmax,elj,sigma

      dimension r(ntm3)

      k=1
      kk=0
      do 10 i=1,natom1
      ia=3*i
```

```
      rtxyz(k)=r(ia-2)
      rtxyz(k+1)=r(ia-1)
      rtxyz(k+2)=r(ia)
      k=k+3

      do 20 j=1,inum(i)
      kk=kk+1
      ic=3*icord(kk)
      rtxyz(k)=r(ic-2)
      rtxyz(k+1)=r(ic-1)
      rtxyz(k+2)=r(ic)
20    k=k+3
10    continue

      k=0
      kk=1
      do 30 i=1,natom1
      i1=kk
      i2=kk+1
      i3=kk+2
      kk=kk+3
      do 40 j=1,inum(i)
      j1=kk
      j2=kk+1
      j3=kk+2
      kk=kk+3
      k=k+1
      rxt(k)=rtxyz(i1)-rtxyz(j1)
      ryt(k)=rtxyz(i2)-rtxyz(j2)
      rzt(k)=rtxyz(i3)-rtxyz(j3)
      rxt(k)=rxt(k)-alxyz*anint(rxt(k)/alxyz)
      ryt(k)=ryt(k)-alxyz*anint(ryt(k)/alxyz)
      rzt(k)=rzt(k)-alxyz*anint(rzt(k)/alxyz)
40    r2t(k)=rxt(k)**2+ryt(k)**2+rzt(k)**2
30    continue
      knum=k

      jkb=inum(1)

      do 59 i=1,jkb
59    r2t(i)=cvmgt(r2t(i),rneix,r2t(i).le.r2cuti)

      jkb=jkb+1
      do 60 i=jkb,knum
      r2t(i)=cvmgt(r2t(i),rnew,r2t(i).le.r2cut)
60    continue

      return
      end

C  ********************************************
      subroutine energy(ek,ep,etot,vxyz,rxyz)
```

```
C  ******************************************

      parameter(ntm=108,ntm1=ntm-1,ntm3=3*ntm,nn3=ntm*ntm1/2)
      parameter(nn4=3*nn3)

      common /bk1/r2t(nn3),rxt(nn3),ryt(nn3),rzt(nn3),rtxyz(nn4)
      common /blk1/alxyz,c1,c2,smass
      common /blk2/h,h2,h26,hi
      common /blk6/c1d,c2d,inum(ntm1),icord(nn3),r2cut,r2max
      common /blk7/rmin
      common /blk8/vzero
      common /blk9/natom,natom1,natom3
      common /i21/c1i2,c2i2,c3i2,c4i2,c1i2d,c2i2d,c3i2d,c4i2d
      common /i22/vzix,smassi,r2maxi,r2cuti,rnewix
      common /stin/rzero,rmax,elj,sigma

      dimension vxyz(ntm3),rxyz(ntm3)

      ek=0.
      ek1=0.

      do 100 i=1,3
100   ek1=ek1+vxyz(i)**2
      ek1=ek1*smassi/2.

      do 110 i=4,natom3
110   ek=ek+vxyz(i)**2
      ek=ek1+ek*smass/2.

      call ssort(rxyz,r2cut,r2cuti,knum)

      k=0

      ep=0.

      do 130 j=1,inum(1)
      k=k+1
      temp=1./r2t(k)**3
130   ep=ep+temp*(c3i2*temp-c4i2)

      do 140 i=k+1,knum
      temp=1./r2t(i)**3
140   ep=ep+temp*(c1*temp-c2)

      ep=ep-(knum-inum(1))*vzero-(inum(1))*vzix

      etot=ek+ep

      return
      end

C  **********************
```

```
      subroutine stinfo
C  *********************

      parameter(bk=0.831)
      parameter(ntm=108,ntm1=ntm-1,ntm3=3*ntm,nn3=ntm*ntm1/2)
      parameter(nn4=3*nn3)

      common /bk1/r2t(nn3),rxt(nn3),ryt(nn3),rzt(nn3),rtxyz(nn4)
      common /blk1/alxyz,c1,c2,smass
      common /blk2/h,h2,h26,hi
      common /blk4/v(ntm3)
      common /blk5/r(ntm3),r2(ntm3),a(ntm3),a2(ntm3)
      common /blk6/c1d,c2d,inum(ntm1),icord(nn3),r2cut,r2max
      common /blk7/rmin
      common /blk8/vzero
      common /blk9/natom,natom1,natom3
      common /i21/c1i2,c2i2,c3i2,c4i2,c1i2d,c2i2d,c3i2d,c4i2d
      common /i22/vzix,smassi,r2maxi,r2cuti,rnewix
      common /rand/xseed
      common /stin/rzero,rmax,elj,sigma

      real g05ddf,g05caf

C integration time step (ps):
      h=.005

C lj potential zero for Xe-Xe surface (ang):
      rzero=10.

C maximum range for Xe-Xe sort (ang):
      rmax=rzero+1.

C Xe-Xe lj energy parameter (cm-1):
      elj=154.

C Xe-Xe lj distance parameter (ang.):
      sigma=4.10

C Xe mass (amu):
      smass=131.3

C Iodine mass (amu):
      smassi=126.9

C total number of atoms in system:
      natom=108

C length of an edge of the integration cube (ang):
      alxyz=(108.*smass/.6023/1.8)**(1./3.)
C   Adjust the density as follows, for a simulation at 3.0 g/cc
C     alxyz=alxyz*((1.8/3.0)**(1./3.))
```

```
C I-Xe lj potential parameter (cm-1):
      eljix=225.

C I-Xe lj distance parameter (ang):
      sigix=3.94

C cut-off distance squared for I-Xe potential (ang**2):
      r2cuti=(10.)**2

C cut-off distance squared for I-Xe sort (ang**2):
      r2maxi=(11.)**2


      h2=h*h
      h26=h2/6.
      hi=1./h

      r2cut=rzero**2
      r2max=rmax**2

      elj=4.*elj*1.196
      c2=elj*sigma**6
      c1=c2**2/elj
      c1d=12.*c1
      c2d=6.*c2

      eljix=4.*eljix*1.196
      c4i2=eljix*sigix**6
      c3i2=c4i2**2/eljix
      c3i2d=12.*c3i2
      c4i2d=6.*c4i2

      natom1=natom-1
      natom3=3*natom

      vzero=1./rzero**6
      vzero=vzero*(c1*vzero-c2)
      rmin=sigma*(2.**(1./6.))
      rmin=rmin**2

      vzix=1./r2cuti**3
      rnewix=sigix*(2.**(1./6.))
      rnewix=rnewix**2
      vzix=vzix*(c3i2*vzix-c4i2)

      read(2,99) k
 99   format(i5)

      do 156 i=1,natom3,3
      read(2,98) (r(i+j), j=0,2)
      read(2,98) (v(i+j), j=0,2)
 156  continue
```

```
 98   format(3e15.7)

      tmp=300.

      totv = 0.
      sktm=sqrt(bk*tmp/smassi)
      do 260 i=1,3
      v(i)=g05ddf(0.,sktm)
      totv = totv + v(i)*smassi
 260  continue

      sktm=sqrt(bk*tmp/smass)
      do 155 i=4,natom3
      v(i)=g05ddf(0.,sktm)
      totv = totv + v(i)*smass
 155  continue

C   remove center of mass motion.
      scl = totv/natom3
      do 265 i=1,3
 265  v(i) = v(i) - scl/smassi
      do 256 i=4,natom3
 256  v(i) = v(i) - scl/smass

      return

      end

C  ********************************
      subroutine tempc(ek,akt,v)
C  ********************************

      parameter(ntm=108,ntm3=3*ntm)

      common /blk9/natom,natoml,natom3

      dimension v(ntm3)
C
C   Since the energy in the system is approximately equally distributed
C   among kinetic and potential energy, adjust the kinetic energy of
C   the system twice as far as required to give the correct
C   instantaneous temperature.
C
      r=sqrt(akt/(ek*2.-akt)
      if (r.lt.0.9) r=.900

      do 10 i=1,natom3
      v(i)=r*v(i)
 10   continue

      return
      end
```

```
C  ************************
      subroutine fcorr
C  ************************
C
C   subroutine to calculate the total acceleration autocorrelation
C   function.
C
      parameter(ntm=108,ntm1=ntm-1,ntm3=3*ntm,nn3=ntm*ntm1/2)
      parameter(ncr=256)

      common /blk4/v(ntm3)
      common /blk5/r(ntm3),r2jk(ntm3),a(ntm3),a2(ntm3)
      common /fcor/fsave(3,ncr),corf(ncr)
      common /corp/kn,ncor,iflag
      common /atcor/ aa(3),ab(3)

      kn = kn-1
      if(kn.eq.0) kn=ncor
      ncor1 = ncor-kn+1
      kn1=kn-1

      do 30 i=1,3
 30   fsave(i,kn) = a(i)
C
C   'iflag' determines whether the correlation function should be
C   calculated or not.  It is not calculated during initial calls to
C   fcorr() since these are for the purpose of initializing the array
C   'fsave'.
C
      if(iflag.eq.0) goto 110

      do 100 j=1,3

      do 40 i=kn,ncor
 40   corf(i-kn1) = corf(i-kn1)+fsave(j,kn)*fsave(j,i)

      do 50 i=1,kn1
 50   corf(ncor1+i) = corf(ncor1+i) + fsave(j,kn)*fsave(j,i)

 100  continue
 110  continue

      return
      end

C  ******************************
      subroutine wfcorr()
C  ******************************
C
C   subroutine to write out the total acceleration autocorrelation
C   function
```

```
C
      parameter(ncr=256)

      common /fcor/fsave(3,ncr),corf(ncr)
      common /corp/kn,ncor,iflag,nstep

      div=float(nstep*30)
      write(9,99) ncor,nstep
      do 10 i=1,ncor
 10   write(9,98) corf(i)/div
 98   format(2e18.7)
 99   format(2i8)

      return
      end
```

II. PROGRAM LISTING "CALCAR"

```
C     program "CALCAR"
C     David Smith
C     July 24, 1989
C
C   This program takes as input an arbitrary correlation sequence and
C   calculates a set of autoregressive (AR) parameters which gives a
C   best fit to the sequence, following fairly closely the procedures
C   described in chapter 3 of this thesis.  Equation numbers
C   throughout the comments on this program refer to Cha. 3.
C
C   The definitions of each global parameter and variable are given
C   below:
C
C   Parameters:
C
C        NPAR      -    upper limit on AR model order.
C        NPOINT    -    maximum size of correlation sequence arrays.
C        NCOR      -    2*NPOINT, and gives the maximum size of the
C                       arrays associated with the fast Fourier
C                       transform (FFT) of various correlation
C                       sequences.
C
C   Variables:
C
C        np        -    AR model order (p of Eq. (3)).
C        npt       -    number of points in input correlation sequence.
C        npcor     -    number of correlation sequence points
C                       considered in the fit.  Corresponds to the
C                       variable N in equation (2.2).
C        fcor      -    'npt' point array containing the input
C                       correlation sequence.
C
C        ac        -    'np'+1 point array for AR parameters, with the
C                       first value defined as 1., ac(1) = 1.
C        b0        -    AR parameter scaling the normalized gaussian
C                       white-noise driving source.
C        ec        -    'np' point complex array for the mode weights
C                       A_j of Eq. (2.11).
C
C        zc        -    'np' point complex array for the modes,
C                       exp(ω_j), of Eq. (2.11).  The complex valued
C
C                       zc(j) is represented throughout this program in
C                       terms of its magnitude and phase, i.e., zc(j) =
C                       rho(j)*exp(i*omega(j)), where
C        rho       -    'np' point array of the magnitudes of zc.
C        omega     -    'np' point array of the phases of zc.
```

```
C
C       xreal     -    2*'npt' point array containing AR model
C                      correlation sequence calculated via a FFT of
C                      the spectrum in Eq. (2.10a).
C       rtfit     -    'npt' point correlation sequence calculated in
C                      subroutine genfit(), the sequence associated
C                      with a (typically) 1 fs time step.
C
C       epst      -    sum of squared residuals, (ε_r)^2, (Eq. (2.6))
C

      parameter(NPAR=25,NPOINT=256,NCOR=512)

      complex*16 zc(NPAR),ec(NPAR)
      double precision ac(NPAR+1)
      complex multi
      double precision rho(NPAR),omega(NPAR)
      double precision zr,zi,rho1
      double precision xreal(NCOR),fcor(NCOR),b0
      double precision rtfit(NPOINT)
      double precision epst

      common /comp/ zc,ec,ac
      common /par/ rho,omega
      common /nint/ np,npt,npcor
      common /cor/xreal,b0,fcor
      common /fit/ rtfit
      common /coreps/ epst

C   The input file consists of a (typically) 256 point correlation
C   sequence headed by the input parameters np, npcor, and npt.  There
C   is an implicit time step in the input correlation sequence, the
C   spacing between points. This time step, δ, is typically 10 fs.

      read(5,*) np
      read(5,*) npcor
      read(5,*) npt
      do 10 i=1,npt
 10   read(5,*) fcor(i)

      np1 = np+1

C   Subroutine calca() calculates the AR parameters from the input
C   correlation sequence.

      call calca()

C   Subroutine spa() calculates the correlation sequence associated
C   with the AR parameters using FFT techniques.  The resulting
C   sequence is stored in the array 'xreal'.

      call spa(ac,np1)
```

```
C   Subroutine calcz() calculates the modes, exp(ω_j) of Eq. (2.11),
C   and stores results in the complex array 'zc'.

      call calcz()

C   In this loop the values of 'p' and 'omega' are calculated from the
C   array 'zc'.  The relationship is zc(j) = p(j)exp(i*omega(j)).

      do 20 i=1,np
        zr = real(zc(i))
        zi = dimag(zc(i))
        rhol = zr**2 + zi**2
        if(rhol.gt.1.) then
          zr = zr/rhol
          zi = zi/rhol
          rhol = 1./rhol
        endif
        rho(i) = dsqrt(rhol)

C   The next statement gives values of omega from -π/2 to π/2.

        omega(i) = dasin(zi/rho(i))

C   These nested if-then statements convert the value of omega just
C   calculated into the appropriate value between -π and π.

        if(zr.lt.0.) then
          if(zi.lt.0.) then
            omega(i) = -3.1415926 - omega(i)
          else
            omega(i) = 3.1415926 - omega(i)
          endif
        endif
 20   continue

C   Subroutine calcen() calculates the mode weights A_j of Eq. (2.11)
C
C   from the modes 'zc' using Eq. (2.12).  The mode weights are scaled
C   such that they sum to fcor(1).  This ensures that the resulting
C   correlation sequence is normalized to the input sequence.

      call calcen()

C   Subroutine genfit() calculates the correlations sequence for a time
C   step different from the input correlation sequence time step.
C   Results are stored in the array 'rtfit'.

      call genfit()

C   In this loop the 'goodness of fit' for the AR correlation sequences
```

```
C   is calculated in comparison with the input sequence, using a least
C   squares criterion to quantify the fit.

      x2 = 0.d0
      x22 = 0.d0
      do 220 i=1,npcor
      x22 = x22 + (fcor(i)-rtfit(i))**2
 220  x2 = x2 + (fcor(i)-xreal(i))**2

C   Results are output, including fitting evaluations, mode weights and
C   modes, AR parameters, and correlation functions.  This format is
C   required for input into the program "I2MEXP" (Sec. V of this
C   appendix), which simulates vibrational relaxation.

      multi = (0.,1.)
      write(6,*) x2,x22
      write(0,*) x2,x22
      write(6,*) np
      do 30 i=1,np
 30   write(6,101) real(ec(i)),real(-multi*ec(i)),rho(i),omega(i)
      do 40 i=1,np
 40   write(6,*) ac(i+1)
      write(6,*) b0
      write(6,*) npt
      do 50 i=1,256
 50   write(6,*) xreal(i),rtfit(i),fcor(i)
 101  format(4f19.14)

      end

C   ******************************************************************
    subroutine calca()
C   ******************************************************************

C   In this subroutine the AR parameters are calculated from the input
C   correlation sequence.  Calculation proceeds using Eq. (2.28) from
C   Spectral Estimation: an Overdetermined Rational Model equation
C   Approach.  by J. A. Cadzow. Proc. IEEE, 70, p. 907, (1982).  This
C   method is equivalent to that presented in Cha. 3 except that the
C   matrix inversion is more stable. In some cases, double precision
C   is insufficient to effectively invert the matrix in this
C   subroutine.  In this case the inversion has been accomplished on
C   the SDSC Cray X-MP/48 using "cray double precision" (128 bit
C   precision).  A modified version of this program which is suitable
C   for running on the Cray X-MP/48 with the CFT Fortran compiler is
C   contained in Sec. III of this appendix.

C   parameter NMCOR is a maximum allowed value for npcor, giving one
C   dimension of the rectangular matrix rmat.

      parameter (NMCOR=50)
      parameter (NPAR=25,NCOR=512)
```

```
      complex*16 zc(NPAR),ec(NPAR)
      double precision ac(NPAR+1)
      double precision xreal(NCOR),fcor(NCOR),b0
      double precision rmat(NMCOR,NPAR),r2mat(NPAR,NPAR)
      double precision work(2*NPAR)
      double precision eps(NMCOR),epst

      common /comp/ zc,ec,ac
      common /nint/ np,npt,npcor
      common /cor/ xreal,b0,fcor
      common /coreps/ epst

C   rmat = matrix of correlation coefficients.
C
C   r2mat = matrix rmat^T*rmat, where the superscript T indicates the
C   matrix transpose.

C   The matrix 'rmat' is calculated first.

      npl = np+1
      do 10 i=1,npcor
        do 20 j=1,npl
        index = abs(i-j+1)+1
 20     rmat(i,j) = fcor(index)
 10   continue

C   Next the matrix 'r2mat' is calculated from 'rmat'.

      do 30 i=1,npl
        do 40 j=1,npl
        r2mat(i,j) = 0.d0
          do 50 k=1,npcor
            r2mat(i,j) = r2mat(i,j)+rmat(k,i)*rmat(k,j)
 50       continue
 40     continue
 30  continue


C   Inversion of the matrix r2mat is accomplished by the subroutine
C   "kvertd", a double precision version of the napack subroutine
C   "kvert" available from netlib.  Parameters sent to "kvertd" are the
C   array containing the matrix to be inverted (r2mat), the leading
C   (row) dimension of the array (NPAR), the dimension of the matrix
C   (npl), and an array at least twice the size of the matrix dimension
C   (work).  The resulting inverted matrix is stored in the original
C   array (r2mat).

      call kvertd(r2mat,NPAR,npl,work)

C   The AR parameters are calculated from the resulting matrix and
C   normalized such that ac(1) = 1.
```

```
      do 55 i=1,np1
 55   ac(i) = r2mat(i,1)

      do 60 i=np1,1,-1
 60   ac(i) = ac(i)/ac(1)

C   eps is an 'npcor' point array containing the residuals ε_τ from Eq.
C   (2.6).

      do 310 j=1,npcor
      eps(j)=0.d0
        do 320 i=1,np1
 320    eps(j)=eps(j)+rmat(j,i)*ac(i)
 310  continue

C   Sum the squared residuals and write the results.  The value of epst
C   should be quite small, and gives an indication of the effectiveness
C   of the matrix inversion.  It is a better indication of the
C   stability of the inversion than the summed residuals 'x2' or 'x22'
C   of the main section of this program.

      epst = 0.d0
      do 330 i=1,npcor
 330  epst = epst + eps(i)**2
      write(6,*) epst

      return
      end

C   ******************************************************************
      subroutine calspa(ac,np)
C   ******************************************************************

C   Subroutine to calculate the correlation sequence associated with a
C   set of AR parameters. The spectrum is first calculated as in Eq.
C   (2.10), and then Fourier transformed to give the correlation
C   sequence. The resulting sequence is normalized to the input
C   sequence and stored in 'xreal'.   The parameter b_0 is calculated
C
C   from the normalization.  This subroutine is generalized to run on
C   the cray since it does not contain any complex double precision
C   variables.  (The cray does not support 128 bit complex).

C   omega     -    frequencies in the correlation sequence of the
C                  spectrum, calculated from the number of points in
C                  the input correlation sequence 'fcor'.  Values range
C                  from 0 to 2π.
C   spectr    -    real part of spectrum.
C   specti    -    imaginnary part of spectrum.

      parameter (NPAR=25,NCOR=512)
```

```
      double precision ac(NPAR+1)
      double precision xreal(NCOR),fcor(NCOR),b0
      double precision omega(NCOR),spectr,specti

      common /cor/xreal,b0,fcor

C   The frequencies are calculated first, followed by a calculation of
C   the denominator in Eq. (2.8).  The denominator is then multiplied
C   by its complex conjugate to give the spectrum.  ncr = 2*npt

      ncr = 512
      ncr2 = ncr/2
      do 40 i=1,ncr2
      omega(i) = (dble(i-1)/dble(ncr))*2.*3.14159265
      specti = 0.
      spectr = 0.
        do 60 j=1,np+1
        spectr=spectr + ac(j)*dcos(omega(i)*dble(j-1))
 60     specti=specti + ac(j)*dsin(omega(i)*dble(j-1))
      xreal(i)=1.d0/(spectr**2+specti**2)
 40   continue

C   Calculation of the Fourier transform proceeds using an double
C   precision version of the FFT algorithm from The Fast Fourier
C   Transform, by E. Oran Brigham, (Prentice-Hall, New Jersey, 1974).
C   The global array 'xreal' is used in the FFT subroutine.  The
C   resulting time correlation sequence is returned in 'xreal'.

      call fft(ncr2)

C   The variable b0 is now calculated and the time correlation
C   sequence is normalized.

      b0 = fcor(1)/xreal(1)
      do 70 i=1,ncr2
 70   xreal(i) = xreal(i)*b0
      b0 = sqrt(b0)

      end

C   ******************************************************************
      subroutine calcz()
C   ******************************************************************

C   In this subroutine the modes, zc, are calculated from the AR
C   parameters via the relationship of Eq. (4.2).

      parameter (NPAR=25,NCOR=512)

C   p    =    complex double precision array of AR parameters, in
C             descending order.
C   work =    work array.  Must be at least 6*NPAR elements long.
```

```
C   zd   =    array of zeros.

      complex*16 zc(NPAR),ec(NPAR)
      double precision ac(NPAR+1)
      complex*16 work(10*NPAR),zd(NPAR),p(NPAR+1)

      common /comp/ zc,ec,ac
      common /nint/ np,npt,npcor

C   put AR parameters in array 'p' in descending order.

      np1 = np+1
      do 10 i=1,np1
 10   p(np1+1-i) = cmplx(ac(i))

C   czero is a complex double precision root-finding subroutine from
C   napack, available through netlib.  Complex roots of the polynomial
C   'p' are stored in the array 'zd'.

      call czero(zd,p,np,work)

C   Finally, the results are stored in array 'zc'.

      do 20 i=1,np
 20   zc(i) = zd(i)

      return
      end

C   ******************************************************************
    subroutine calcen()
C   ******************************************************************

C   Subroutine calcen calculates the mode weights A(j) of Eq. (2.11)
C   from the frequencies via Eq. (2.12).  The sum of the mode weights
C   are normalized to the r = 0 value of the input correlation
C   sequence.

      parameter(NPAR=25,NPOINT=256,NCOR=512)

C   artot     =    sum of resulting mode weights.
C   om        =    array of new frequencies.

      double precision artot
      complex*16 om(NPAR)
      double precision rho(NPAR),omega(NPAR)
      complex*16 zc(NPAR),ec(NPAR)
      double precision ac(NPAR+1)
      double precision xreal(NCOR),fcor(NCOR),b0

      common /comp/ zc,ec,ac
      common /cor/xreal,b0,fcor
```

```
      common /par/ rho,omega
      common /nint/ np,npt,npcor

C   calculate the frequencies from the radii and phase.

      do 10 i=1,np
 10   om(i) = dlog(rho(i)) + (0.d0,1.d0)*omega(i)

C   Calculation of the resulting mode weights from Eq. (2.12).
C   The variable artot contains a sum of the mode weights, and is used
C   below to scale the mode weights such that the resulting sequence
C   equals the input sequence at τ = 0.

      artot = 0.d0
      do 20 j=1,np
      ec(j) = 1.d0/(1.d0-cdexp(2.d0*om(j)))
        do 30 i=1,np
        if(i.ne.j) then
         ec(j)=ec(j)/((1.d0-cdexp(om(i)-om(j)))
     $    *(1.d0-cdexp(om(i)+om(j))))
        endif
 30     continue
      artot = artot + ec(j)
 20   continue

C   scaling of the mode weights.

      do 40 i=1,np
 40   ec(i) = ec(i)*fcor(1)/artot

      return
      end

C   ******************************************************************
      subroutine genfit()
C   ******************************************************************

C   Subroutine genfit calculates the correlations sequence for a time
C   step different from the input correlation sequence time step.
C   Results are stored in the array rtfit.  Calculation proceeds by
C   first scaling the known frequencies by the ratio h/δ, as described
C   in Sec. IV(A) of Cha. 3, followed by determination of the mode
C   weights corresponding to the new frequencies via Eq. (2.12).
C   Finally, the mode weights are scaled such that the resulting
C   correlation sequence is equal to the input sequence at δ = 0.

      parameter(NPAR=25,NPOINT=256,NCOR=512)

C   artot     =    sum of resulting mode weights.
C   ar        =    array containing resulting mode weights.
C   rtt       =    complex working array in calculation of resulting
C                  correlation sequence.
```

```
C   c          =    working variable in calculation of resulting
C                   correlation sequence.
C   om         =    array of new frequencies.
C   rtfit      =    array containing new correlation sequence.

      double precision artot
      complex*16 ar(NPAR)
      complex*16 rtt(NPOINT),c,om(NPAR)
      double precision rtfit(NPOINT)
      double precision rho(NPAR),omega(NPAR)
      complex*16 zc(NPAR),ec(NPAR)
      double precision ac(NPAR+1)
      double precision xreal(NCOR),fcor(NCOR),b0

      common /comp/ zc,ec,ac
      common /cor/xreal,b0,fcor
      common /par/ rho,omega
      common /nint/ np,npt,npcor
      common /fit/ rtfit

C   Calculation of the new frequencies.  The value scaling om(i) in
C   line 10 is h/δ.

      do 10 i=1,np
      om(i) = dlog(rho(i)) + (0.d0,1.d0)*omega(i)
 10   om(i) = om(i)*.1d0

C   Calculation of the resulting mode weights from Eq. (2.12).
C   The variable artot contains a sum of the mode weights, and is used
C   below to scale the mode weights such that the resulting sequence
C   equals the input sequence at τ = 0.

      artot = 0.d0
      do 20 j=1,np
      ar(j) = 1.d0/(1.d0-cdexp(2.d0*om(j)))
        do 30 i=1,np
        if(i.ne.j) then
         ar(j)=ar(j)/((1.d0-cdexp(om(i)-om(j)))
     $    *(1.d0-cdexp(om(i)+om(j))))
        endif
 30     continue
      artot = artot + ar(j)
 20   continue

C   The mode weights are scaled.

      do 40 i=1,np
 40   ar(i) = ar(i)*fcor(1)/artot

C   The correlation sequence array 'rtt' is initialized.  The array
C   'rtt' must be complex since ar(i) and zc(i) are complex, yet the
```

```
C   final values for 'rtt' are real since the modes and mode weights
C   occur in complex conjugate pairs.

      do 50 i=1,npt
 50   rtt(i) = 0.d0

C   calculation of the resulting correlation sequence.

      do 60 i=1,np
        c = (1.,0.)
        do 70 j=1,npt
          rtt(j) = rtt(j) + c*ar(i)
          c = c*zc(i)
 70     continue
 60   continue

C   The resulting sequence is stored in the array 'rtfit'

      do 80 i=1,npt
 80   rtfit(i) = dreal(rtt(i))

      return
      end
```

III. SAMPLE OUTPUT "CALCAR"

```
8.050e-02   1.879e-03
  4
  18.89368247985840 -17.07722854614258   0.93179094804143
0.06211889371038
  -1.22960925102234   0.46247446537018   0.86883149110107
-0.21965109661084
  -1.22960925102234  -0.46247446537018   0.86883149110107
0.21965109661084
  18.89368247985840  17.07722854614258   0.93179094804142
-0.06211889371037
  -3.5559004835248
   4.7774797266044
  -2.8764953413998
  0.65540248553589
   1.5964376931920d-02
  256
   35.328850000000   35.328850519632   35.328850000000
   35.207396431238   35.208253201119   35.209130000000
   34.846543922422   34.849872736596   34.853400000000
   34.256485895648   34.263667773099   34.271170000000
   33.453304654852   33.465398849831   33.477870000000
   32.457867507146   32.475576248632   32.493470000000
   31.294585113532   31.318246615653   31.341840000000
   29.990121193525   30.019738175595   30.048910000000
   28.572180228029   28.607457349239   28.642060000000
   27.068393604096   27.108803662763   27.148430000000
   25.505413897670   25.550246274174   25.594460000000
   23.908157300328   23.956585057346   24.004650000000
   22.299273599676   22.350402408106   22.401530000000
   20.698771236359   20.751698880768   20.804820000000
   19.123849118329   19.177696291738   19.231750000000
   17.588835559484   17.642785724626   17.696490000000
   16.105282861561   16.158594507941   16.210830000000
   14.682102541277   14.734145218816   14.783780000000
   13.325839586521   13.376080583170   13.422310000000
   12.040913370704   12.088930299889   12.131100000000
   10.829929712952   10.875398861267   10.913290000000
   9.6939570484001   9.7366569796048   9.7702610000000
   8.6328324868861   8.6726229333376   8.7023720000000
   7.6454057403187   7.6822237581633   7.7086660000000
   6.7297908117415   6.7636295403537   6.7876270000000
   5.8835456997340   5.9144569943052   5.9368070000000
   5.1038726976648   5.1319409526446   5.1535490000000
   4.3877339214764   4.4130743387063   4.4345700000000
   3.7319759650737   3.7547186377164   3.7766060000000
   3.1333962469157   3.1536878709359   3.1759940000000
   2.5888211800689   2.6068096606791   2.6292930000000
   2.0951311272154   2.1109672182177   2.1327940000000
   1.6493022593872   1.6631260598643   1.6831220000000
```

1.2483902873934	1.2603490267262	1.2767240000000
0.88957637358703	0.89980281376071	0.91046610000000
0.57013738500482	0.57875875959172	0.58108390000000
0.28745755332199	0.29459015705474	0.28574780000000
3.9011282688611d-02	4.4767853243341d-02	2.1529680000000d-02
-0.17762776739198	-0.17314455472748	-0.21405170000000
-0.36480300247238	-0.36149503627267	-0.42358820000000
-0.52476799882986	-0.52254641819249	-0.60923600000000
-0.65970258156859	-0.65847840001492	-0.77325990000000
-0.77169611136349	-0.77138826618908	-0.91750610000000
-0.86276060973663	-0.86329042736933	-1.0439520000000
-0.93482033020530	-0.93611502706527	-1.1541790000000
-0.98971873028983	-0.99170590629453	-1.2499050000000
-1.0292056053697	-1.0318182365244	-1.3324650000000
-1.0549433206257	-1.0581161191552	-1.4033500000000
-1.0684963886819	-1.0721704166359	-1.4636740000000
-1.0713427834204	-1.0754570336802	-1.5147180000000

IV. PROGRAM LISTING "CALCARC"

```
C     program "CALCARC"
C     David Smith
C     Aug. 2, 1989
C
C   This is a cray-compatible version of a portion of the program
C   "CALCAR" of Sec. II of this appendix. The matrix inversion
C   involved in solving for the AR parameters in program "CALCAR"
C   sometimes requires greater than 64-bit precision to be stable. In
C   these cases, inversion has been accomplished using cray double
C   precision (128-bit). This listing contains only the main section
C   of a program which calculates the set of AR parameters which give a
C   best fit of the input correlation sequence. All subroutines called
C   from this program are described in Sec. II, and can be used as
C   presented there. This program calculates only the AR parameters
C   and a measure of the goodness of fit, x2, as in "CALCAR". "CALCAR"
C   must then be modified (a simple task) to read in the AR parameters
C   from the output of this program, giving in turn the complete
C   output file required in the program "I2MEXP" of Sec. V. For
C   details on compiling and running programs on the San Diego
C   Supercomputer Center Cray X-MP/48 see the SDSC users guide or other
C   SDSC documentation.

      program calcarc

C   All parameters and variables and subroutine calls in this program
C   are identical to those used in "CALCAR", and the reader is referred
C   to Sec. II of this appendix for more details.

      parameter(NPAR=55,NPOINT=256,NCOR=512)

      double precision ac(NPAR+1)
      double precision p(NPAR),omega(NPAR)
      double precision energy(NPAR),rt(NPOINT*2)
      double precision zr,zi,rho
      double precision xreal(NCOR),fcor(NCOR),b0
      double precision rtfit(NPOINT)

      common /comp/ zc,ec,ac
      common /par/ p,omega,energy,rt
      common /nint/ np,npt,npcor
      common /cor/ xreal,b0,fcor,epst
      common /fit/ rtfit

C   The following command is specific to the CFT compiler on the Cray
C   X-MP/48. It must be the first executable line in the program, as it
C   is responsible for defining all of the I/O unit numbers and for
C   opening necessary files. The file names, 'out6' and 'fc18r' can be
C   redefined upon execution of the program. See the Cray documentation
```

```
C   for more details.  'fc18r' is the input file, and has the same form
C   as the input files to "CALCAR".  'out6' is the output file and
C   contains the resulting AR parameters among other quantities.

      call link("unit6=(out6,create,text),unit5=(fc18r,open,text)
     $//")

      read(5,*) np
      read(5,*) npcor
      read(5,*) npt
      do 10 i=1,npt
 10   read(5,*) fcor(i)

      call calca()

      call spa(ac,np)

      x2 = 0.
      do 122 i=1,np1
 122  x2 = x2 + (fcor(i)-xreal(i))**2

      write(6,*) x2,epst

      write(6,*) np
      do 151 i=1,np1
 151  write(6,103) ac(i)

      do 500 i=1,51
 500  write(6,103) xreal(i),fcor(i)
 103  format(2e19.9)

C   'exit' is called at the end of the program, and is responsible for
C   closing all of the I/O connections.

      call exit

      end
```

V. PROGRAM LISTING "I2MEXP"

```
C     program "I2MEXP"
C     David Smith
C     July 31, 1989
C
C   This program calculates the vibrational relaxation of an iodine
C   molecule in a generalized Langevin model solvent.  The calculation
C   proceeds as in the model described in Cha. 4 of this thesis.
C   Equation numbers throughout the comments refer to Cha. 4 except
C   where noted.  Input into this program is from the output file from
C   the program "CALCAR" of Sec. II, modified slightly as described
C   below.  Several aspects of this program are derived directly from
C   the program "I2XENON" by J. Keenan Brown, a complete listing of
C   which is available in his PhD thesis, including lengthy comments.
C   Specific parts of this program which are derived from "I2XENON"
C   will be noted below.
C
C   The basic units associated with the integrator of this program are
C
C        length  -  angstroms
C        time    -  picoseconds
C        mass    -  amu's.
C
C   In some places, mostly at output, these are converted to more
C   convenient units.  In particular, output energies are in units of
C   wavenumbers.  All cases where units vary from those above are
C   clearly marked in the comments.  The definitions of each global
C   parameter and variable are given below:
C
C   Parameters:
C
C        natm3     -    maximum number of degrees of freedom in the
C                       system, 6.
C        nrkr      -    number of points in arrays associated with
C                       calculation of intramolecular forces and
C                       potential energies from the iodine RKR
C                       potential.
C        npr       -    maximum number of AR parameters.
C
C   Variables:
C
C        natom3    -    number of degrees of freedom in the system, 6.
C                       This is distinguished from the parameter ntm3
C                       because this program can be used to derive a
C                       program for motion of a single atom in the GLE
C                       fluid, with 'natom3' = 3 and the intramolecular
C                       equations of motion eliminated.
C        npar      -    number of AR parameters.
C        npar1     -    'npar' + 1.
```

```
C        nseed     -    random number generator seed, from the input
C                       file.
C
C        h         -    time step.
C        h2        -    h**2.
C        h26       -    h2/6.
C        hi        -    1./h
C        ek        -    kinetic energy of vibrational coordinate.
C        ep        -    potential energy of vibrational coordinate.
C        ekt       -    total kinetic energy of the diatomic.
C        rxyz      -    array of atom positions.
C        vxyz      -    array of atom velocities.
C        axyz      -    array of atom accelerations.
C        rxyz2     -    array of atom positions in the previous time
C                       step.
C        v2        -    array of atom velocities in the previous time
C                       step.
C        axyz2     -    array of atom accelerations in the previous
C                       time step.
C        smassi    -    iodine mass, 126.9 amu's.
C        smass     -    1./smassi
C        rad2      -    square of the iodine molecule bond length.
C        nover     -    flag to indicate whether the iodine bond length
C                       is over 8 amu's.  The iodine molecule
C                       occasionally dissociates instead of relaxing
C                       down the well.  The 'nover' flag is activated
C                       if the radius reaches a large enough value to
C                       indicate dissociation, and the trajectory is
C                       then restarted.
C        apar      -    array of AR parameters.  apar(1) = a1, etc., as
C                       in Cha. 4.
C        b0        -    b0 from Cha. 4.
C
C        eeps      -    array of current and 'npar' previous values of
C                       the random force.
C
C   the friction is calculated as described in section IV(B) of Cha. 3,
C   except that a second order integrator is used.  The following are
C   arrays associated with that calculation.  Equation numbers refer to
C   Cha. 3.  For a multiexponential force it can be seen from Eq. (4.7)
C   that the total force at time r is equal to the sum of p complex
C   forces associated with each mode of the AR model.  Each of these p
C   forces consist of two terms, an exponential factor times the force
C   at the previous time step, minus the present velocity scaled by a
C   factor of the time step and the mode weight.  (See Eq. (4.6))
C
C        fric          -    array of current frictional forces for
C                           each coordinate.
C        (frcr,frci)   -    matrices of the real and imaginary parts
C                           of the pth force on each coordinate.
C
```

```
C   The following two quantities, the exponential factors and velocity
C   scaling factors, are identical for each coordinate.
C
C        (magr,magi)    -    arrays of the real and imaginary parts of
C                            the velocity scaling factor of the pth
C                            force, hC in Eq. (4.6).
C        (expr,expi)    -    arrays of the real and imaginary parts of
C                            the exponential factor of the pth force
C
C   The following arrays are interpolation tables for the iodine RKR
C   force and potential energy.  Details of the technique used in this
C   calculation are described in detain in J. K. Brown, PhD thesis.
C
C
C        (ftab,ftabi)   -    force arrays.
C        (vtab,vtabi)   -    potential energy arrays.
C
C   The force is found in the following way.
C
C        n = int(4750./rad2)
C        force = ftab(n) + sqrt(rad2)*ftabi(n)
C
C   with rad2 being the square of the current iodine bond length.  The
C   potential energy is found in a similar manner.

      implicit double precision (a-h,o-z)

      parameter(natm3=6)
      parameter(nrkr=1000)
      parameter(npr=15)

      dimension tum(151),tum2(151),tum3(151),tum4(151)
      dimension sum(151),sum2(151),sum3(151),sum4(151)
      double precision eps(natm3)
      double precision f,v,fl,vl,vt,r2
      double precision hadj
      double precision rho,omega
      double precision bufr(npr),bufi(npr)
      double precision vxyz
      double precision rxyz,rxyz2,axyz,axyz2
      double precision ftab,ftabi,vtab,vtabi
      double precision eeps
      double precision ek,ep,ekt
      double precision h,h2,h26,hi
      double precision rad2
      double precision smassi,smass
      double precision fric(natm3),v2(natm3)
      double precision magr,magi,expr,expi
      double precision frcr,frci
      double precision apar(npr),b0
      complex*16 pc(npr+1),zc(npr)
      complex*16 om(npr),ar(npr)
```

```
      common /gstart/nseed
      common /eng/ek,ep,ekt
      common /blk2/h,h2,h26,hi
      common /blk4/vxyz(natm3)
      common /blk5/rxyz(natm3),rxyz2(natm3),axyz(natm3),axyz2(natm3)
      common /blk6/ftab(nrkr),ftabi(nrkr),vtab(nrkr),vtabi(nrkr)
      common /blk7/rad2,nover
      common /blk9/natom3,npar,npar1
      common /blk10/frcr(natm3,npr),frci(natm3,npr)
      common /lang/smassi,smass
      common /kexp2/eeps(natm3,npr)
      common /forc/fric,v2
      common /poly/magr(npr),magi(npr),expr(npr),expi(npr)
      common /rnd/apar,b0

C   Input begins with the header on the ouput file from "CALCAR":
C
C   nseed     -    random seed, initializes random number generator.
C   nstep     -    number of points in trajectory at which energy
C                  information is saved.  Typically 150.
C   nrun      -    number of trajectories.
C   nstep2    -    number of calls to 'integ' between saving energy
C                  information.
C   h         -    time step.  h*nstep*2 - picoseconds between points
C                  in output file.
C
C   The rest of the input file is directly from the output of "CALCAR"

      read(5,*) nseed,nstep
      read(5,*) nrun,nstep2
      read(5,*) h

C   iflag     -    flag to indicate whether to use the correlation
C                  sequence of Eq. (4.10) of of Eq. (4.12) for the
C                  memory function, depending on whether x2 or x22 of
C                  "CALCAR" is larger.  Set iflag to '0' if x2 < x22
C                  and to '1' if x22 < x2.

      read(5,*) x2,x22
      if(x2.lt.x22) then
        iflag - 0
      else
        iflag - 1
      endif
C
C   npar      -    number of AR parameters.
      read(5,*) npar
      npar1 - npar+1

C   acon is a scale factor for the memory function.  Typically the AR
C   fits in "CALCAR" are performed on the random acceleration
```

```
C   autocorrelation function. The memory function equals the random FAF
C   divided by k_bTm. acon therefore scales the input by m**2, to give
C
C   the FAF, and divides by k_bT to give the memory function times the
C
C   mass, mγ(t) of the GLE.  (see Eq. (1))  The time step factor
C   corresponds to the factor 'h' in Eq. (4.6) of Cha. 3.

      acon = h*126.9d0**2/(.831d0*300.d0)

C   'hadj' is a time step adjustment variable.  Used in making this
C   program general for any time step.  hadj = h/δ, where δ is the time
C   between points in the original single atom FAF.

      hadj = h*100.d0

C   This loop reads in the mode weights (magr,magi), radii (rho), and
C   phases (omega) for the AR model of the memory function.  Values of
C   the mode weights are modified slightly to give a second order
C   integrator in the calculation of the friction integral, and also
C   multiplied by the scaling factor 'acon'. (bufr,bufi) and
C   (amgr,amgi) are temporary variables associated with these
C   modifications.  The frequencies for the simulation time step, 'om',
C   are calculated, as well as the exponential factors (expr,expi) and
C   the modes associated with the simulation time step, 'zc'.

      do 13 i=1,npar
        read(5,*) magr(i),magi(i),rho,omega
        om(i) = dlog(rho) + (0.d0,1.d0)*omega
        om(i) = om(i)*hadj
        bufr(i) = dexp(dlog(rho)*hadj*.5d0)*dcos(omega*hadj*.5d0)
        bufi(i) = dexp(dlog(rho)*hadj*.5d0)*dsin(omega*hadj*.5d0)
        amgr = magr(i)
        amgi = magi(i)
        magr(i) = acon*(amgr*bufr(i)-amgi*bufi(i))
        magi(i) = acon*(amgi*bufr(i)+amgr*bufi(i))
        expr(i) = dexp(dlog(rho)*hadj)*dcos(omega*hadj)
        expi(i) = dexp(dlog(rho)*hadj)*dsin(omega*hadj)
        zc(i) = dexp(dlog(rho)*hadj)*cdexp((0.d0,1.d0)*omega*hadj)
 13   continue

C   The rest of the parameters are now read in, including 'b_0', and
C   'fcorl', the value of the original correlation sequence input into
C   "CALCAR".

      do 14 i=1,npar
 14   read(5,*) junk
      read(5,*) b0
      read(5,*) i
      read(5,*) fcorl
```

```
C   Next the new mode weights are calculated from the frequencies 'om'
C   as in Eq. (4.9), and scaled such that the resulting correlation C
sequence equals 'fcorl' at τ = 0.  b0 is scaled in the process.

      artot = 0.d0
      do 920 j=1,npar
      ar(j) = 1./(1.d0-cdexp(2.d0*om(j)))
        do 925 i=1,npar
        if(j.ne.i) then
          ar(j)=ar(j)/((1.d0-cdexp(om(i)-om(j)))
     $    *(1.d0-cdexp(om(i)+om(j))))
        endif
 925  continue
      artot = artot + ar(j)
 920  continue
      do 930 j=1,npar
 930  ar(j) = ar(j)*fcorl/artotl
      b0 = b0*dsqrt(fcorl/artotl)

C   If x22 < x2 (from "CALCAR"), use ar(i) instead of magr(i) for the
C   mode weights in the calculation of the friction.

      if(iflag.eq.1) then
      do 940 i=1,npar
      magr(i) = acon*(dreal(ar(i))*bufr(i)-dimag(ar(i))*bufi(i))
      magi(i) = acon*(dimag(ar(i))*bufr(i)+dreal(ar(i))*bufi(i))
 940  continue
      endif

C   The new AR parameters are now calculated from the modes 'zc', using
C   the relationship of Eq. (4.11).  'pc' is a temporary complex
C   variable used in the calculation of the AR parameters, which are
C   eventually stored in the array 'apar'.

      pc(1) = (1.,0.)
      do 17 i=2,npar+1
 17   pc(i) = (0.,0.)

      do 15 i=1,npar
        do 16 k=i+1,2,-1
 16     pc(k) = pc(k)-zc(i)*pc(k-1)
 15   continue

      do 18 i=1,npar
 18   apar(i) = dreal(pc(i+1))

C   These variables are used in the integrator and are calculated once
C   here to minimize computations in 'integ'.

      h2=h*h
      h26=h2/6.d0
      hi=1.d0/h
```

```
C   tmp (temperature) is not used in the calculation, but it is printed
C   out for completeness.  Temperature in these simulations is implicit
C   in the FAF's used for input into "CALCAR".

      tmp=300.d0

C   Write the header for the output file

      write(6,*) h,nstep2,tmp,nstep,nrun

C   natom3 , the total degrees of freedom in the system, is only
C   modified if the program is drastically modified, for example to
C   calculate motion of a single atom.

    natom3=6

C   The input file 'rkrf' is the same as used by J. K. Brown.  See his
C   PhD thesis for more details of the derivation.  The force table is
C   input directly from the file 'rkrf', and the potential table is
C   then calculated by integrating the force.  the variables 'f', 'v',
C   'fl', 'vl', 'vt', and 'r2' are local variables used in this
C   calculation.

      open(1,status='old',file='/usr/users/dave/rkrf')
      do 40 i=1,1000
 40   read(1,*) j,vtab(i),ftab(i),ftabi(i)
      close(1)

      vt = 0.d0
      vl = 0.d0
      vtabi(1) = 0.d0
      fl = 0.d0
      do 820 i=2,1000
      r2 = vtab(i)**2
      n = int(4750.d0/r2)
      f = ftab(n) + vtab(i)*ftabi(n)
      v = v + .5d0*(vtab(i-1)-vtab(i))*(f+fl)
      vtabi(i) = (v-vl)/(vtab(i)-vtab(i-1))
      vtab(i-1) = vt
      vt = v-vtab(i)*vtabi(i)
      fl = f
      vl = v
 820  continue
      vtab(1000)=vt

C   Various quantities are initialized prior to starting the
C   trajectories.  The variables 'sum', 'sum2', 'sum3', and 'sum4' are
C   stored quantities which will eventually comprise the output.

      smassi=126.9d0
      smass=1/smassi
```

```
      do 155 i=1,151
      sum3(i)=0.
      sum4(i)=0.
      sum2(i)=0.
 155  sum(i)=0.

C   nrun trajectories are run.

      do 145 l=1,nrun

 9    continue

C   Various quantities are initialized for each trajectory, including
C   positions, velocities, accelerations, frictional forces, and random
C   forces.

      nover=0
      do 10 i=1,natom3
      fric(i)=0.d0
      do 12 j=1,npar
      frci(i,j)=0.d0
 12   frcr(i,j)=0.d0
      axyz(i)=0.d0
      axyz2(i)=0.d0
 11   vxyz(i)=0.d0
      rxyz2(i)=0.d0
 10   rxyz(i)=0.d0

C   Positions and velocities are initially chosen such that the iodine
C   is in the ground state with ~11,500 wavenumbers of energy and
C   random phase.  This is accomplished by reading in a table of
C   230 positions and velocities corresponding to one oscillation of an
C   iodine molecule with the desired energy.  The particular position
C   and velocity are then chosen at random from this table.

      open(2,status='old',file='/usr/users/dave/startdat')
      x1=urand(nseed)
      kn2=int(x1*230)+1

      do 335 l2=1,kn2
      read(2,*)rx1,vx1
 335  continue
      vxyz(1)=vx1
      vxyz(4)=-vx1
      rxyz(1)=rx1
      rxyz(4)=-rx1

      close(2)

C   In this section the random numbers are initialized.  Random numbers
C   are generated as in equation (4.1) for 1000 steps.
```

```
      do 215 i2=1,natom3
      do 215 i3=1,npar1
215   eeps(i2,i3)=0.d0

      ntot1 = 1000
      do 517 i1=1,ntot1/10
        do 535 ii = 1,10
        call gauss(eps,natom3)
        do 518 j=1,natom3
          do 527 j2=npar1,2,-1
527       eeps(j,j2) = eeps(j,j2-1)
        eeps(j,1) = eps(j)*b0
          do 525 j2=2,npar1
525       eeps(j,1) = eeps(j,1) - apar(j2-1)*eeps(j,j2)
518     continue
535     continue
517   continue

C   The trajectory is no calculated for nstep*nstep2*2 time steps.
C   (Each call to 'integ' corresponds to two time steps)  After each
C   nstep2 calls to 'integ', the subroutine 'energy' is called,
C   returning the kinetic and potential energies of the vibratinal
C   coordinate and the total kinetic energy of the iodine molecule.
C   The total vibrational energy, total vibrational energy squared,
C   total kinetic energy and vibrational kinetic energy are stored in
C   the variables tum, tum2, tum3, and tum4 respectively.  All of these
C   energies are converted to units of wavenumbers before being stored.
C   If the nover flag has been tripped, the trajectory is restarted.
C   Progress of the trajectory including the total energy is printed
C   and flushed to the output file allowing for easy tracing of the
C   program's progress.

      do 100 i=1,nstep

      call energy(rxyz,vxyz)
      et=(ek+ep)/1.1961d0 +12568.4d0
      tum(i)=et
      tum2(i)=et**2
      tum3(i)=ekt/1.1961d0
      tum4(i)=ek/1.1961d0
      if(nover.eq.1) goto 9

      write(6,113) 1,i,et
113   format(2i5,f15.6)
      call flush(6)

        do 120 j=1,nstep2
120     call integ()

100   continue
```

```
C   The 'energy' subroutine is called one final time to give the final
C   state of the iodine molecule.

      call energy(rxyz,vxyz)
      et=(ek+ep)/1.1961d0 +12568.4d0
      tum(nstep+1)=et
      tum2(nstep+1)=et**2
      tum3(nstep+1)=ekt/1.1961d0
      tum4(nstep+1)=ek/1.1961d0

C   Results are tabulated in the variables sum, sum2, sum3, and sum4,
C   corresponding to the same quantities as the variables tum, tum2,
C   etc.  Storing these values in the 'sum' variables after the
C   trajectory is complete prevents the inclusion of any data from
C   trajectories which dissociated.

      do 141 k=1,nstep+1
      sum(k)=tum(k)+sum(k)
      sum3(k)=sum3(k)+tum3(k)
      sum4(k)=sum4(k)+tum4(k)
 141  sum2(k)=sum2(k)+tum2(k)
 145  continue
      close (1)

      do 165 i=1,nstep+1

      sum(i)=sum(i)/nrun
      sum3(i)=sum3(i)/nrun
      sum4(i)=sum4(i)/nrun

C   Finally, the variable 'sum2' is turned into a variance instead of
C   just the average squared total energy.  This gives a measure of the
C   width of the energy distributions.  Results are printed in the
C   output file.

      sum2(i)=sum2(i)/nrun - sum(i)**2
      if(sum2(i).gt.0) sum2(i)=dsqrt(sum2(i))
      write(6,96)nstep2*2*h*(i-1),sum(i),sum2(i),sum3(i),sum4(i)
 96   format(5f15.4)

 165  continue

      end

C  ******************************************************************
      subroutine integ()
C  ******************************************************************

C   This subroutine is the integrator for the simulation.  It is a
C   second order integrator described by Beeman (reference in Cha. 4).
C   It is a slightly modified version of an integrator written by John
C   Tully at AT&T Bell Laboratories.  It is identical to that used in
```

```
C   the molecular dynamics simulations of J. K. Brown, and a complete
C   description of it can be found in Cha. 2 of his PhD thesis.

      implicit double precision (a-h,o-z)

      parameter (natm=2,natml=natm-1,natm3=3*natm)

      double precision vxyz
      double precision rxyz,rxyz2,axyz,axyz2
      double precision h,h2,h26,hi
      double precision rad2

      common /blk2/h,h2,h26,hi
      common /blk4/vxyz(natm3)
      common /blk5/rxyz(natm3),rxyz2(natm3),axyz(natm3),axyz2(natm3)
      common /blk9/natom3,npar,npar1
      common /blk7/rad2,nover

C   Two integration steps are taken for each call to this subroutine.

      do 100 i=1,natom3
      rxyz2(i)=rxyz(i)+h*vxyz(i)+h26*(4.d0*axyz(i)-axyz2(i))
100   axyz2(i)=0.d0

      call accel(axyz2,rxyz2)

      do 110 i=1,natom3
      vxyz(i)=(rxyz2(i)-rxyz(i)+h26*(2.d0*axyz2(i)+axyz(i)))*hi
      rxyz(i)=rxyz2(i)+h*vxyz(i)+h26*(4.d0*axyz2(i)-axyz(i))
110   axyz(i)=0.d0

      call accel(axyz,rxyz)

      do 120 i=1,natom3
120   vxyz(i)=(rxyz(i)-rxyz2(i)+h26*(2.d0*axyz(i)+axyz2(i)))*hi

      return
      end

C  ****************************************************************
      subroutine accel(axyz,rxyz)
C  ****************************************************************

C   This subroutine includes calculation of the accelerations on each
C   of the six coordinates due to the I-I interaction, the friction,
C   and the random forces.

      implicit double precision (a-h,o-z)

      parameter(natm3=6)
      parameter(nrkr=1000)
      parameter(npr=15)
```

```
      double precision vxyz
      double precision ftab,ftabi,vtab,vtabi
      double precision eeps
      double precision eps(natm3)
      double precision rad2
      double precision smassi,smass
      double precision axyz(natm3)
      double precision rxyz(natm3)
      double precision r(3)
      double precision for(natm3)
      double precision fric(natm3),v2(natm3)
      double precision magr,magi,expr,expi
      double precision apar(npr),b0

      common /lang/smassi,smass
      common /blk4/vxyz(natm3),gam
      common /blk6/ftab(nrkr),ftabi(nrkr),vtab(nrkr),vtabi(nrkr)
      common /blk7/rad2,nover
      common /blk9/natom3,npar,npar1
      common /blk10/frcr(natm3,npr),frci(natm3,npr)
      common /kexp2/eeps(natm3,npr)
      common /forc/fric,v2
      common /poly/magr(npr),magi(npr),expr(npr),expi(npr)
      common /rnd/apar,b0

C     This section calculates the force on each coordinate due to the I-I
C     interaction.  It is first necessary to calculate the bond length,
C     and then the forces along the bond axis are calculated from the
C     force table (ftab,ftabi) as described above.  In the process the
C     forces from the force table are converted to amu-angstrom-ps units.
C     Finally, the forces along the bond axis are projected onto the
C     various coordinates in the system.

      rad2=0.d0
      do 10 i=1,3
      r(i)=rxyz(i)-rxyz(i+3)
 10   rad2=rad2+r(i)**2

      n=int(4750.d0/rad2)
      rad2=dsqrt(rad2)

      forc= ftab(n)/rad2 + ftabi(n)
      do 20 i=1,3
      for(i)=forc*r(i)*1.1961d0
 20   for(i+3)=-for(i)

C     The frictional forces are calculated in this section following the
C     derivation of Sec IV(B) of Cha. 3.  First the predicted velocity at
C     1/2 time step in the future is calculated from the current velocity
C     and the previous velocity (stored in 'v2').  This is necessary to
C     provide for a second order integration of the friction integral.
```

```
C   the friction is then calculated for each mode of each coordinate
C   and summed in the variable 'fric'.  See also the description of the
C   variables (expr,expi) and (magr,magi) above.

      do 100 i=1,natom3
      vave = vxyz(i)+(vxyz(i)-v2(i))*.5d0
      fric(i) = 0.d0
        do 110 j=1,npar
        freal = frcr(i,j)
        fimag = frci(i,j)
        frcr(i,j)=freal*expr(j)-fimag*expi(j)+vave*magr(j)
        frci(i,j)=fimag*expr(j)+freal*expi(j)+vave*magi(j)
        fric(i)=fric(i) + frcr(i,j)
 110    continue
      v2(i)=vxyz(i)
 100  continue

C   Finally, the random force is calculated and all of the forces are
C   summed together in the variable 'axyz'.  The subroutine 'gauss' is
C   a gaussian random number generator and returnes natom3 unit-
C   normalized gaussian-distributed random numbers in the array 'eps'.
C   These correspond to the variable w_r in Eq. (4.1).  The random
C   forces are then calculated as in Eq. (4.1).  Actually, the random
C   "force" is calculated as an acceleration, and the other "forces"
C   calculated above are converted to accelerations before being summed
C   in the acceleration array 'axyz'.  Finally, the scale factor of .5
C   discussed in Cha. 4 is included in line 30 below, scaling the
C   friction by .5 and the random force by sqrt(.5).

      call gauss(eps,natom3)

      do 50 i=1,natom3
        do 40 j=npar1,2,-1
 40     eeps(i,j) = eeps(i,j-1)
 50   continue

      do 30 i=1,natom3
      eeps(i,1) = b0*eps(i)
        do 60 j=2,npar1
 60     eeps(i,1) = eeps(i,1) - apar(j-1)*eeps(i,j)
 30   axyz(i)=(for(i) - fric(i)*.5)*smass + eeps(i,1)*.707107

      return
      end

C  ****************************************************************
      subroutine energy(rxyz,vxyz)
C  ****************************************************************

C   This subroutine calculated the kinetic and potential energy of the
C   vibrational coordinate of the iodine molecule and the total kinetic
C   energy of the iodine molecule.
```

```
      implicit double precision (a-h,o-z)

      parameter(natm3=6)
      parameter(nrkr=1000)
      parameter(npr=15)

      double precision ftab,ftabi,vtab,vtabi
      double precision eeps
      double precision rxyz(natm3),vxyz(natm3),r(natm3)
      double precision ek,ep,ekt
      double precision rad2
      double precision smassi,smass

      common /lang/smassi,smass
      common /eng/ek,ep,ekt
      common /blk6/ftab(nrkr),ftabi(nrkr),vtab(nrkr),vtabi(nrkr)
      common /blk7/rad2,nover
      common /kexp2/eeps(natm3,npr)

C   The bond length is calculated.  If it is greater than 8 angstroms
C   'nover' is set to 1 and the trajectory will be restarted.

      rad2=0.d0
      do 50 i=1,3
      r(i)=rxyz(i)-rxyz(i+3)
 50   rad2=rad2+r(i)**2
      if(rad2.gt.64.) goto 66

C   The total kinetic energy and the kinetic energy of the vibrational
C   coordinate are calculated here.

      ek=0.d0
      ekt=0.d0
      rad2=dsqrt(rad2)
      do 10 i=1,3
      v=vxyz(i)-vxyz(i+3)
      ekt=ekt+vxyz(i)**2+vxyz(i+3)**2
 10   ek=ek+r(i)*v/rad2
      ek=.25d0*ek*ek*smassi
      ekt=.5d0*ekt*smassi

C   Finally, the vibrational potential energy is calculated using the
C   rkr-based potential table (vtab,vtabi).  This technique is also
C   used in the program "I2XENON" and the reader is referred to the
C   thesis of J. K. Brown for more details of the technique.

      ep=0.d0
      n=int(4750.d0/rad2**2)
      ep=vtab(n) +rad2*vtabi(n)
      ep=ep*1.1961d0
      rad2=rad2**2
```

```
      goto 67
 66   nover=1
 67   return
      end

C  ****************************************************************
      subroutine gauss(eps,n)
C  ****************************************************************

C   This subroutine takes input from a standard random number
C   generator, with values returned between 0. and 1., and generates
C   gaussian distributed random numbers with zero mean and a width (σ)
C   of 1.  I did not write this subroutine, but I am uncertain of its
C   origin and it is therefore included for completeness.

      common /gstart/iy
      double precision eps(1)

C   Function urand is a random number generator from Computer Methods
C   for Mathematical Computations, by Forsythe, et al., p. 246.
C   Some random number generators are not sufficient for these
C   calculations, so some care should be taken if a different one is
C   used.  Inadequate random number generators have been found to
C   greatly decrease the stability of the integrator.
C
      do 100 i=1,n-1,2
 10   x=urand(iy)
      r1=2.*x-1.
      x=urand(iy)
      r2=2.*x-1.
      s=r1**2+r2**2
      if (s.gt.1.) goto 10
      eps(i)=sqrt(-2.*alog(s)/s)
      eps(i+1)=r2*eps(i)
      eps(i)=r1*eps(i)
 100  continue

      if (2*(n/2).eq.n) goto 200

 20   x=urand(iy)
      r1=2.*x-1.
      x=urand(iy)
      r2=2.*x-1.
      s=r1**2+r2**2
      if (s.gt.1.) goto 20
      eps(n)=r1*sqrt(-2.*alog(s)/s)

 200  continue

      return
      end
```

VI. SAMPLE OUTPUT "I2MEXP"

1.00000e-03 2000 0. 300.000 150 62
0. 0.

0.	11560.1924	0.7477	2912.9561	2912.9561
4.0000	9683.1748	514.1491	4587.7817	3971.6970
8.0000	8693.7148	460.8474	4143.1816	3692.0630
12.0000	8050.8618	454.3560	4118.9468	3561.2126
16.0000	7640.6489	451.3124	3720.7451	3079.7842
20.0000	7358.2646	453.6525	3751.3979	3261.7793
24.0000	6969.9263	483.4705	3492.0972	2995.8811
28.0000	6617.9697	486.7811	3204.6516	2784.0293
32.0000	6412.3691	501.2810	3263.8916	2768.6709
36.0000	6175.8901	514.8914	3028.5007	2582.0889
40.0000	6008.3247	528.8706	2791.0696	2291.7561
44.0000	5851.5571	540.4063	3449.0603	3027.2976
48.0000	5767.2515	563.1180	3116.3821	2488.1233
52.0000	5603.1572	565.3472	2845.5559	2374.8313
56.0000	5465.1455	549.8303	2833.7822	2381.7642
60.0000	5354.1538	572.1776	2956.1021	2478.6716
64.0000	5161.1987	582.0976	2938.1318	2403.6514
68.0000	5036.7837	597.1615	2253.1646	1837.0438
72.0000	4838.0435	563.5169	2787.2017	2340.5247
76.0000	4779.6699	578.9578	2612.4814	2111.4736
80.0000	4688.8315	581.3735	2327.7134	1845.8319
84.0000	4565.6802	568.8546	2611.2100	2092.3271
88.0000	4493.0542	579.6082	2507.2722	2057.8733
92.0000	4387.8271	597.2220	2625.0042	2135.4558
96.0000	4324.0293	615.7619	2665.0854	2091.7773
100.0000	4211.1758	584.7887	2430.6458	1871.5676
104.0000	4144.1680	569.2332	2412.2046	1913.8538
108.0000	4055.4131	554.6161	2218.2786	1737.3627
112.0000	3922.3889	549.9896	2338.2205	1925.8241
116.0000	3844.0244	573.0081	1972.9674	1465.9448
120.0000	3740.7961	520.9611	2506.1157	2131.1294
124.0000	3677.5576	543.0220	2519.3533	2000.6793
128.0000	3626.7490	568.3795	2228.1738	1709.9280
132.0000	3547.0442	565.4366	1904.7020	1331.8391
136.0000	3501.9954	566.5342	1982.7156	1569.2136
140.0000	3457.0579	569.5331	2173.5718	1688.4777
144.0000	3369.6716	575.0400	2306.8582	1757.9001
148.0000	3359.6331	607.7102	2154.5479	1653.9822
152.0000	3258.0505	614.4842	1996.6752	1519.9369
156.0000	3200.0889	583.1400	2066.7258	1627.6549
160.0000	3170.6636	570.1511	1895.6631	1333.3131
164.0000	3105.7129	567.3509	2280.4041	1898.3119
168.0000	3059.3611	563.5172	1878.6927	1444.0099
172.0000	2982.5774	578.7531	1964.8795	1449.4800
176.0000	2963.7251	566.7316	2025.5969	1511.6229
180.0000	2917.5647	566.3452	2018.8030	1510.5046
184.0000	2874.3604	553.3651	1792.9139	1293.7822
188.0000	2835.1631	556.2968	1847.6979	1354.5161

192.0000	2798.3472	523.9656	1973.3706	1532.5155
196.0000	2762.1106	532.4000	1698.5513	1316.1810
200.0000	2718.4333	523.0681	1897.2449	1372.6652
204.0000	2682.2026	533.2335	1790.1974	1287.3812
208.0000	2659.2112	535.5054	1773.2655	1297.3906
212.0000	2613.1216	531.6119	1633.5250	1299.2173
216.0000	2588.6731	512.4233	1711.1041	1163.2606
220.0000	2564.3462	515.2520	1686.0262	1195.1686
224.0000	2538.3132	522.3068	1685.1531	1212.3916
228.0000	2504.9670	520.6630	1601.1489	1181.4670
232.0000	2471.3562	535.7469	1584.5538	1156.4340
236.0000	2401.6848	495.8280	1611.1976	1216.9802
240.0000	2376.9282	488.5052	1602.6473	1039.0455
244.0000	2356.7610	482.9694	1431.7002	963.0814
248.0000	2322.1013	502.8672	1548.0308	1068.9397
252.0000	2306.3062	525.7775	1764.7053	1120.5135
256.0000	2275.1611	507.0321	1545.0576	1050.6930
260.0000	2223.2483	505.0471	1809.7234	1155.4099
264.0000	2204.8513	517.4675	1658.3474	1046.4022
268.0000	2170.8057	508.8105	1444.5885	947.9626
272.0000	2130.6814	507.8073	1411.9471	971.1681
276.0000	2127.4343	510.7294	1414.0271	923.0223
280.0000	2092.4746	530.5145	1448.1409	945.6625
284.0000	2051.0012	526.9006	1622.9541	1192.6642
288.0000	2028.3638	495.0981	1615.1317	1089.4591
292.0000	2012.6180	500.6653	1338.2944	814.8387
296.0000	2015.6659	510.7190	1401.0278	905.8057
300.0000	1970.6963	476.5818	1278.3220	832.6859
304.0000	1946.6776	493.7444	1453.6914	880.4163
308.0000	1921.8311	481.7494	1667.2926	990.2775
312.0000	1914.5936	500.6960	1304.0775	813.8951
316.0000	1887.2297	499.7447	1463.5630	957.3034
320.0000	1858.9388	502.9406	1331.1013	875.6343
324.0000	1810.8312	458.6109	1389.8286	876.8123
328.0000	1792.3663	447.1033	1195.0770	822.0078
332.0000	1754.4154	435.4118	1507.1243	903.1356
336.0000	1745.9254	445.6682	1371.8513	893.7214
340.0000	1747.3575	445.0387	1445.8132	778.6466
344.0000	1728.7118	441.0842	1421.0883	956.6512
348.0000	1705.4844	447.5644	1267.1799	769.9052
352.0000	1694.7272	448.6219	1332.3433	828.6503
356.0000	1661.2375	451.5166	1418.8293	906.1428
360.0000	1636.6263	429.4163	1353.0260	880.2971
364.0000	1615.3684	417.4428	1347.0396	759.4026
368.0000	1619.0027	448.1198	1268.6617	818.3054
372.0000	1608.0039	436.2839	1102.6909	698.2794
376.0000	1569.9772	429.4434	1184.5034	719.3941
380.0000	1577.0948	441.9138	1532.5643	905.2527
384.0000	1543.7753	441.7476	1255.7035	725.3937
388.0000	1516.5353	421.9795	1407.2126	808.4236
392.0000	1492.9623	433.2946	1259.3638	739.4848
396.0000	1471.7042	441.9077	1197.2961	779.5206

400.0000	1462.2759	419.9994	1204.6890	704.6997
404.0000	1431.9384	435.4955	1231.9418	789.8763
408.0000	1418.0151	431.1653	1245.1036	742.1560
412.0000	1402.2808	444.2040	1197.2256	706.6040
416.0000	1384.3115	431.2777	1118.8993	647.1771
420.0000	1375.2104	414.5921	1155.6394	704.0056
424.0000	1345.2908	410.3203	1205.0687	670.8033
428.0000	1334.6898	403.8003	1203.4033	607.2258
432.0000	1312.8942	380.6803	935.0182	566.8735
436.0000	1291.7415	396.3160	1097.4904	599.9177
440.0000	1280.7052	406.6926	957.3923	533.5691
444.0000	1274.5947	398.3354	1075.1869	594.3573
448.0000	1241.1838	384.4865	1005.7128	535.5132
452.0000	1240.8888	392.1182	1061.7140	531.4199
456.0000	1213.7256	380.4577	1085.0732	596.8441
460.0000	1211.4823	368.8305	1144.4008	589.4124
464.0000	1177.5475	374.4355	964.1932	448.3803
468.0000	1182.0886	391.1276	947.5753	420.4963
472.0000	1154.0795	377.3023	885.7092	482.4267
476.0000	1146.3849	377.9941	1027.3242	547.2398
480.0000	1151.6138	386.2561	1086.4353	534.4333
484.0000	1147.7317	399.2293	1294.8859	594.4832
488.0000	1160.4840	410.8235	1030.5405	550.7576
492.0000	1133.7870	399.9296	1009.3494	494.0874
496.0000	1118.0128	409.5198	900.0882	528.7245
500.0000	1098.5267	405.3016	1018.0562	519.5702
504.0000	1089.4650	411.3709	997.7358	507.5717
508.0000	1070.4462	397.5539	1026.0035	572.4506
512.0000	1059.1414	393.5566	955.5870	471.8267
516.0000	1032.3407	368.0522	1098.7654	489.0554
520.0000	1030.2428	368.1724	963.7539	467.3589
524.0000	1026.9553	384.9178	1206.1604	559.4647
528.0000	1011.8091	366.9922	973.8162	444.9461
532.0000	1006.5298	380.4351	959.5434	466.9827
536.0000	1007.3096	415.5062	940.9011	466.2606
540.0000	994.2856	422.2887	1053.4972	503.7625
544.0000	972.9785	420.8181	998.8173	485.1467
548.0000	965.6558	422.9778	1078.4989	499.6131
552.0000	963.7000	425.1677	902.0887	463.2231
556.0000	951.9882	422.2589	1056.3854	529.0507
560.0000	947.9466	401.4041	959.7426	376.8813
564.0000	943.4919	401.5354	1008.2579	538.3574
568.0000	937.9088	408.3051	996.9477	474.7726
572.0000	938.1332	404.1285	955.6819	472.8659
576.0000	935.3958	393.8807	1108.0079	496.2622
580.0000	941.4200	392.3081	993.6860	459.0787
584.0000	926.0236	394.9339	934.5616	385.8978
588.0000	916.2811	384.4836	998.6096	431.1057
592.0000	932.4585	408.8212	867.7282	379.7058
596.0000	916.1964	384.7968	903.8621	441.3227
600.0000	897.3652	371.0936	960.7576	423.2967

PROGRAM LISTING MD3DKE

```
/*
*   Program "MD3DKE"
*   David Smith
*   Aug. 11, 1989
*
*   The following C program consists of a molecular dynamics simulation
*   of the dissociation, recombination, and relaxation dynamics of an
*   iodine molecule in liquid xenon with graphical output to a video
*   tape recorder.  It is designed to run on a Silicon Graphics (SG)
*   3130 workstation.  The molecular dynamics portions of the program
*   are for the most part simply a translation of the molecular
*   dynamics program of J. Keenan Brown titled "I2XENON", a listing of
*   which is contained in his PhD thesis.  These portions of this
*   program will not be described in detail in the comments, except
*   where the "translation" of the code from Fortran to C is unclear,
*   and the reader is referrred to Cha. 2 and Appendix A of J. K.
*   Brown's PhD thesis for more details.  The rest of this program
*   consists of graphically representing the iodine and xenon motions
*   and storing the resulting images on video tape.
*/
#include "gl.h"
#include "stdio.h"
#include "math.h"
#include "device.h"
#include <termio.h>
#include "fcntl.h"
#include <sys/ioctl.h>
/*
*   The system size considered here is 109 atoms, 107 xenon atoms and
*   the iodine molecule.  This is a small system size for simulations
*   of the dynamics of dissociation and recombination.  The small size
*   allows for calculation of the trajectory directly on the Silicon
*   Graphics Iris 3130 workstation while simultaneously dumping the
*   results to video tape.  For many possible graphical applications it
*   may be necessary to simulate larger of more complicated systems.
*   In this case it would be desirable to calculate the trajectories on
*   a more powerful computer, storing relevant particle postions, and
*   using the Silicon Graphics only to graphically represent the
*   resulting system.
*/
#define NATM 109
#define NATM3 3*NATM
#define NATM1 NATM-1
#define NN123 NATM*NATM1/2
#define NN1234 2*NN123
/*
*   'crosshatch1' is a pattern definition which allows for drawing
*   "transparent" spheres.  This technique allows the iodine molecule
*   to be seen in its local solvent environment without artificially
*   removing solvent molecules which might otherwise hinder the
```

```
*   visibility of the iodine.  See the Graphics Programming manual for
*   more details of the nature of pattern definitions.
*/
Pattern16 crosshatch1 = {0xaaaa, 0x5555, 0xaaaa, 0x5555,
                         0xaaaa, 0x5555, 0xaaaa, 0x5555,
                         0xaaaa, 0x5555, 0xaaaa, 0x5555,
                         0xaaaa, 0x5555, 0xaaaa, 0x5555};
long float rxyz[NATM#],vxyz[NATM3],axyz[NATM3];
long float alxyz[NATM3],r1xyz[NATM3];
long float r2t[NN123],rxt[NN123],ryt[NN123],rzt[NN123];
long float ftab[1000],ftabi[1000],vtab[1000],vtabi[1000];
long float ekatom[NATM],ekavg,epatom[NATM];
int orderz[NATM];
int natom,natom1,natom3,nn123;
long float h,hi,h2,h26;
long float ek,ek1,ep,et,ei2,eki2;
long float c1,c2,smass,smassi;
long float c1d,c2d,r2cut,r2max;
long float c1i2,c2i2,c1i2d,c2i2d;
long float rmin,vzero,vzix;
long float alxyz,alxyzi3;
long float gcon,rad,radi2,radi2a;
long float rci2x,rci2y,rci2z;
long float rsw,akt;
long float plu,plud;
long float sktm;
FILE *fdin,*finit,*fdout,*fenergy,*fopen(),*finfo;
int fd;
int ktm,loop;
int neqtemp,nequil,ndiss,nvib;
int iflag,iflag1,keflag;
long float rzero,rmax,elj,sigma;
long seedval;
/*****************************************/
main()
/*****************************************/
{
    register i,j,j1;

/*  Disables the screen blanking feature on the Silicon Graphics.  This
*   is essential since otherwise the screen will automatically blank
*   out during the execution of the program.  The value is reset when
*   the program is executed.
*/
    blanktime(0);
/*
*   open_line opens the connection between the SG and the Diaquest
*   computer which controls video recording system.
*/
    open_line();
/*
*   set the time for the initial frame to be drawn
```

```
/*
    fprintf(fdout,"setin 00:05:04:00\r");
/*
*   write the response from the Diaquest computer to the silicon
*   graphics screen.  If execution is not successfully completed the
*   message will indicate this.
*/
    get_response();
/*
*   open various files
*/
    open_files();
/*
*   initialize the random number generator
*/
    seedval=39402139;
    srand48(seedval);
/*
*   restart if the trajectory does not recombine
*/
    restart:
/*
*   initialize the trajectory
*/
    init();
/*
*   graphically sort the particles such that they are stored
*   according to their z-axis positions.  This is the initial graphical
*   sort for the system.  Subsequent sorts use the subroutine sortgr().
*/
    sortgr1();
/*
*   calculates the energy of the system and stores the results
*/
    energy();
/*
*   various parameters defining the length of the trajectory
*/
    ndiss = 500;
    nvib = 4000;
    nequil = 10;
    neqtemp = 5;
/*
*   time index
*/
    ktm=0;
/*
*   loop to equilibrate the system temperature and the iodine energy.
*   The time step in these simulations is 1 fs.  Output is drawn to
*   video tape, subroutine draww(), every 4 fs.  checkinput() looks to
*   see if the exit signal, button zero on the button box, has been
*   pushed and exits the program if it has.  At this point the program
```

```
*   is not written such that the trajectory can be restarted, although
*   information which would allow for a restart is stored in an output
*   file.  tempc() equilibrates the system temperature, and bondc() the
*   iodine temperature.  These routines are translated from "I2XENON".
*/
    for(jl=0; jl<1; jl++){
    for(i=0; i<1; i++) {
         checkinput();
         draww();
         for (j=0; j<2; j++) {
              integ();
         }
    energy();
    fflush(fenergy);
    tempc();
    bondc();
    }
    }
/*
*   loop to equilibrate the system temperature.  'neqtemp' contains
*   number of iterations of this loop to be calculated.
*/
    for(jl=0; jl<neqtemp; jl++){
    for(i=0; i<10; i++) {
         checkinput();
         draww();
         for (j=0; j<2; j++) {
              integ();
         }
    energy();
    fflush(fenergy);
    }
    tempc();
    }
/*
*   loop to calculate motion of the iodine molecule at equilibrium for
*   a time.  'nequil' contains the number of iterations of this loop to
*   be calculated.
*/
    for(jl=0; jl<nequil; jl++){
    for(i=0; i<10; i++) {
         checkinput();
         draww();
         for (j=0; j<2; j++) {
              integ();
         }
    energy();
    fflush(fenergy);
    }
    }
/*
```

```
*   the iodine molucule is dissociated and the time index is reset to
*   now indicate the time from dissociation
*/
    diss();
    ktm = 0;
/*
*   'iflagl' is a flag which indicates the iodine motion now
*    proceeds on the repulsive potential.
*/
    iflagl = 1;
/*
*   loop to calculate dissociation. 'ndiss' is the maximum number of
*   iterations allowed before the trajectory is assumed to have crossed
*   to the ground state.  Crossing occurs when the iodine bond length
*   surpasses a certain switching value, 'rsw', defined in init().
*/
    for(i=0; i<ndiss; i++){
         checkinput();
         draww();
         for (j=0; j<2; j++) {
              integ();
         }
         energy();
         fflush(fenergy);

         if(r2t[0]>rsw){
         potchk();
         iflagl = 0;
         break;
         }
    }
/*
*   loop to calculate vibrational relaxation of the iodine molecule.
*   nvib contains the number of iterations of this loop to be
*   calculated.  The trajectory is restarted if the iodine atoms move
*   beyond ~ 10 angstroms apart.  'keflag' is a flag to indicate
*   whether the kinetic energy of the fragments should be displayed or
*   not.  The kinetic energy is displayed during the dissociation
*   process but not during the vibrational relaxation.
/*
    for (i=0; i<nvib; i++) {
         if(r2t[0]>98.){
         goto restart;
         }
         if(r2t[0]<20.)
              keflag = 0;
         checkinput();
         energy();
         fflush(fenergy);
         draww();
         for (j=0; j<2; j++) {
              integ();
```

```
        }
    }


}
/*****************************************/
checkinput()
/*****************************************/
{
    int dev;
    short val;
/*
*   See the Graphics Programming manual or the graphics tutorial for
*   a description of device polling.  This section checks to see if
*   button 0 (SW0) on the button box has been pressed.  If it has, the
*   trajectory is stopped, the positions and velocities of all the
*   particles are saved, the screen blanking function is restored, and
*   the program is exited gracefully.
/*
    while (qtest()){
         dev=qread(&val);
         switch(dev){
         case SW0:
              saveinfo();
              blanktime(13000);
              gexit();
              exit(1);
              break;
         default:
              break;
         }
    }
}
/*****************************************/
saveinfo()
/*****************************************/
{
    int i;

    fprintf(finfo,"%lf %lf %lf %lf %lf %lf"
            ,h,rzero,rmax,elj,sigma,smass);
    fprintf(finfo,"%lf %d\n",alxyz,natom);
    fprintf(finfo,"%d \n",ktm);
    fprintf(finfo,"%d %d %d %d\n",neqtemp,nequil,ndiss,nvib);
    for(i=0; i<natom; i++){
        fprintf(finfo,"%lf %lf%lf\n"
                ,rxyz[3*i+0],rxyz[3*i+1],rxyz[3*i+2]);
        fprintf(finfo,"%lf %lf %lf\n"
                ,vxyz[3*i+0],vxyz[3*i+1],vxyz[3*i+2]);
    }
    fflush(finfo);
```

```
}

/*****************************************/
drawv()
/*****************************************/
/*
*   This subroutine is responsible for drawing the system.
*/
{
    register i,j,kl;
    float x1,x2,x1a,x2a;
    float ratz,ratz1;
    float radf;
    float dpth;
    long float ke;
    long float rx1,ry1,rx2,ry2;
    char a[80];
    int iflag2,iflag3;
    int cj,cke;
/*
*   call the graphics sorting routine.  See this subroutine for
*   definitions of the colors used to draw the various spheres.
*/
    sortgr(rxyz);
/*
*   draw the background
*/
    color(BLACK);
    clear();
/*
*   iflag2 and iflag3 are local flags used in drawing the system
*/
    iflag2 = 0;
    iflag3 = 0;

    for(i=0; i<natom; i++){
    kl=orderz[i];
/*
*   colors are determined from the total energy, as defined in
*   subroutine energy().  Kinetic energy could be used equally well.
*   The value of '800' in the next line is an empirical value which
*   gives a reasonable range of colors for the equilibrium system,
*   where most of the spheres should look cold.
*/
    ke = ekatom[kl/3]+epatom[kl/3]+800.;
/*  ke = ekatom[kl/3];  */
/*
*   There are 100 possible colors for each sphere.  The total range of
*   energies represented by these colors is defined here to be 10 times
*   the average kinetic energy of each particle.
*/
    cke = (int)(ke*10./ekavg);
```

```
    if(cke>99) cke=99;
    if(cke<0) cke=0;
/*
*   perspective variable, takes values ranging from .9 to 1 depending
*   on the z-coordinate and scales the radius of the sphere.
*/
    ratz=(float)(rxyz[kl+2]*alxyzi3+.9);
/*
*   x1 and x2 give x and y coordinates of the center of the sphere.
*   The coordinate (290,190) is the center of the system.  The
*   perspective variable is used here to shift the center of each
*   sphere toward the center of the system, to be consistent with
*   changing the size of each sphere.
*/
    x1=(gcon*rxyz[kl]-290.)*ratz+290.;
    x2=(gcon*rxyz[kl+1]-190.)*ratz+190.;
/*
*   if either of the iodine atoms is to be drawn, the black circles
*   representing the I-Xe interaction radius for both atoms are drawn
*   as long as the iodine atoms are in close proximity (to eliminate
*   overlap of these black circles with the iodine spheres).  'iflag2'
*   is used to indicate whether these black circles have already been
*   drawn, thus avoiding drawing them twice.  'iflag3' is used below to
*   determine is a transparent sphere should be drawn.  Transparent
*   spheres are only necessary after an iodine atom has been drawn.
*/
    if (kl == 0 || kl == 3) {

      iflag3 = 1;

      if(r2t[0]<20.0){
        if(iflag2 == 0){
         iflag2 = 1;
         color(BLACK);
         radf = (float)(ratz*radi2a);
         circ(x1,x2,radf);
         if(kl == 0){
           ratz1=(float)(rxyz[kl+5]*alxyzi3+.9);
           x1a=(gcon*rxyz[kl+3]-290.)*ratz+290.;
           x2a=(gcon*rxyz[kl+4]-190.)*ratz+190.;
           radf = (float)(ratz1*radi2a);
           circ(x1a,x2a,radf);
         }
         else{
           ratz1=(float)(rxyz[kl-1]*alxyzi3+.9);
           x1a=(gcon*rxyz[kl-3]-290.)*ratz+290.;
           x2a=(gcon*rxyz[kl-2]-190.)*ratz+190.;
           radf = (float)(ratz1*radi2a);
           circ(x1a,x2a,radf);
         }
        }
      }
```

```
      else{
         color(BLACK);
         radf = (float)(ratz*radi2a);
         circ(x1,x2,radf);
      }
/*
*   The iodine sphere is now drawn.  Forty coincentric circles of
*   gradually diminishing radius and gradually increasing intensity are
*   drawn.  'cj' scales the intensity, with the color determined from
*   'cke',  and 'radf' is the radius.
*/
      for(j=0; j<40; j++){
      cj = 10 + cke*40 + j;
      color(cj);
      radf = (float)(ratz*radi2*(float)(39-j)/39.);
      circf(x1,x2,radf);
      }
    }
    else{
/*
*   If an iodine atom has already been drawn, check and see whether the
*   present xenon sphere overlaps either iodine atom sufficiently to
*   warrent representing is as a transparent sphere.  If so, set the
*   pattern to transparent before drawing the sphere.
*/
    if(iflag3 == 1){
       rx2 = rxyz[k1]-rxyz[3];
       ry2 = rxyz[k1+1]-rxyz[4];
       rx1 = rxyz[k1]-rxyz[0];
       ry1 = rxyz[k1+1]-rxyz[1];
       rx2 *= rx2;
       ry2 *= ry2;
       rx1 *= rx1;
       ry1 *= ry1;
       if(rx2+ry2 < 7. || rx1+ry1 < 7.){
         setpattern(4);
       }
    }
/*
*   The xenon sphere is now drawn.  Forty coincentric circles of
*   gradually diminishing radius and gradually increasing intensity are
*   drawn.  'cj' scales the intensity, with the color determined from
*   'cke',  and 'radf' is the radius.
*/
    for(j=0; j<40; j++){
    cj = (int)(10.+(cke*40) + (float)j);
    color(cj);
    radf = (float)(ratz*rad*(float)(39-j)/39.);
    circf(x1,x2,radf);
    }
/*
*   reset pattern
```

```
*/
    setpattern(0);
    }
    }
/*
*   draw various labels.  Kinetic energy of the iodine atoms is only
*   displayed during dissociation, as stored in 'keflag'.
*/
/*
*   display time (ps)
*/
    color(YELLOW);
    sprintf(a,"%7.3f ps",(float)h*(float)ktm);
    cmov2i(350,430);
    charstr(a);
/*
*   display iodine atom separation
*/
    sprintf(a,"%7.2lf ang",_lsqrt(r2t[0]));
    cmov2i(450,430);
    charstr(a);
/*
*   display kinetic energy of iodine atoms if appropriate
*/
    if(keflag == 1){
    sprintf(a,"%7.0lf cm-1 ke",eki2/1.1961);
    cmov2i(100,390);
    charstr(a);
    }
/*
*   display iodine vibrational energy
*/
    sprintf(a,"%7.0lf cm-1 ",ei2/1.1961);
    cmov2i(100,430);
    charstr(a);
/*
*   determine whether the video tape recorder has finished drawing the
*   previous screen
*/
    get_ready();
/*
*   display new screen
*/
    swapbuffers();
/*
*   output to video tape recorder and check for proper response
*/
    fprintf(fdout,"editp\r\n");
    get_response();
}
/*****************************************/
sortgr()
```

```
/*****************************************/
/*
*   graphical sorting routine.
*/
{
    register i,j;
    long float radin;
    int in,ibuf;
    long float buf;
    long float z[NATM];
    register long float *pr;
    long float rcx,rcy,rcz;

    pr=rxyz;
/*
*   the system coordinates are adjusted each time sortgr() is called.
*   The z-coordinates of each atom are shifted such that the center-of-
*   mass of the iodine molecule is at coordinate rci2z.  This maintains
*   the iodine molecule at a depth which is convenient for viewing.
*   The x and y coordinates are shifted similarly to rci2x and rci2y
*   except that they are shifted gradually toward the correct
*   coordinate.  This ensures that the iodine molecule will not drift
*   off of the screen, yet also avoids the unrealistic appearance of
*   collective system motion which would occur is coordinates were
*   adjusted instantaneously.  The scaling factor of 1000 is purely
*   empirical, and appears to we quite adequate.
*/
    rcx = (rci2x - (*pr + (*(pr+3)))/2.)/1000.;
    pr++;
    rcy = (rci2y - (*pr + (*(pr+3)))/2.)/1000.;
    pr++;
    rcz = rci2z - (*pr + (*(pr+3)))/2.;

    pr=rxyz;
/*
*   The iodine atoms are shifted here
*/
    for(i=0; i<2; i++){
         *pr += rcx;
         pr++;
         *pr += rcy;
         pr++;
         *pr += rcz;
         pr++;
    }
/*
*   followed by shifting all of the xenon atoms.  If any atom is
*   shifted over a box boundary, it is placed on the opposite side of
*   the box. This ensures that all of the atoms remain in the primary
*   box for the purposes being displayed.  One result of this is that
*   atoms are seen to "pop" in and out of the front of the screen, a
*   necessary yet sometimes distracting feature.
```

```
*/
    for(i=2; i<natom; i++){
         *pr += rcx;
         radin = *pr/alxyz;
         in = (radin >= 0 ? (int)radin : (int)(radin-1.));
         *pr -= in*alxyz;
         pr++;
         *pr += rcy;
         radin = *pr/alxyz;
         in = (radin >= 0 ? (int)radin : (int)(radin-1.));
         *pr -= in*alxyz;
         pr++;
         *pr += rcz;
         radin = *pr/alxyz;
         in = (radin >= 0 ? (int)radin : (int)(radin-1.));
         *pr -= in*alxyz;
         pr++;
    }

/*
*   Next the atoms are all sorted according to their Z-coordinates.
*   This allows draww() to draw every sphere without making a
*   complicated determination of which portions of a givin sphere
*   should be visible to the viewer.  Spheres are drawn back to front.
*   the array 'orderz' contains the atom numbers in the order in which
*   they should be drawn.
*/
    for(j=0; j<natom; j++)
    z[j]=rxyz[3*j+2];

    for(j=0; j<natom; j++){
    ibuf=0;
    buf=z[0];
    for(i=1; i<natom; i++){
         if(z[i]<buf){
         buf=z[i];
         ibuf=i;
         }
    }
    orderz[j]=ibuf*3;
    z[ibuf]=1000.;
    }

}
/*****************************************/
sortgr1()
/*****************************************/
/*
*   initial graphics sort.  This routine is identical to sortgr()
*   except that the iodine molecule is initially placed in the middle
*   of the screen instead of being gradually moved toward the center.
/*
```

```
{
    register i,j;
    long float radin;
    int in,ibuf;
    long float buf;
    long float z[NATM];
    register long float *pr;
    long float rcx,rcy,rcz;

    pr=rxyz;

    rcx = rci2x - (*pr + (*(pr+3)))/2.;
    pr++;
    rcy = rci2y - (*pr + (*(pr+3)))/2.;
    pr++;
    rcz = rci2z - (*pr + (*(pr+3)))/2.;

    pr=rxyz;

    for(i=0; i<2; i++){
         *pr += rcx;
         pr++;
         *pr += rcy;
         pr++;
         *pr += rcz;
         pr++;
    }

    for(i=2; i<natom; i++){
         *pr += rcx;
         radin = *pr/alxyz;
         in = (radin >= 0 ? (int)radin : (int)(radin-1.));
         *pr -= in*alxyz;
         pr++;
         *pr += rcy;
         radin = *pr/alxyz;
         in = (radin >= 0 ? (int)radin : (int)(radin-1.));
         *pr -= in*alxyz;
         pr++;
         *pr += rcz;
         radin = *pr/alxyz;
         in = (radin >= 0 ? (int)radin : (int)(radin-1.));
         *pr -= in*alxyz;
         pr++;
    }

    for(j=0; j<natom; j++)
    z[j]=rxyz[3*j+2];

    for(j=0; j<natom; j++){
    ibuf=0;
    buf=z[0];
```

```
    for(i=1; i<natom; i++){
         if(z[i]<buf){
         buf=z[i];
         ibuf=i;
         }
    }
    orderz[j]=ibuf*3;
    z[ibuf]=1000.;
    }

}
/*****************************************/
open_files()
/*****************************************/
/*
*   open energy and output files.  'md.save' contains the output file
*   from the program when it is exited gracefully.  Potentially,
*   trajectories could be restarted from this file.  'energy.md'
*   contains energy information on the trajectory.
*/
{
    if((finfo=fopen("md.save","w")) == NULL){
         printf("error opening save file\n");
         exit(1);
    }
    if((fenergy=fopen("energy.md","w")) == NULL){
         printf("error opening energy file\n");
         exit(1);
    }

}
/*
*   gaussian random number generator.  Places 'n' normalized gaussian
*   distributed random numbers in the array at *peps.  Uses drand48(),
*   the C random number generator on the SG 3130.
*/
/*****************************************/
float gauss(peps,n)
/*****************************************/
int n;
float *peps;
{

    float x,r1,r2;
    float s;
    int i;

    for(i=0; i<n-1; i+=2){

    do{
         x=drand48();
         r1=2.*x-1.;
```

```
        x=drand48();
        r2=2.*x-1.;
        s=r1*r1+r2*r2;
        }while(s > 1.);
    *peps=sqrt(-2.*log(s)/s);
    *(peps+1)=r2**peps;
    *peps*=r1;
    peps+=2;
    }

    if(2*((n)/2)==n)
        return;
    else{
    do{
        x=drand48();
        r1=2.*x-1.;
        x=drand48();
        r2=2.*x-1.;
        s=r1*r1+r2*r2;
        }while(s > 1.);
    *peps=r1*sqrt(-2.*log(s)/s);
    }
    return;
}

static struct termio term;
/*****************************************/
open_line()
/*****************************************/
/*
*   opens connections to Diaquest computer.
*/
{
    int fd;

    if ((fd=open("/dev/ttyd3",O_WRONLY)) == NULL) {
        printf("failure opening file\n");
        abort();
    }

    ioctl(fd,TCGETA,&term);
    term.c_cflag -= 4;
    term.c_lflag -= 8;
    ioctl(fd,TCSETA,&term);

    if ((fdout=fopen("/dev/ttyd3","w")) == NULL) {
        printf("error opening ttyd3 for output\n");
        abort();
    }

    if ((fdin=fopen("/dev/ttyd3","r")) == NULL) {
        printf("error opening ttyd3 for input\n");
```

```
        abort();
    }

    close(fd);

}
/*****************************************/
get_response()
/*****************************************/
/*
*   reads output from Diaquest computer and prints it to the screen
*/
{
    char c;

    while((c=fgetc(fdin)) != '\n') {
        putchar(c);
    }
    putchar('\n');
}
/*****************************************/
get_ready()
/*****************************************/
/*
*   Subroutine to read output from the Diaquest computer when the video
*   tape recorder is ready to draw another frame.
*/
{
    char c;

    while((c=fgetc(fdin)) != '\n') {
        putchar(c);
    }
    putchar('\n');
}
/*****************************************/
init()
/*****************************************/
/*
*   Subroutine to initialize the trajectory.
*/
{
    int i,j,kl;
    long float all;
    int ntot,j3;
    long float eljix,sigix;
    long float scale,temp;
    float gauss();
    float eps[NATM3];
    float a,a2;
    FILE *frkr;
/*
```

```
*   initialize the graphics.
*/
    graphinit();
/*
*   open input file
*/
    if((finit=fopen("start.3.0","r")) == NULL){
         printf("error opening start \n");
         exit(1);
    }
/*
*   read relevant information in the input file header.  Typical values
*   are
*   h      =  .001 ps
*   rzero  =  10 angstroms
*   rmax   =  11 angstroms (Not used in this program)
*   elj    =  616 amu*angstroms**2/ps**2, (Xe-Xe ε = 154 cm-1,
*             times 4).
*   sigma  =  4.1 (Xe-Xe σ)
*   smass  =  131.3 (xenon mass)
*   alxyz  =  19.8725 (3.0 g/cc)
*   natom  =  109
*
*   This input file also contains the positions and velocities of the
*   109 atom system in an appropriate initial equilibrium
*   configuration.  For simulations at different densities, the startup
*   files are not available in this form and may need to be generated.
*/
fscanf(finit,"%lf%lf%lf%lf%lf%lf",&h,&rzero,&rmax,&elj,&sigma,&smass);
    fscanf(finit,"%lf %d",&alxyz,&natom);
/*
*   scale factors for alxyz to give densities of 2.4 and 1.8 g/cc
*/
/*  alxyz = alxyz*1.0772173;  */
/*  density = 2.4  */
/*  alxyz = alxyz*1.1856311;  */
/*  density = 1.8  */
/*
*   read in initial positions and velocities
*/

    for(i=0; i<natom; i++){
    fscanf(finit,"%lg%lg%lg",&rxyz[3*i],&rxyz[3*i+1],&rxyz[3*i+2]);
    fscanf(finit,"%lg%lg%lg",&vxyz[3*i],&vxyz[3*i+1],&vxyz[3*i+2]);
    }

    fclose(finit);
/*
*   read in force and potential energy tables.  Calculations of iodine
*   force and potential energy proceeds as in "I2XENON".
*/
    if((frkr=fopen("rkri2g","r")) == NULL){
```

```
        printf("error opening rkri2g \n");
        exit(1);
    }

    for(i=0; i<1000; i++)
    fscanf(frkr,"%d %g %lg %lg %g ",&j3,&a,&ftab[i],&ftabi[i],&a2);
    for(i=0; i<1000; i++)
    fscanf(frkr,"%d %g %lg %lg %g ",&j3,&a,&vtab[i],&vtabi[i],&a2);
/*
*   convert values to amu-ang-ps units
*/
    for(i=0; i<1000; i++){
    ftab[i] *= 1.1961;
    ftabi[i] *= 1.1961;
    vtab[i] *= 1.1961;
    vtabi[i] *= 1.1961;
    }

    fclose(frkr);
/*
*   initialize flags.
*/
    iflagl = 0;
    keflag = 0;
/*
*   initialize various other parameters, as in "I2XENON"
*/
    eljix = 225.*4.*1.1961;
    sigix = 3.94;
    plu=1.1961*4.0e07;
    plud=9.5*plu;
    c2i2 = eljix*pow((float)sigix,6.);
    cli2 = c2i2*c2i2/eljix;
    cli2d = 12.*cli2;
    c2i2d = 6.*c2i2;
    h2=h*h;
    h26=h2/6.;
    hi=1./h;
    r2cut=rzero*rzero;
    r2max=rmax*rmax;
    elj *= 1.1961;
    c2 = elj*pow((float)sigma,6.);
    cl = c2*c2/elj;
    cld = 12.*cl;
    c2d = 6.*c2;
    vzero = 1./pow((float)rzero,6.);
    vzix = vzero*(cli2*vzero-c2i2);
    vzero *= (cl*vzero-c2);
    rmin = sigma*pow(2.,(1./6.));
    rmin *= rmin;
    natoml = natom-1;
    natom3 = natom*3;
```

```
    nnl23 = (natom*natoml)/2;
    scale = sqrt(.831*280./(float)(smass));
    smassi = 126.9;
    ekavg = 1.5*280.*.831;
    rsw = 4.998;
    rsw *= rsw;
    akt = 280.*1.5*.831*(long float)natom;
    sktm = _lsqrt(.831*280./smassi);

/*
*   The following quantities are associated with graphics routined in
*   this program.
*   alxyz3     =    perspective variable.  The spheres are scaled
*                   slightly to appear smaller when farther away.  The
*                   radius of each sphere is adjusted from 1 to .9 times
*                   its normal value depending on its z-coordinate.  See
*                   draww() for use of this variable.
*   gcon       =    size variable. 'gcon' scales the size of every
*                   object and insures that the relevant parts of the
*                   system are displayed on the screen.
*   rad        =    radius of xenon spheres.  (L-J sigma)
*   radi2a     =    radius of iodine atom which give an appropriate I-Xe
*                   interaction sigma.  This radius is shown as a black
*                   circle in the graphical representation.  A smaller
*                   radius for the iodine spheres is desirable given the
*                   considerably shorter I-I bond length.
*   radi2      =    radius of iodine spheres, somewhat shorter than the
*                   equilibrium bond length.
*   rci2z      =    z-coordinate of the iodine molecule center-of-mass.
*                   The entire system is translated in sortgr() such
*                   that this is true.
*   rci2x,rci2y    =    x and y coordinates of the center of the
*                   screen.  The iodine center-of-mass is initially
*                   placed at this locations and the entire system is
*                   gradually restored to this position during the
*                   simulations.
*/
    alxyzi3 = .1/alxyz;
    gcon=717./alxyz;
    rad=sigma*gcon/2.;
    radi2a = 3.78*gcon/2.;
    radi2=(2.60)*gcon/2.;
    rci2z = alxyz - 2.0*sigma;
    rci2x = alxyz/2.2;
    rci2y = alxyz/2.5;

/*
*   initialize velocities
*/
    gauss(eps,natom3);
    for(i=0; i<natom3; i++)
    vxyz[i] = eps[i]*scale;
```

```
}
/*****************************************/
graphinit()
/*****************************************/
/*
*   This subroutine is responsible for initializing the graphics.
*   See the Graphics Reference manual and the Graphics Programmer's
*   manual for more details.
*/
{

    register i,j;
    int bl,rd,gr,cj;
    int bll,rdl,grl;
    long float ke;
/*
*   open the graphics window
/*
    prefposition(0,1023,0,767);
    winopen("md3dke");
/*
*   initialize the doublebuffering graphics feature.
/*
    doublebuffer();
    gconfig();
    frontbuffer(TRUE);
    color(BLACK);
    clear();
    frontbuffer(FALSE);
/*
*   enable button 0, used for exiting the program.
/*
    qdevice(SW0);
/*
*   Define the color map.  In this version of the program the color of
*   each sphere is a function of the atom's total (or kinetic) energy,
*   'blue' being the coldest and 'yellow' being the hottest
*   temperature.  There are 100 total colors assigned ranging smoothly
*   from blue through red to yellow.  In addition, each sphere is drawn
*   using 40 different intensities to give an illusion of depth.  That
*   is, each "sphere" actually consists of 40 circles drawn
*   sequentially one on top of the other, with each circle being
*   slightly brighter and slightly smaller than the previous circle.
*   The result looks like a reflective ball with a bright spot in the
*   middle and dark edges.  This scheme therefore requires 4000 total
*   colors.  Specific details of this color map, i.e. the maximum
*   intensities, etc., are simply a matter of taste and may be modified
*   at will.
/*
    for(i=0; i<100; i++){
```

```
      ke = (float)i/25.;
/*
*   The first set of 25 colors:  set blue to a maximum value of 205
*   (out of a possible 255) and scale red up to a value of 205.
/*
      if(ke<1.) {
        bl = 205;
        rd = (int)(ke*205.);
        gr = 0;
      }
/*
*   The second set of 25 colors:  set red to a value of 205 and scale
*   blue down from 205 to 0.
/*
      else if(ke<2.){
        rd = 205;
        gr = 0;
        bl = 205-(int)((ke-1.)*205.);
      }
/*
*   The final set of 50 colors:  set red to a value of 205, blue to 0,
*   and scale green up to a value of 205.
/*
      else if(ke<4.){
        rd = 205;
        bl = 0;
        gr = (int)((ke-2.)*102.);
      }
      else {
        rd = 205;
        bl = 0;
        gr = 205;
      }
/*
*   For each color defined above, define 40 intensities and place them
*   in the color map beginning with mapcolor(10).  See subroutine
*   draww() for the method used to actually draw the spheres.
*/
      for(j=0; j<40; j++){
        rdl = rd+(int)((float)((205-rd)*j)/40.);
        bll = bl+(int)((float)((205-bl)*j)/40.);
        grl = gr+(int)((float)((205-gr)*j)/40.);
        cj = j+10+i*40;
        mapcolor(cj,rdl,grl,bll);
      }
    }
/*
*   Define the pattern for drawing transparent spheres.
*/
    defpattern(4,16,crosshatch1);

}
```

```
/*****************************************/
integ()
/*****************************************/
/*
*   Integration subroutine.  Identical to that in "I2XENON".  Two
*   integration steps are taken for each call to integ().  New
*   positions are calculated first, followed by accelerations (from
*   accel()), and velocities.
*/
{
    register i;
    register long float *pr,*prl,*pv,*pa,*pal;

    ktm+=2;

    prl=rlxyz;
    pr=rxyz;
    pv=vxyz;
    pal=alxyz;
    pa=axyz;

    for (i=0; i<natom3 ;i++) {
         *prl = *pr + h*(*pv) + h26*(4.*(*pa) - *pal);
         *pal = 0;
         pal++;
         pa++;
         pv++;
         pr++;
         prl++;
    }

    iflag=1;
    accel(alxyz,rlxyz);

    prl=rlxyz;
    pr=rxyz;
    pv=vxyz;
    pal=alxyz;
    pa=axyz;

    for (i=0; i<natom3 ;i++) {
         *pv = (*prl - (*pr) + h26*(2.*(*pal) + (*pa)))*hi;
         *pr = *prl + h*(*pv) + h26*(4.*(*pal) - (*pa));
         *pa = 0.;
         pal++;
         pa++;
         pv++;
         pr++;
         prl++;
    }

    iflag=0;
```

```
    accel(axyz,rxyz);

    prl=rlxyz;
    pr=rxyz;
    pv=vxyz;
    pal=alxyz;
    pa=axyz;

    for (i=0; i<natom3 ;i++) {
         *pv = (*pr - (*prl) + h26*(2.*(*pa) + (*pal)))*hi;
         pal++;
         pa++;
         pv++;
         pr++;
         prl++;
    }

}
/*****************************************/
accel(pa,pr)
/*****************************************/
long float *pa,*pr;
/*
*   Acceleration subroutine, identical to that in "I2XENON".
*   Calculated the accelerations of every particle in the system due to
*   interactions within the system, including the I-I and I-Xe
*   interactions.
*/
{
    register i,j;
    long float temp;
    register long float *paj,*pai;
    register long float *prxt,*pryt,*przt,*pr2t;
    int n;

    sort();

    prxt=rxt;
    pryt=ryt;
    przt=rzt;
    pr2t=r2t;

    n=(int)(4750./(*pr2t)) - 1;
    *pr2t = _lsqrt(*pr2t);
    if(iflagl==0){
    *pr2t = ftab[n]/(*pr2t) + ftabi[n];
    }
    else{
    *pr2t = plud/pow((float)*pr2t,11.5);
    }
    *prxt *= *pr2t;
    *pryt *= *pr2t;
```

```
*przt *= *pr2t;

pr2t++;
prxt++;
pryt++;
przt++;

for(i=0; i<2; i++){
for(j=2; j<natom; j++){
  if(*pr2t <= r2cut){
  temp = 1./((*pr2t)*(*pr2t)*(*pr2t)*(*pr2t));
  *pr2t = temp*(cli2d*(*pr2t)*temp - c2i2d);
  *prxt *= (*pr2t);
  *pryt *= (*pr2t);
  *przt *= (*pr2t);
  }
  else{
  *prxt = 0;
  *pryt = 0;
  *przt = 0;
  }
pr2t++;
prxt++;
pryt++;
przt++;
}
}

for (i=2; i<natoml; i++){
     for(j=i+1; j<natom; j++){
     if(*pr2t<=r2cut){
     temp = 1./((*pr2t)*(*pr2t)*(*pr2t)*(*pr2t));
     *pr2t = temp*(cld*(*pr2t)*temp-c2d);
     *prxt *= (*pr2t);
     *pryt *= (*pr2t);
     *przt *= (*pr2t);
     }
     else{
     *prxt = 0;
     *pryt = 0;
     *przt = 0;
     }
     prxt++;
     pryt++;
     przt++;
     pr2t++;
     }
}

prxt=rxt;
pryt=ryt;
przt=rzt;
```

```
    pr2t=r2t;
    pai=pa;
    for (i=0; i<natoml; i++){
    paj=pai+3;
         for(j=i+1; j<natom; j++){
         *pai += (*prxt);
         *paj -= (*prxt);
         pai++;
         paj++;
         *pai += (*pryt);
         *paj -= (*pryt);
         pai++;
         paj++;
         *pai += (*przt);
         *paj -= (*przt);
         pai -= 2;
         paj++;
         prxt++;
         pryt++;
         przt++;
         }
    pai += 3;
    }

    pai = pa;
    for (i=0; i<6; i++){
         *pai /= smassi;
         pai++;
    }
    for (i=6; i<natom3; i++){
         *pai /= smass;
         pai++;
    }

}
/*****************************************/
energy()
/*****************************************/
/*
*   In this subroutine, the kinetic, potential, and total energies of
*   the 108 atom system are calculated, along with the iodine
*   vibrational energy.  Once again, calculation proceeds as in
*   "I2XENON".
*/
{
    register i,j;
    int npart,nparti,n;
    long float temp,rb,vb,tot,r21;
    register long float *pv,*pr2,*pk,*pp;

    sort();
```

```
pv = vxyz;
pk = ekatom;
for(i=0; i<2; i++){
  *pk = (*pv)*(*pv);
  pv++;
  *pk += (*pv)*(*pv);
  pv++;
  *pk += (*pv)*(*pv);
  pv++;
  (*pk) *= smassi/2.;
  pk++;
}
for(i=2; i<natom; i++){
  *pk = (*pv)*(*pv);
  pv++;
  *pk += (*pv)*(*pv);
  pv++;
  *pk += (*pv)*(*pv);
  pv++;
  (*pk) *= smass/2.;
  pk++;
}

ek = 0.;
pk = ekatom;
for(i=0; i<natom; i++)
ek += (*pk++);

eki2 = 0.;
for(i=0; i<3; i++){
rb = rxyz[i]-rxyz[i+3];
vb = vxyz[i]-vxyz[i+3];
eki2 += rb*vb/_lsqrt(r2t[0]);
}
eki2 *= eki2*.25*smassi;

pp = epatom;
for(i=0; i<natom; i++)
(*pp++) = 0.;

pr2 = r2t;
n=(int)(4750./(*pr2)) - 1;
r21 = _lsqrt(*pr2);
ep = vtab[n]+r21*(vtabi[n]);
pr2++;
if(iflag1=1){
epatom[0] = plu/(pow((float)r21,9.5));
epatom[1] = epatom[0];
}

ei2 = ep + eki2;
```

```
    npart = 0;
    nparti = 0;
    for(j=0; j<2; j++){
    for(i=2; i<natom; i++){
      if(*pr2<=r2cut){
      nparti++;
      temp = 1./((*pr2)*(*pr2)*(*pr2));
      temp *= c1i2*temp-c2i2;
      ep += temp;
      epatom[j] += (temp-vzix)/2.;
      epatom[i] += (temp-vzix)/2.;
      }
    pr2++;
    }
    }

    for(j=2; j<natoml; j++){
    for(i=j+1; i<natom; i++){
      if(*pr2<=r2cut){
      npart++;
      temp = 1./((*pr2)*(*pr2)*(*pr2));
      temp *= c1*temp-c2;
      ep += temp;
      epatom[j] += (temp-vzero)/2.;
      epatom[i] += (temp-vzero)/2.;
      }
    pr2++;
    }
    }

    ep -= (npart*vzero + nparti*vzix);
    et = ek + ep;
    fprintf(fenergy,"%lg %lg %lg %lg %lg
         \n",ek,ep,et,ei2,_lsqrt(r2t[0]));

}
/*****************************************/
sort()
/*****************************************/
/*
*   Sorting subroutine, similar to ssort() of "I2XENON".  No
*   preliminary sort, as in sort() of "I2XENON" is necessary for this
*   small system.
*/
{
    register i,j;

    register long float *pri1,*pri2,*pri3,*prj1,*prj2,*prj3;
    long float *prxt,*pryt,*przt,*pr2t;
    long float xdif,ydif,zdif;
    int ix,iy,iz;
```

```
    if (iflag == 0){
    pri1=rxyz;
    pri2=(&rxyz[1]);
    pri3=(&rxyz[2]);
    }
    else{
    pri1=r1xyz;
    pri2=(&r1xyz[1]);
    pri3=(&r1xyz[2]);
    }

    prxt=rxt;
    pryt=ryt;
    przt=rzt;
    pr2t=r2t;

    for(i=0; i<natom1; i++){
    prj1=pri1+3;
    prj2=pri2+3;
    prj3=pri3+3;
      for(j=i+1; j<natom; j++){
      xdif=(*pri1-*prj1)/alxyz;
      ydif=(*pri2-*prj2)/alxyz;
      zdif=(*pri3-*prj3)/alxyz;
      ix = (xdif>0 ? (int)(xdif+.5) : (int)(xdif-.5));
      iy = (ydif>0 ? (int)(ydif+.5) : (int)(ydif-.5));
      iz = (zdif>0 ? (int)(zdif+.5) : (int)(zdif-.5));
      *prxt=(*pri1)-(*prj1)-ix*alxyz;
      *pryt=(*pri2)-(*prj2)-iy*alxyz;
      *przt=(*pri3)-(*prj3)-iz*alxyz;
      *pr2t=(*prxt)*(*prxt)+(*pryt)*(*pryt)+(*przt)*(*przt);
      pr2t++;
      prxt++;
      pryt++;
      przt++;
      prj1 += 3;
      prj2 += 3;
      prj3 += 3;
      }
    pri1 += 3;
    pri2 += 3;
    pri3 += 3;
    }

}
/*******************************************/
 potchk()
/*******************************************/
/*
*   This is the subroutine to switch potential surfaces from the
*   dissociative surface to the ground state surface.  Energy is
```

```
*   conserved by adding the difference in energy between the two states
*   at the hopping point to the kenetic energy of the iodine atoms.
*/
{

    register i;
    int n;
    long float r22;
    long float rb[3],r21;
    long float p1,p2,vsw,v21,v22;

    r22 = 0.;

    for(i=0; i<3; i++){
    rb[i] = rxyz[i]-rxyz[i+3];
    r22 += rb[i]*rb[i];
    }

    r21 = _lsqrt(r22);

    for(i=0; i<3; i++)
    rb[i] /= r21;

    p2 = p1u/(pow((float)r21,9.5));
    n=(int)(4750./r22)-1;
    p1=vtab[n]+r21*vtabi[n];

    vsw = (p2-p1)/smassi;

    v21 = 0.;

    for(i=0; i<3; i++)
    v21 += rb[i]*(vxyz[i]-vxyz[i+3]);

    v22 = _lsqrt(v21*v21 + 4.*vsw);

    if(v21>=0)
    v21 = .5*(v22-v21);
    else
    v21 = .5*(-v22-v21);

    for(i=0; i<3; i++){
    vxyz[i] += rb[i]*v21;
    vxyz[i+3] -= rb[i]*v21;
    }

    iflag=0;

}
/*****************************************/
bondc()
/*****************************************/
```

```
/*
*   This subroutine equilibrates the iodine vibrational energy.  The
*   bond length is set to the equilibrium bond length, and the
*   velocities of the particles are randomly assigned from a Boltzmann
*   distribution.  It is identical to the subroutine bondc() in
*   "I2XENON".
*/
{
    register i;
    long float r2,c;
    long float rb[3],rc[3];
    float gauss();
    float eps[6];

    r2=0.;
    for(i=0; i<3; i++){
    rc[i] = (rxyz[i]+rxyz[i+3])/2.;
    rb[i] = (rxyz[i]-rxyz[i+3]);
    r2 += rb[i]*rb[i];
    }

    c = 2.6657/(_lsqrt(r2)*2.);

    for(i=0; i<3; i++){
    rxyz[i] = rc[i] + c*rb[i];
    rxyz[i+3] = rc[i] - c*rb[i];
    }

    gauss(eps,6);
    for(i=0; i<6; i++)
    vxyz[i] = eps[i]*sktm;

}
/*****************************************/
tempc()
/*****************************************/
/*
*   This subroutine equilibrates the temperature of the system, as in
*   "I2XENON".
*/
{

    register i;
    long float r;

    r = _lsqrt(akt/ek);
    if(r<.9) r=.978;

    for(i=0; i<natom3; i++)
    vxyz[i] *= r;

}
```

```
/*****************************************/
diss()
/*****************************************/
/*
*   Subroutine to photoexcite the iodine molecule.  17,000 wave numbers
*   of energy are added to the system as the molecule is placed on the
*   excited state dissociative surface.  Energy in excess of the
*   difference in energy between the two states at the excitation
*   radius is added to the vibrational kinetic energy of the diatomic
*   in the same direction as the vibrational motion just prior to
*   excitation.  See "I2XENON" for details.
*/
{
    register i;
    long float ed,r2,r21,rb[3];
    long float esw;
    long float p1,p2;
    long float v21,v22;
    int n;

    ed = 17000.*1.1961;
    iflag1 = 1;
    keflag = 1;

    r2 = 0.;
    for(i=0; i<3; i++){
    rb[i] = rxyz[i]-rxyz[i+3];
    r2 += rb[i]*rb[i];
    }

    r21 = _lsqrt(r2);
    for(i=0; i<3; i++)
    rb[i] /= r21;

    p2 = plu/(pow((float)r21,9.5));
    n = (int)(4750./r2)-1;
    p1 = vtab[n]+r21*vtabi[n];

    esw = (ed + p1 - p2)/smassi;
    if(esw < 0.) esw = 0.;

    v21 = 0.;
    for(i=0; i<3; i++)
    v21 += rb[i]*(vxyz[i]-vxyz[i+3]);

    v22 = _lsqrt(v21*v21 + 4.*esw);
    if(v21<=0)
    v21 = .5*(-v22-v21);
    else
    v21 = .5*(v22-v21);

    for(i=0; i<3; i++){
```

```
    vxyz[i] += rb[i]*v21;
    vxyz[i+3] -= rb[i]*v21;
    }

}
```

LAWRENCE BERKELEY LABORATORY
TECHNICAL INFORMATION DEPARTMENT
1 CYCLOTRON ROAD
BERKELEY, CALIFORNIA 94720