LBNL-42856

Study of Pair and Many-Body Interactions in Rare-Gas Halide Atom Clusters Using Negative Ion Zero Electron Kinetic Energy (ZEKE) and Threshold Photodetachment Spectroscopy

Ivan Yourshaw
Ph.D. Thesis

Department of Chemistry
University of California, Berkeley

and

Chemical Sciences Division
Ernest Orlando Lawrence Berkeley National Laboratory
University of California
Berkeley, CA 94720

July 1998

This work was supported by the Director, Office of Science, Office of Basic Energy Sciences, Chemical Sciences Division, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098, and the National Science Foundation.

DISCLAIMER

Study of Pair and Many-Body Interactions in Rare-Gas Halide Atom Clusters Using Negative Ion Zero Electron Kinetic Energy (ZEKE) and Threshold Photodetachment Spectroscopy

by

Ivan Yourshaw

B.S. (University of Iowa) 1991

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy
in

Chemistry

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Daniel M. Neumark
Professor Richard J. Saykally
Professor Arup K. Chakraborty

Fall 1998

Abstract

Study of Pair and Many-Body Interactions in Rare-Gas Halide Atom Clusters Using Negative Ion Zero Electron Kinetic Energy (ZEKE) and Threshold Photodetachment Spectroscopy

by

Ivan Yourshaw

Doctor of Philosophy in Chemistry

University of California, Berkeley

Professor Daniel M. Neumark, Chair

The diatomic halogen atom-rare gas diatomic complexes $KrBr^-$, $XeBr^-$, and $KrCl^-$ are studied in this work by zero electron kinetic energy (ZEKE) spectroscopy in order to characterize the weak intermolecular diatomic potentials of these species. Also, the ZEKE and threshold photodetachment spectra of the polyatomic clusters Ar_nBr^- (n = 2-9) and Ar_nI^- (n = 2-19) are studied to obtain information about the non-additive effects on the interactions among the atoms. This work is part of an ongoing effort to characterize the pair and many-body potentials of the complete series of rare gas halide clusters. In these studies we obtain information about both the anionic and neutral clusters.

In the spectra of the diatomic complex we observe well-resolved vibrational structure which is used in conjunction with scattering results from the literature to

construct accurate intermolecular potentials. We also obtain accurate electron affinities for these species.

From the spectra of the polyatomic species we obtain accurate electron affinities and electronic state term values. We observe some partially resolved vibrational structure for Ar_nBr^- (n = 2-3) and Ar_nI^- (n = 2-3). The global minimum energy structures of the clusters are found using a molecular-dynamics based simulated annealing algorithm. The electron affinities calculated using these structures and the pair potentials obtained from previous ZEKE studies of the $ArBr^-$ and ArI^- complexes are found to be inconsistent with the experimentally observed electron affinities, indicating the importance of non-additive effects in these clusters. Various non-additive interactions are considered, and the most important are found to be non-additive induction effects and the effect arising from the interaction of the halide charge with the multipole moments due to the distortion of the rare-gas electron clouds. For the neutral clusters Ar_nBr (n = 2-3) and Ar_nI (n =2-7) we also observe many-body effects in the electronic structure due to the presence of the open-shell halogen atom. These effects are successfully modeled using a simple degenerate perturbation theory treatment of the open shell-closed shell interaction.

Dedicated to my parents

Michael Yourshaw

and

Sarah Mulholland

Table of Contents

Acknowledgments

I would first of all like to thank my advisor Dan Neumark for hiring me, and for being a constant source of creative inspiration. I have learned a lot about science and about life from his mentorship.

The other members of the Neumark group all also contributed to this work in one way or another. Primary thanks go to the past and present students and post-docs who have worked with me in the "second lab," as the ZEKE lab is affectionately known. Yuexing Zhao taught me almost everything I needed to know about running the experiment, and his amiable good nature made him a pleasure to work with. Caroline Arnold and Georg Reiser also provided me with valuable insights into the mysteries and rituals of anion ZEKE spectroscopy. Esther DeBeer contributed greatly to the development of the Raman experiment, not covered in this thesis, and patiently dealt with the trials and tribulations of burnt YAG rods and fused silica windows. Thomas Lenzer "the radioactive post-doc" joined the group recently, and his quick mind and easy-going nature have made him a pleasure to work with and valuable addition to the ZEKE team. Thomas has also patiently and uncritically endured my attempts to practice my very rudimentary German on him. Mike Furlanetto has contributed a great deal to the ZEKE experiment recently, and his organizational skills have been especially appreciated. He and the newest member of the ZEKE team Nick Provinka are currently engaged in an overhaul and transfiguration of the ZEKE machine into a Raman photoelectron spectrometer, a task which they have undertaken with enthusiasm and skill. I would wish them good luck, but I don't believe luck will be necessary with their talent.

I would also like to acknowledge the friendship and support of several other group members. Former group members Dave Osborn and Don Arnold deserve special mention because it was through talking with them and Dan that I was first persuaded to join the group, and I continued to look to them for advice and as valuable role models during my stay at Berkeley. Knut Asmis, "the yodeling post-doc" has been of great help in tutoring me in some of the finer points of the German language. Two group members, Travis Taylor and Jeff Greenblatt have been special friends in times of distress. Travis has been inspiration by example in how to stay true to oneself in an often chaotic environment. Jeff has also been a special friend, and I have benefited a lot from our discussions of what it means to be a scientist at the end of the 20th Century. Special thanks also to my office mates Ryan Bise and Hyeon Choi for their companionship during the final phase of my thesis writing, and for not complaining about my insistence that the office door be kept locked.

I also wish to thank my family. My parents Michael Yourshaw and Sarah Mulholland, to whom this thesis is dedicated, instilled in me from an early age a love of learning. For their emotional and financial support throughout my educational career I will always be grateful. Special thanks are also due to my step-father Dan Mulholland for all the email, and the support of my step-mother Kathy Yourshaw.

These acknowledgments would not me complete without thanking the fine professional staff at Alta Bates Herrick Hospital and my fellow patients there, for restoring me to sanity and literally saving my life.

Finally I would like to thank Gilda Mark for her love and unfailing faith in me even at the times when I had lost faith in myself.

Chapter 1. Introduction

The spectroscopic study of weakly bound van der Waals clusters has experienced an explosive period of growth in the past two decades, prompted mainly by innovations in laser and molecular beam technology. At the same time, a tremendous amount of complementary theoretical work has been done on these systems. This work is still underway, and it is probably not an understatement to describe the current era as a "golden age" of cluster research. In this introductory chapter, we will summarize the reasons for this outpouring of interest in this field, briefly review some highlights of what has been learned so far, and review the results previous work in the Neumark group in this field. In particular we will focus on recent advances in the understanding of "many-body" interactions, i.e., interactions involving synergistic effects among three or more atoms or molecules.

Here, and in the rest of this work, we will restrict our attention to weakly bound clusters interacting by dispersion or induction forces, and leave aside discussion of covalently bound or "metallic" clusters, a field of study which has also experienced a massive amount of attention during this same time period, but is beyond the scope of the present work.

1.1 Intermolecular forces: importance and brief historical review

> "Since the end of the nineteenth century a considerable amount of work has been devoted to the exact formulation of the connection between the properties of matter in bulk and intermolecular forces. Such a formulation represents the ultimate aim of the molecular theory of matter since, when a theory of this kind is established, a knowledge of the intermolecular forces is sufficient for

the evaluation of all the properties of the bulk materials." -G. C. Maitland, M. Rigby, E. B. Smith, and W. A. Wakeham in *Intermolecular Forces*, p. 3.[1]

Interest in the weak interactions of atoms and molecules actually preceded these recent spectroscopic and theoretical advances by many years. The importance of the concept of intermolecular interactions with both attractive and repulsive components, and their connection with the bulk properties of gases was first made clear by van der Waals in 1873 with his formulation of the van der Waals equation of state. (See Ref.1, p. 2.) Subsequently, the term "van der Waals forces" was adopted as a blanket phrase for all of these weak interactions. An understanding of van der Waals forces is necessary to understand many bulk properties of matter, such of heats of sublimation and evaporation,[2] and solvation. The existence of the inductive dipole-induced dipole and dipole-dipole interactions were first proposed to explain these bulk properties by Debye (1920), and Keesom (1921). However it was not until the work of London in 1930 that dispersion forces were proposed to explain the bulk properties of atoms which do not posses multipole moments, such as rare gas atoms. For reviews of this early work, see the articles by Margenau[3] and London.[4] For a very basic and readable introduction to van der Waals forces, see Chapter 13 of Kauzmann[2] as well as Chapter 1 of Maitland *et. al.*[1]

With the advent of the full development of quantum mechanics, the conceptual framework to understand van der Waals forces was established. However, because of the practical intractability of exact quantum mechanical solution of even the simplest problem involving van der Waals interactions, it was necessary to take a more empirical approach to the problem. And thus was born what Maitland *et al.* refer to as the

"Lennard-Jones era" in the study of intermolecular forces. (See Chapter 9 of Ref. 1.) Under this paradigm, the Born-Oppenheimer approximation was assumed to separate electronic from nuclear motions and simple analytic forms for the internuclear potential--such as the familiar Lennard-Jones 12-6 potential--were used, with the Coulomb interactions of the electrons implicitly included in this potential function. Also, pairwise additivity of the potentials was usually assumed. The methodology under this paradigm was then to attempt to reconcile these model potentials with the experimentally observed properties of the bulk substances. For a comprehensive treatment of these early efforts to understand the relation between intermolecular forces and bulk properties see Hirschfelder, *et. al., Molecular Theory of Gases and Liquids.*[5]

During this time some theoretical work was done to advance knowledge about the non-pairwise interactions among atoms and molecules which are essential to understand if one is to bridge the gap between molecular and bulk properties. In 1943 Axilrod and Teller[6] proposed a three-body dispersion force, which was long accepted, with little *direct* experimental confirmation, as the primary many-body effect in bulk substances. In the early 1960s Jansen[7] proposed a model of many-body interactions based on exchange effects, which seemed to account for discrepancies between pairwise additive potential predictions and experimental measurements of the binding energies of alkali-halide crystals. For a review and critique of this early work on many-body forces see Meath and Aziz.[8] For a recent, comprehensive review of experimental and theoretical research in this field see Elrod and Saykally.[9]

1.2. Early spectroscopic work on clusters

It became apparent that in order to gain a truly firm understanding of van der Waals interactions one must begin by studying the potentials of small clusters, beginning with diatomic complexes, in order to avoid the confounding factors involved with extrapolating from the properties of bulk materials to fundamental intermolecular forces. Such work began in earnest in the late sixties and early seventies. The earliest work in this field did not involve the use of molecular beams to produce clusters, but relied on spectroscopic probes of the high vibrational levels of molecules such as Mg_2 to determine the (Rydberg-Klein-Rees) RKR potential at long range in order to characterize the van der Waals interactions,[10] and vacuum ultraviolet absorption spectroscopy of dilute rare gases with very long absorption path lengths, such as the work of Tanaka *et al.* on the rare-gas dimers such as Ar_2.[11] Klemperer and co-workers pioneered the use of supersonic expansions to form molecular beams of clusters in conjunctions with electric resonance spectroscopy, to study van der Waals systems with strong dipole moments, such as Ar-HCl.[12] For a review of this early spectroscopic work see Chapter 7 of Maitland *et al.*[1]

The field of cluster research began its period of exponential growth in the late seventies with the landmark LIF studies by Levy and co-workers on the I_2-Rg (Rg = rare gas) complexes.[13] Since then the field of cluster research has expanded to such an extent that it is not possible to begin a summary here. For a comprehensive treatment of recent cluster research we refer the reader to the review by Castleman and Bowen.[14]

1.3. ZEKE studies of simple van der Waals complexes

The focus of the present work in on anion ZEKE and threshold photodetachment spectroscopy of halide and halogen atoms with rare gas atoms. This work is of fundamental importance because we are able to study a very simple example of the interaction of an open-shell species with a rare gas atom. We also gain important information about the anionic interaction. In this section we review the basic nature of the interaction of an open-shell (^{2}P) halogen atom with a single rare gas atom or closed shell molecule, and then review the results of previous anion ZEKE studies of these systems.

1.3.1. The interaction of an open-shell (^{2}P) halogen atom with a closed-shell species.

In this section we describe the electronic states that arise when an open-shell ($^2P_{1/2,3/2}$) halogen atom interacts with a ^{1}S rare gas atom. First we consider the case when the spin orbit coupling of the halogen is negligible. In this case we have two electronic states, pictured schematically in Figure 1.1. When the p-orbital containing the "hole" is aligned along the internuclear axis, we have a $^2\Sigma$ state (Λ=0), and when this p-orbital is oriented perpendicular to the internuclear axis we have two $^2\Pi$ states (Λ=1).

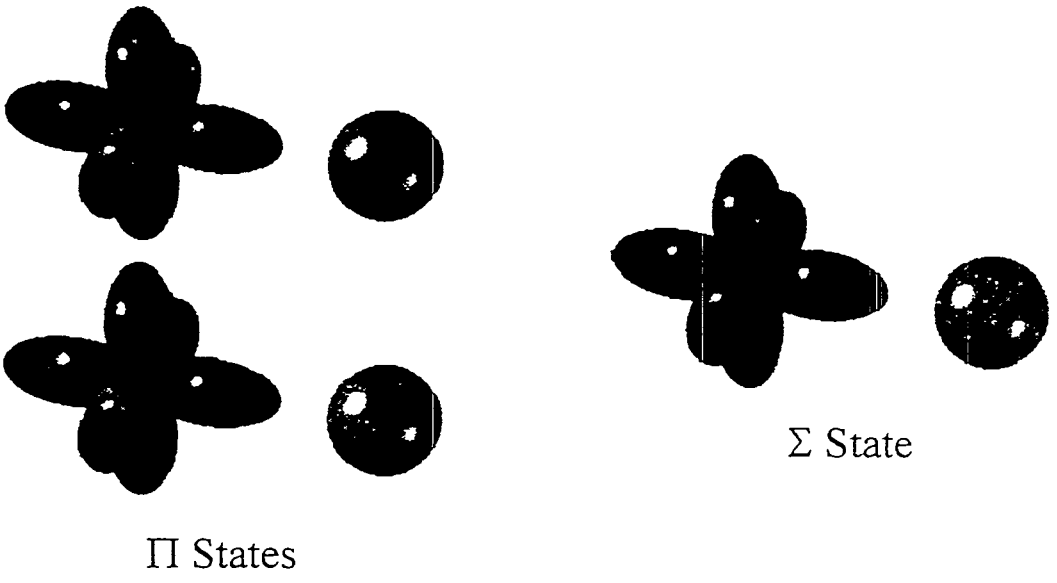

Figure 1.1. The electronic states arising from the interaction of a ^{2}P halogen atom and a ^{1}S rare-gas neglecting spin-orbit effects.

When spin orbit coupling is included and is small relative to the electrostatic interaction Hund's case (a) prevails, and the two states are split into the $^2\Pi_{1/2}$ and $^2\Pi_{3/2}$ states, which in the case of $^2s^5p$ atoms like the halogens would lie higher in energy than the ground $^2\Sigma_{1/2}$ state. This situation is shown on the left-hand side of Figure 1.2 on the following page. As the spin-orbit interaction increases--or as the internuclear nuclear increases so that the electrostatic interaction is relatively smaller--the $^2\Pi_{3/2}$ and $^2\Sigma_{1/2}$ states mix, and the complex is then described by Hund's coupling case (c) as Λ ceases to be a good quantum number. This is shown on the right-hand side of Figure 1.2. The $^2\Pi_{1/2}$ and $^2\Sigma_{1/2}$ states have mixed to form the lower $X\,\frac{1}{2}(j=\frac{3}{2},\Omega=\frac{1}{2})$ state and the upper $II\,\frac{1}{2}(j=\frac{1}{2},\Omega=\frac{1}{2})$ state, separated by an amount similar to the atomic spin orbit splitting. The $^2\Pi_{3/2}$ state remains as a pure Π state, called the $I\,\frac{1}{2}(j=\frac{3}{2},\Omega=\frac{3}{2})$ state in case (c)

notation, higher in energy than the $X\,\frac{1}{2}$ state by an amount related to the electrostatic interaction between the atoms.

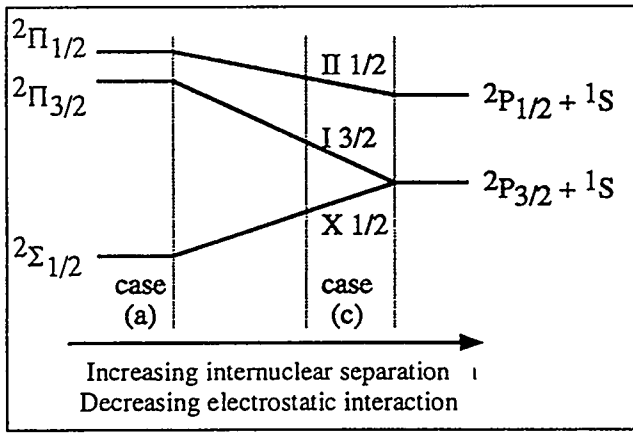

Figure 1.2. Correlation diagram of the coupling cases arising from the interaction of a ^2P halogen atom and a closed shell species. Adapted from H. Haberland, *Z. Physik A*, **307** 35 (1982).

The Hund's case (c) limit applies to the well and long-range region of most of the rare-gas halogens. The amount of Σ–Π mixing in the three electronic states is shown in Figure 1.3 as a function of internuclear separation.

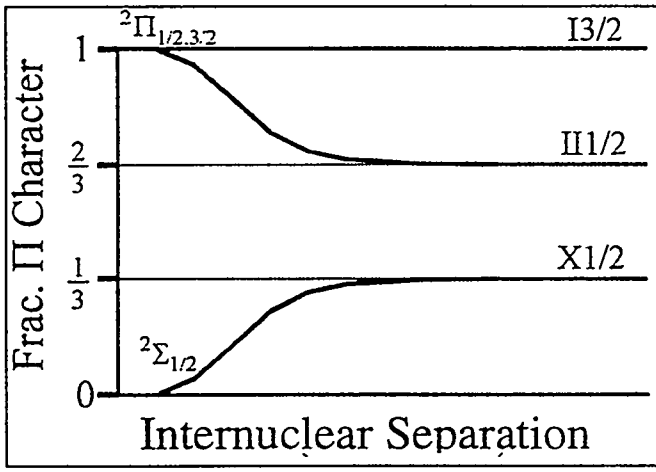

Figure 1.3. Diagram showing the relative Π contribution in the interaction between a ^2P halogen atom and a ^1S closed-shell species as a function of internuclear separation.
Adapted from V. Hoffmann and H. Morgner, *J. Phys. B* **12**, 2857 (1979).

Again the situation on the right-hand side of this figure [case (c)] is the most pertinent to the complexes studied in this work, where the $X\,\frac{1}{2}$ state has $\frac{2}{3}\,\Sigma$ character and $\frac{1}{3}$ Π character, and the $II\,\frac{1}{2}$ state has $\frac{1}{3}$ Σ and $\frac{2}{3}$ Π character. As already mentioned the $I\,\frac{3}{2}$ state has pure Π character.

It should also be mentioned here that all three of the neutral electronic states can in principle be observed by ZEKE spectroscopy, since the transitions involve removal of a single electron from a p-orbital of the halide anion. A detailed derivation of the rovibronic transition strengths for the anion → neutral transition can be found in Appendix B. For more detailed information about the interaction between open-shell and closed shell atoms see Refs. 15 and 16.

1.3.2. Review of previous anion ZEKE studies of halogen atom-closed shell systems

In this section we review previous applications of anion ZEKE spectroscopy to study the complexes of halide/halogen atoms with closed shell species. The first such study was the ZEKE spectrum observed in the Neumark group of the I^--CO_2 complex.[17] In this work it was found that the complex is "T"-shaped, with the CO_2 slightly bent. However, the interaction between the I atom and the CO_2 molecule is sufficiently weak (44.5 meV for the $X\,\frac{1}{2}$ state) that the interaction resembles a van der Waals interaction much more than a covalent bond (in contrast to FCO_2, which is a covalently bound molecule.[18]) Low frequency vibrational progressions were observed in the van der Waals stretching mode for each electronic state, and the spectra were fit to model one-dimensional anion and neutral potentials. The binding energy of the anionic complex was found to be 212.0 meV. Structure was also observed corresponding to the CO_2 internal bending vibration. Also in this laboratory, the I^--CH_3I complex was studied by ZEKE spectroscopy.[19]

Schlag and co-workers have studied the I^--H_2O complex by ZEKE spectroscopy.[20] They observed the three electronic states discussed above, as well as two

low frequency vibrational progressions. However, due to the uncertainty in the cluster geometry and the possibility of observable rotational structure, they were not able to conclusively assign the vibrational features or fit the spectrum to a model potential.

The work of most direct relevance here was our previous anion ZEKE study of the ArI^-, $ArBr^-$, and KrI^- complexes.[21] Because of the simple nature of the interaction compared to the studies involving molecular solvents, we were able to observe well resolved vibrational progressions in all three of the electronic states and fit the spectra to model potentials, obtaining accurate vibrational frequencies, well depths and bond lengths for the anion as well as neutral states. The potential parameters for the ArI and ArBr systems can be found in Chapter 4 (Table 4.3). The work described in Chapter 3 is a continuation of this study.

1.4. Ar_nX^- and Ar_nX clusters: many-body interactions

As mentioned above, the study of polyatomic clusters is ideal to directly observe the many-body effects which are of importance in bulk phenomena, such as solvation in liquids and the binding of crystals. In Chapter 4 we will delve in great depth into the nature of the many-body interactions among a halide neutral or anion and a number of rare gas atoms. In this introductory section, therefore, we will give a brief, more intuitive overview of the nature of the most important of these interactions.

In the case of the neutral Rg_nX clusters, the non-additivity of the potentials arises because of the open-shell nature of the halogen atom. Consider again the spinless states shown in Figure 1.1. The axis of symmetry is defined by the two nuclei. Because the electronic structure of the halogen atom is not spherically symmetric, the potential

functions differ according to the orientation of the singly occupied p-orbital. Now consider the situation when a second rare-gas atom is brought near the Rg-X pair. The symmetry of the diatomic complex is broken, and the p-orbital must reorient to accommodate the presence of the second Rg atom. Considering this, it seems rather unlikely that the potentials would add in a simple pairwise fashion. In this sense, that one must consider the orientation of the orbitals involved, this type of open shell-closed shell interaction may be considered something of an intermediate case between a "pure" van der Waals interaction and a covalent bond. This is only meant to be an intuitive discussion of the topic; for all the "gory details" of how this calculation is accomplished, we refer the reader to Chapter 4.

The many-body forces at work in Rg_nX^- anionic clusters are of fundamental interest as models of the forces involved in solvation of ions. The two most important non-additive interactions in these systems are the induction and "exchange charge" forces.

The induction non-additivity is the dominant many body effect in the Rg_nX^- clusters, and is quite easy to understand. In Figure 1.4 we show the induction interaction between a halide anion and two rare gas atoms. The halide induces dipole moments in both rare gas atoms. If the rare gases are next to each other, as shown, the induced dipoles repel, reducing the net binding energy of the cluster from what it would be without this effect.

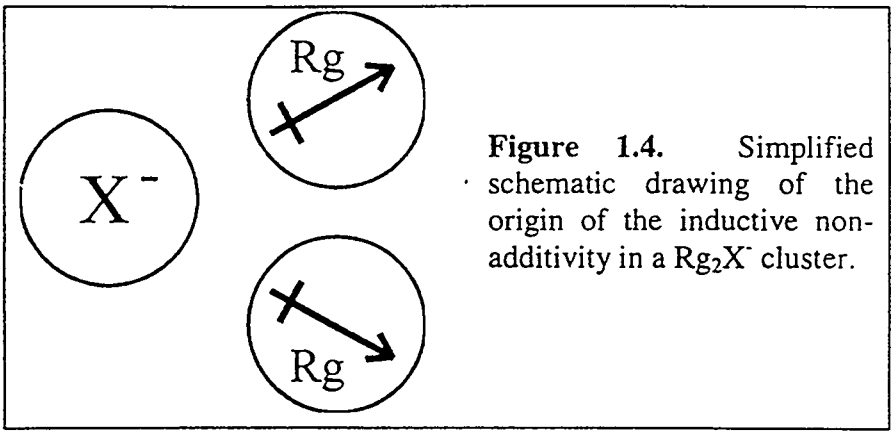

Figure 1.4. Simplified schematic drawing of the origin of the inductive non-additivity in a Rg_2X^- cluster.

Again, we will wait until Chapter 4 to discuss a more complete model of the induction non-additivity.

The other type of non-additive interaction we have found to be important arises from the distortion of the electron clouds of two Rg atoms when they are brought close together with a nearby halide anion. This effect is pictured in Figure 1.5. The exchange repulsion between the Rg electron clouds causes a net positive charge, δ^+, to appear at the midpoint along the Rg-Rg axis, and partial negative charges, each half the magnitude of δ^+, to appear at the Rg nuclei. Because the positive charge is closer to the halide than the negative charges, the net effect is an attractive force, and an increase in the binding energy over what it would be without this effect.

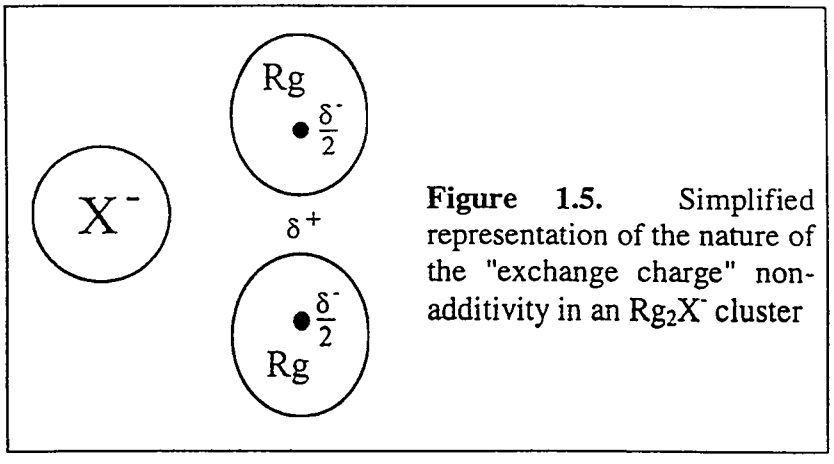

Figure 1.5. Simplified representation of the nature of the "exchange charge" non-additivity in an Rg_2X^- cluster

We have found, from the experiments and calculations described in Chapter 4, that this "exchange-charge" effect makes a quite significant contribution to the non-additive energy of the cluster anions, being about half as large as the induction non-additivity with opposite sign.

1.5. References

[1] G. C. Maitland, M. Rigby, E. B. Smith, and W. A. Wakeham, *Intermolecular Forces: Their Origin and Determination* (Oxford University Press, Oxford, 1981).

[2] W. Kauzmann, *Quantum Chemistry: An Introduction* (Academic Press Inc., New York, 1957).

[3] H. Margenau, Revs. Mod. Pys. **11**, 1 (1939).

[4] F. London, Trans. Faraday Soc. **33**, 8 (1937).

[5] J. O. Hirschfelder, C. F. Curtiss, and R. B. Bird, *Molecular Theory of Gases and Liquids* (Wiley, New York, 1954).

[6] B. M. Axilrod and E. Teller, J. Chem. Phys. **11**, 299 (1943).

[7] L. Jansen, Adv. Quantum Chem. **2**, 119 (1965).

[8] W. J. Meath and R. A. Aziz, Mol. Phys. **52**, 225 (1984).

[9] M. J. Elrod and R. J. Saykally, Chem. Rev. **94**, 1975 (1994).

[10] W. C. Stwalley, Chem. Phys. Lett. **7**, 600 (1970).

[11] Y. Tanaka and K. Yoshino, J. Chem. Phys. **53**, 2012 (1970).

[12] S. E. Novick, K. C. Janda, S. L. Holmgren, M. Waldman, and W. Klemperer, J. Chem. Phys. **65**, 1114 (1976).

[13] M. S. Kim, R. E. Smalley, L. Wharton, and D. H. Levy, J. Chem. Phys. **65**, 1216 (1976).

[14] A. W. Castleman and K. H. Bowen, J. Phys. Chem. **100**, 12911 (1996).

[15] V. Aquilanti, G. Liuti, F. Pirani, and F. Vecchiocattivi, J. Chem. Soc. Faraday Trans 2 85, 955 (1989).

[16] V. Aquilanti and G. Grossi, J. Chem. Phys. **73**, 1165 (1980).

[17] Y. Zhao, C. C. Arnold, and D. M. Neumark, J. Chem. Soc. Faraday Trans. **89**, 1449 (1993).

[18] D. W. Arnold, S. E. Bradforth, E. H. Kim, and D. M. Neumark, J. Chem. Phys. **102**, 3493 (1995).

[19] C. C. Arnold, Ph.D. Thesis, University of California, 1994.

[20] C. Bassmann, U. Boesl, D. Yang, G. Drechsler, and E. W. Schlag, International Journal of Mass Spectrometry and Ion Processes **159**, 153 (1996).

[21] Y. Zhao, I. Yourshaw, G. Reiser, C. C. Arnold, and D. M. Neumark, J. Of Chem. Phys. **101**, 6538 (1994).

Chapter 2. Experimental techniques and apparatus

In this chapter, we describe the experimental techniques used to study Rg_nX^- clusters, with an emphasis on recent improvements to the experimental apparatus. First we briefly review the basic concepts of anion photoelectron spectroscopy (PES), then describe the closely related techniques of threshold photodetachment and zero electron kinetic energy (ZEKE) spectroscopy used in these studies. Finally we describe in some detail recent improvements to the experimental apparatus which have enabled us to increase production of large van der Waals clusters, as well as improve the signal-to-noise properties of the ZEKE spectrometer.

2.1 Anion PES and ZEKE spectroscopy

Two related experimental techniques that have been used to great advantage for the study of van der Waals clusters have been anion PES, and ZEKE spectroscopy. In both of these techniques, one begins by producing an anion and photodetaching the excess electron with a laser pulse. In this way one obtains information about both the anion and neutral species. An advantage of anion studies over other types of spectroscopic probes of clusters, is that because one begins with charged species, one can mass select the species of interest. Another advantage is that because one starts from the anion one can in most cases detach to the ground state of the corresponding neutral, and thus obtain direct spectroscopic information about the ground neutral state. Information about the ground states of neutral species is often difficult to gather from optical spectroscopic studies of neutral clusters, such as laser induced fluorescence (LIF) or resonance-enhanced multi-photon ionization (REMPI), which usually are more

informative about the excited states than the ground states. Also, from the photoelectron or ZEKE spectrum, one can directly determine the electron affinity (*EA*) of the neutral species, which provides information about the relative binding energies of the anion and neutral. Finally, both techniques provide valuable information about anionic clusters, which in general are less well studied and understood than neutral systems--for which a more extensive literature exists. (For a recent review of studies of small neutral clusters, see Bačic and Miller.[1])

2.1.1 Anion photoelectron spectroscopy

In anion PES, one produces a source of "cold" clusters using a supersonic expansion anion source (described later in more detail), which are then mass-selected and photodetached with a fixed frequency laser. By measuring the kinetic energy of the photoelectrons, one can, knowing the energy of the detachment laser, infer the amount of internal energy remaining in the neutral, and thus interpret the observed spectrum in terms of spectroscopic transitions from the anion to the neutral. The concept of anion PES is shown schematically in Figures 2.1 and 2.2. For an introduction to the literature on anion photoelectron spectroscopy and a detailed description of the anion photoelectron spectrometer used in the Neumark group, see the dissertation of A. Weaver.[2] In general, the resolution of conventional anion photoelectron spectroscopy is at best about 5-10 meV (40-80 cm^{-1}). This resolution is sufficient to resolve electronic transitions, and in many cases vibrational transitions of covalently bound molecules, but is insufficient to observe the much finer vibrational structure of weakly bound van der Waals clusters.

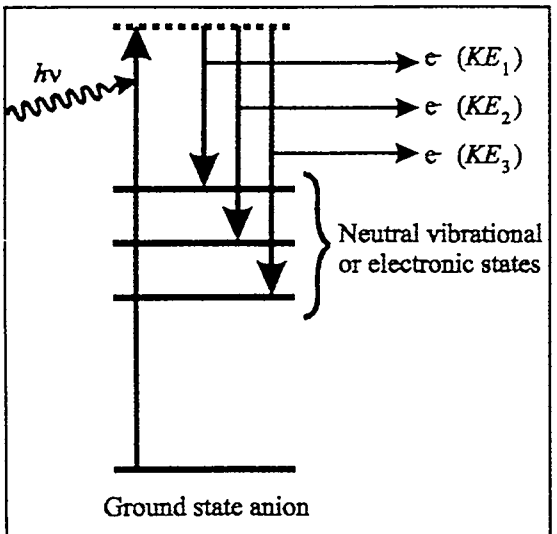

Figure 2.1. Schematic diagram of anion photoelectron spectroscopy

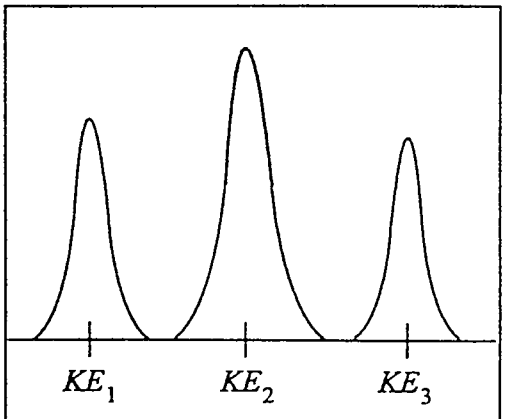

Figure 2.2 The photoelectron spectrum corresponding to Figure 2.1.

Anion PES has been used in several studies of weakly bound clusters. In the Neumark group, $X^-(CO_2)_n$, (X = Cl, Br, I)[3,4], and $I^-(N_2O)_n$[4] clusters have been studied with this technique. More recently, time-resolved PES has been performed in this group on the $I_2^-(Ar)_n$[5] and $I_2^-(CO_2)_n$[6] clusters. Other groups have used anion PES to study I^-Xe_n[7], $(CO_2)^-_n$[8] O^-Ar_n,[9] $X^-(H_2O)_n$[10,11,12] and $X^-(CH_3CN)_n$[13]. From this PES work, one can determine the *EA*s of the clusters, and from these make inferences about their binding energies, but cannot study the details of the low-frequency van der Waals vibrational structure.

2.1.2 Anion ZEKE and threshold photodetachment spectroscopy

Anion zero electron kinetic energy (ZEKE) spectroscopy, the primary focus of this work, offers significant advantages over conventional PES for the study of clusters. With ZEKE spectroscopy we are able to obtain a resolution of at best 1 cm^{-1}, but more

typically 2-3 cm^{-1} (or 0.1-0.4 meV), nearly two orders of magnitude better than conventional PES. This resolution is sufficient to observe the vibrational structure of small van der Waals clusters, as well as to determine very accurate *EA*s. In addition, regardless of the resolution, one can have more confidence in the accuracy of *EA*s determined from ZEKE (or threshold photodetachment) spectroscopy because the *EA*s determined from PES are based on measurement of electron kinetic energies, which are subject to various systematic errors (such as space charge effects), whereas in ZEKE and threshold photodetachment the energy calibration depends only on the calibration of the tunable lasers used in these techniques. The technique is not without its limitations, however, as will be discussed below.

The basic concept of ZEKE spectroscopy was first applied to the photoionization of neutral molecules by Müller-Dethlefs, Schlag and co-workers.[14,15] Since this time ZEKE (also known and pulsed field ionization, or PFI) spectroscopy of neutrals has been employed by many groups to study a wide variety of systems.[16] There has also been extensive theoretical work undertaken to understand the processes taking place in neutral ZEKE-PFI experiments (see, for example, Ref. 17)

ZEKE spectroscopy was first applied to the photodetachment of anions by Neumark and co-workers.[18] In contrast to the rapid growth in the field of neutral ZEKE-PFI, subsequent work in the field of anion ZEKE spectroscopy has been confined to a small number of other groups.[19-21] This is due for the most part to the fundamental difference between ZEKE-PFI of neutrals, in which molecules are excited to long-lived Rydberg states and then ionized by a pulsed electric field, and ZEKE of anions, in which the electron is photodetached from the anion in the first step because anions do not

posess long-lived Rydberg states. However, despite its difficulty, anion ZEKE spectroscopy has proven to be a unique and invaluable tool for the study of clusters, and we exepect it will continue to be in the future.

In anion ZEKE spectroscopy, as in anion PES, one begins by creating a mass-selected packet of anions using pulsed molecular beam techniques, so that one may isolate the species of interest. In ZEKE spectroscopy, one studies the photodetachment process of the anions by scanning a tunable laser over the region of spectroscopic interest, with the apparatus arranged so that only electrons detached with very nearly zero kinetic energy are detected. Electrons ejected with excess kinetic energy are not detected. In this way one obtains a spectrum with peaks corresponding to anion → neutral transitions as a function of laser energy. This process is shown in schematic form in Figures 2.3 and 2.4.

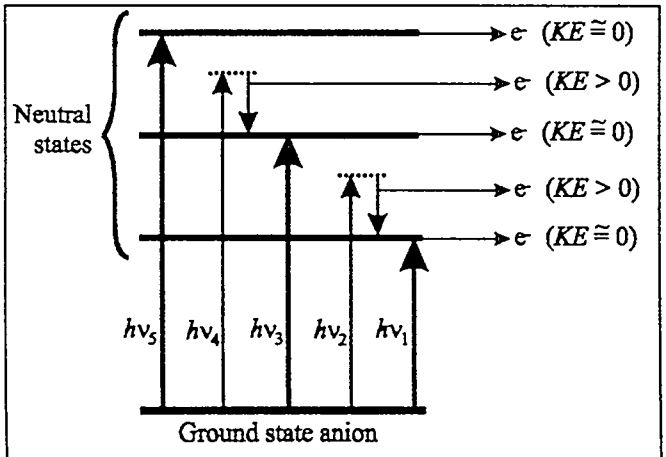

Figure 2.3. Schematic of the process of anion ZEKE spectroscopy.

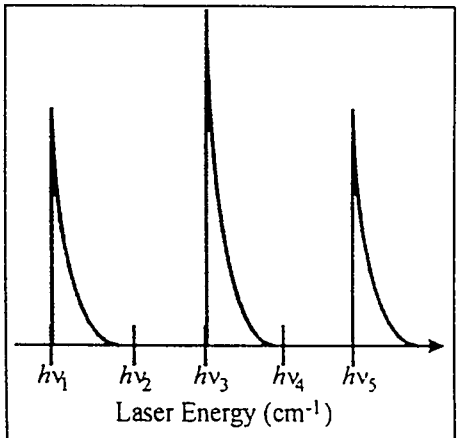

Figure 2.4. The ZEKE spectrum corresponding to Figure 2.3.

The key to the success of the ZEKE technique is the ability to discriminate against the detection of electrons with excess kinetic energy. This is accomplished through a combination of spatial and temporal filtering of the photoelectrons. These techniques

will be described in Section 2.2. Here we comment on the asymmetric line shape observed in ZEKE spectra and shown schematically in Figure 2.4. This lineshape appears because the onset of the detachment threshold is much sharper than the discrimination function of the ZEKE spectrometer, so that one tends to see peaks with "tails" at high energy, due to incomplete discrimination against high energy electrons. Refer to Chapter 3 for a discussion of the detailed form of the ZEKE lineshape. In the next section we discuss the nature of the photodetachment cross-sections near threshold and its ramifications for ZEKE spectroscopy, as well as the partially discriminated threshold photodetachment (PDTP) experiments described in Chapter 4.

2.1.3. Wigner's Law

The problem of the total photodetachment cross section near threshold was first addressed by Wigner,[22] who proposed what has become known as "Wigner's Law." If ℓ is the orbital angular momentum quantum number of the detached electron, then the photodetachment cross section, σ, according to Wigner's Law is

$$\sigma \propto E^{\ell+1/2} \tag{2.1}$$

where E is the kinetic energy of the detached electron, which is equal to $h\nu - E_0$, where $h\nu$ is the photon energy and E_0 is the threshold energy. This relation places restrictions on the systems that may be studied with anion ZEKE spectroscopy. Only those thresholds with sharply rising cross sections can be observed with ZEKE spectroscopy. For the cross section to rise sharply at the threshold, it is necessary to have $\ell = 0$. This occurs, for example, when an electron is detached from a p-orbital of an atom. In order for angular momentum to be conserved when the a photon is absorbed, the orbital angular

momentum of an electron must change according to the selection rule $\Delta\ell = \pm 1$ when it is ejected from an atom. Thus when an electron is photodetached from a p-orbital, it will have $\ell = 0$ (s-wave detachment) or $\ell = 2$ (p-wave detachment). ZEKE spectroscopy is only sensitive to s-wave electrons.

Wigner's law has been experimentally verified for a number of atoms,[23-27] however the range of validity of the Wigner law has been open to question. A recent study of the total photodetachment cross-section of Al^- by Calabrese *et. al.*[28] found good agreement with Wigner's law up to 13 meV (105 cm^{-1}) above threshold. Above this, however, these workers found that the experimental cross-section deviates significantly from the prediction of Equation (2.1). They in fact found a leveling off and *decrease* in the total cross-section, whereas Wigner's law predicts a monotically monotonically increasing cross section. In Figure 2.5 we show the total cross-section of I^- measured in this laboratory, and see that it is well fit by the Equation (2.1) up to 150 cm^{-1} (19 meV) above threshold.

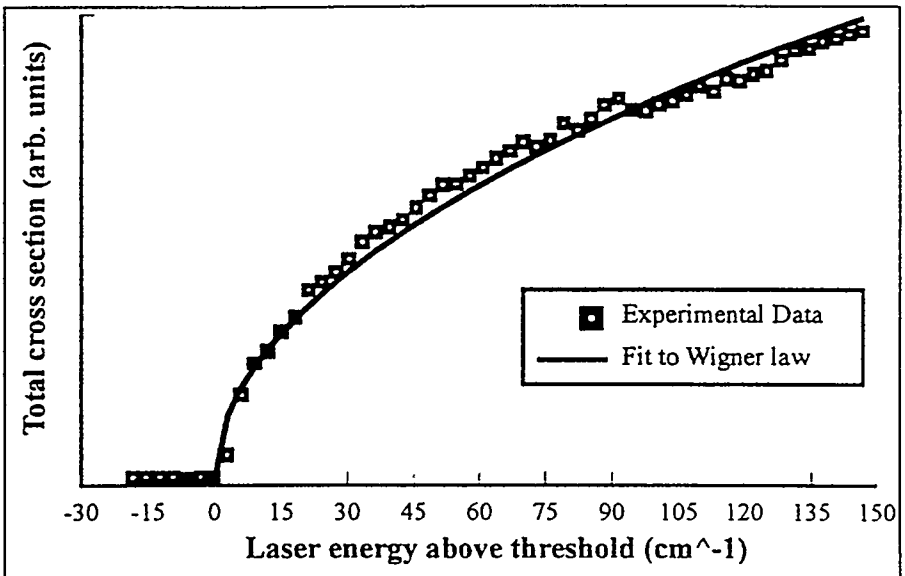

Figure 2.5. The total cross section for the photodetachment process $I^-(^1S) + h\nu \rightarrow I(^2P_{3/2}) + e^-$

The total cross section was not measured at higher energies, so this result can only be considered as a lower bound on the range of validity of Wigner's law for the case of I^-. This result is of little significance for ZEKE spectroscopy, but is of some interest for the "partially discriminated threshold photodetachment" (PDTP) experiments on Rg_nX^- clusters discussed in chapter 4.

Another question worth raising with regard to Wigner's law is whether its range of validity changes as more solvent atoms are added to the halide chromophore of a cluster. Theoretical work by O'Malley[29] suggested a correction term to the Wigner law proportional to the polarizability of the neutral left behind by the departing electron. For s-wave detachment, O'Malley's formulation of the cross-section behavior is given in atomic units by

$$\sigma \propto E^{1/2}\left[1-\tfrac{4}{3}\alpha_d E \ln(2E)+O(E)\right] \qquad (2.2)$$

where α_d is the polarizability of the neutral after photodetachment. One might wonder from this if adding polarizable solvent atoms to the halide would influence the observed PDTP spectra. From our studies of Ar_nI^- and Ar_nBr^- clusters--to be discussed in Chapter 4--this appears not to be the case for Ar atoms. However, whether this effect may influence the observed detachment spectra of halide atoms clustered with more polarizable rare gases, such as the Xe_nI^- clusters recently studied in this group[30] remains to be seen.

Wigner's law has been extended to understand the near-threshold behavior of molecular systems.[31] Because the systems studied in this work may be thought of essentially as perturbed ^1S halide atoms, we shall not concern ourselves with the details of this work. For an interesting discussion of deviations from the Wigner law for molecules with strong dipole moments, see the recent experimental results of Lineberger and co-workers on photodetachment of the OH^- anion.[32] This work is not of direct relevance here, where we are concerned with non-polar rare-gas solvents, but may be of some importance for the interpretation if the ZEKE spectra of clusters containing highly polar solvent molecules.

2.2 The experimental apparatus

The details of the design and operation of the experimental apparatus have been described previously,[33,34] and specific experimental details are given in Chapters 3 and 4 of this work. Therefore here we briefly summarize the operation of the apparatus, and then focus on recent improvements in its design.

The experimental apparatus is shown schematically in Figure 2.6.

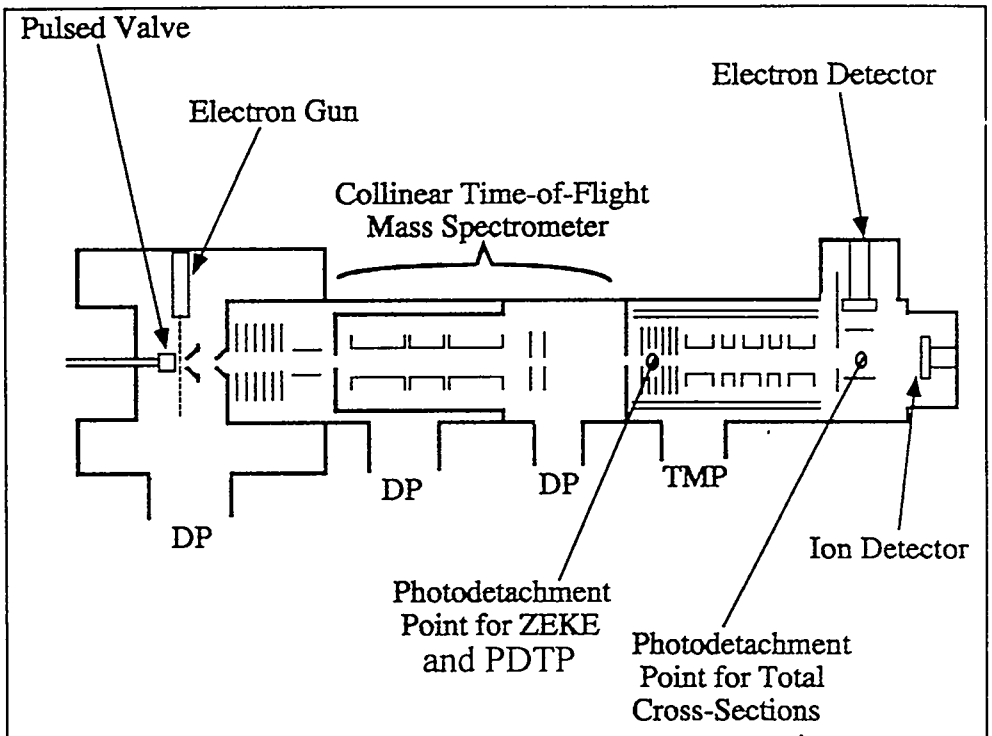

Figure 2.6 The experimental apparatus for ZEKE and threshold photodetachment spectroscopy. DP = Diffusion pump, TMP = Turbomolecular pump.

The cold anions are produced in a supersonic expansion from a pulsed valve crossed with a 1 keV electron beam in the source chamber shown on the left-hand side of the diagram. The expansion passes through two skimmers and enters the first differentially pumped region, where the ions are accelerated to 1 keV. The second skimmer placed very close to the pulsed valve was found to increase production of large clusters, and this "double skimmer" setup is described in more detail in Section 2.2.1. The ions enter the second differentially pumped region and are separated according to mass using a Bakker-type (collinear) time-of-flight arrangement.[35] Finally, the ions enter the detector region shown on the right hand side of the diagram where the mass-selected ion packet of interest is irradiated with a pulsed laser. The electrons are extracted by a weak electric field, and deflected 90° upward to the microchannel plate electron detector. The ion signal is

measured at the microchannel plate detector at the far end of the machine. Total cross sections, such as that shown in Figure 2.5, may be measured by detaching the ions at a point directly beneath the electron detector, so that effectively 100% of the detached electrons are detected.

When operating in partially discriminated threshold photodetachment (PDTP) mode, the electrons are intersected by the laser at a point approximately 0.5 m from the electron detector, and immediately extracted with a weak electric field (*ca.* 1 to 3 V/cm) applied to the extraction plates. In this mode, off-axis high-energy electrons are discriminated against. This mode of operation is equivalent to the "steradiancy detector" described by Spohr *et al.*[36] When optimized for maximum electron signal, the resolution in this mode of operation is about 150 cm^{-1}.[37] However, we have found that by using very low extraction voltages and carefully tuning the electron einzel lens--pictured to the right of the electron extraction plates--to regulate the effective "aperture" size of the steradiancy detector, resolution on the order of 10 cm^{-1} may be achieved in PDTP mode. This enhanced resolution, however, comes at the expense of dramatically decreased electron signal.

To take the ZEKE spectrum of an anion, the anion is photodetached at the same point used for PDTP, however rather than extracting immediately, the electron extraction is delayed by 250-500 ns. During this "waiting period," the higher energy electrons disperse. Thus the "steradiancy effect" for discriminating high energy electrons is greatly enhanced. Furthermore, during the delay, the electrons that are scattered along the beam axis have time to spread out, so that those with different kinetic energies experience different acceleration because of the differing amounts of time they spend in the

extraction field. When the electrons are detected, a 30-120 ns wide gate is used to isolate those with initially zero kinetic energy in the ion beam frame of reference. This gated detection is what is meant by "temporal discrimination," however in the sense that the differences in arrival times of the electrons at the detector are due to the initial spreading-out of the on-axis scattered electrons, this may be also may be thought of as another form of spatial discrimination. In the original design of the ZEKE spectrometer the electrons experienced no electric field during the delay between photodetachment and extraction. Recently, however, we have found that by applying a very small (*ca.* 10 mV/cm) uniform constant electric field to the extraction plates, with polarity opposite to the that of the extraction field, the ZEKE signal is greatly enhanced, with no loss of resolution. The details of this "electron deceleration field" will be described in Section 2.2.2.

2.2.1 "Double skimmer" setup for large cluster production

As mentioned above, it has been found that placing an additional skimmer very close to the opening of the pulsed valve greatly enhances the production of large van der Waals clusters. The skimmer used for this is manufactured by the Beam Dynamics Company, and is contoured, with a 2 mm orifice. It is placed 1 mm from the opening of the pulsed valve (General Valve Corp) which is used with a 0.5 mm or 0.8 mm orifice. This second skimmer is contained within the source chamber; that is, there is no differential pumping on either side of it. The opening of this skimmer is placed 2 cm from the opening of the conical skimmer which separates the source chamber from the first differentially pumped region. This setup is pictured in Figure 2.7.

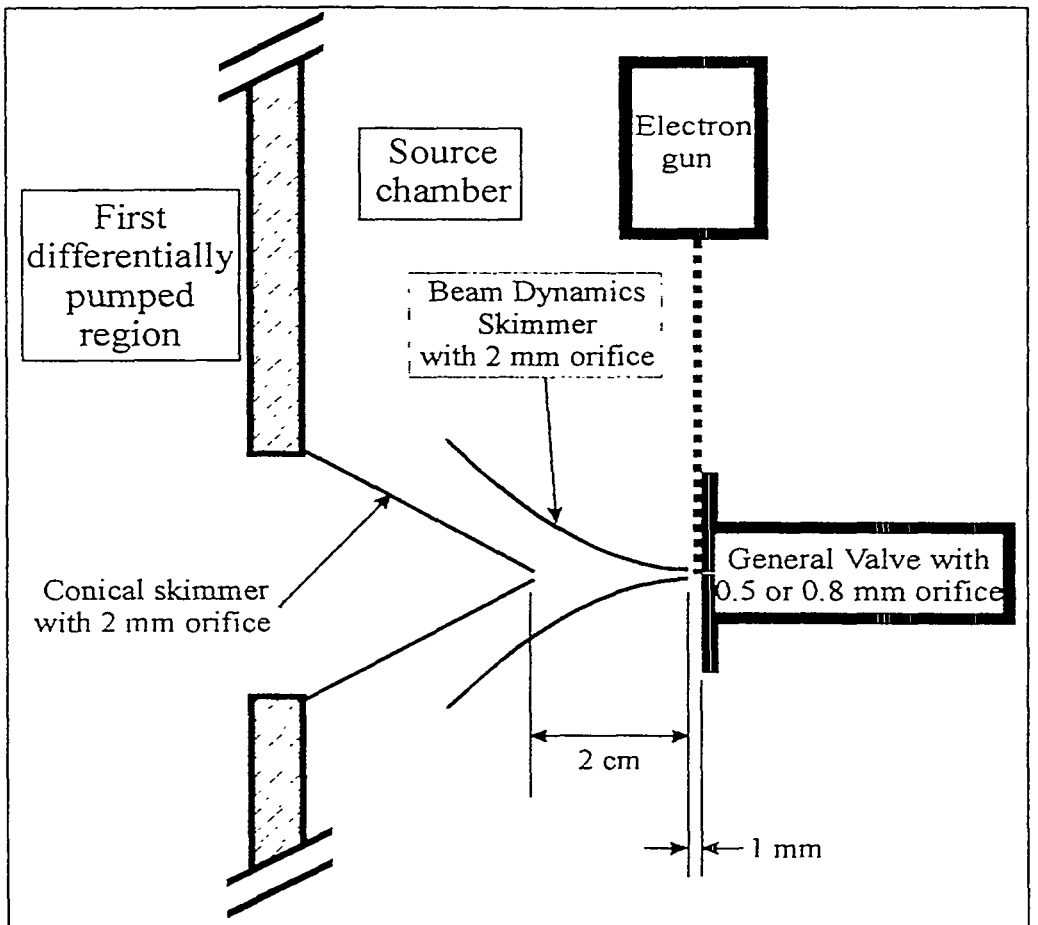

Figure 2.7 The "double-skimmer" setup for large cluster production.

An example of the increased cluster production using this source is shown in Figure 2.8, which shows the mass spectrum of Ar_nCl^- clusters. The mass spectra with and without the second skimmer in place are shown in Figures 2.8(a) and 2.8(b), respectively. We see that without the second skimmer, the cluster with the greatest intensity is $ArCl^-$ and the ion signal of the larger clusters decreases, until it becomes very difficult to observe clusters larger than Ar_7Cl^-. On the other hand, with the double skimmer setup, we can easily observe clusters up to $Ar_{12}Cl^-$. The ion signal for clusters with n larger than 12 drops suddenly, possibly due to a solvent shell closing at this point;[38] however clusters up to $Ar_{19}Cl^-$ are still plainly visible in the mass spectrum. This same enhancement in

large cluster production has also been observed in the mass spectra of other systems studied in this laboratory, including Xe_nI^-[30] and $Cl^-(N_2)_n$[39].

The reason for the success of this scheme for producing large anion clusters is not certain. Relative to the knowledge about the formation of neutral clusters in supersonic expansions,[40,41] little is known about the clustering processes that take place in this type of anion source. It is possible that the skimmer near the pulsed valve acts as a kind of "clustering channel" allowing more collisions to take place between the halide anions and rare-gas atoms in this confined space, so that larger clusters are formed.

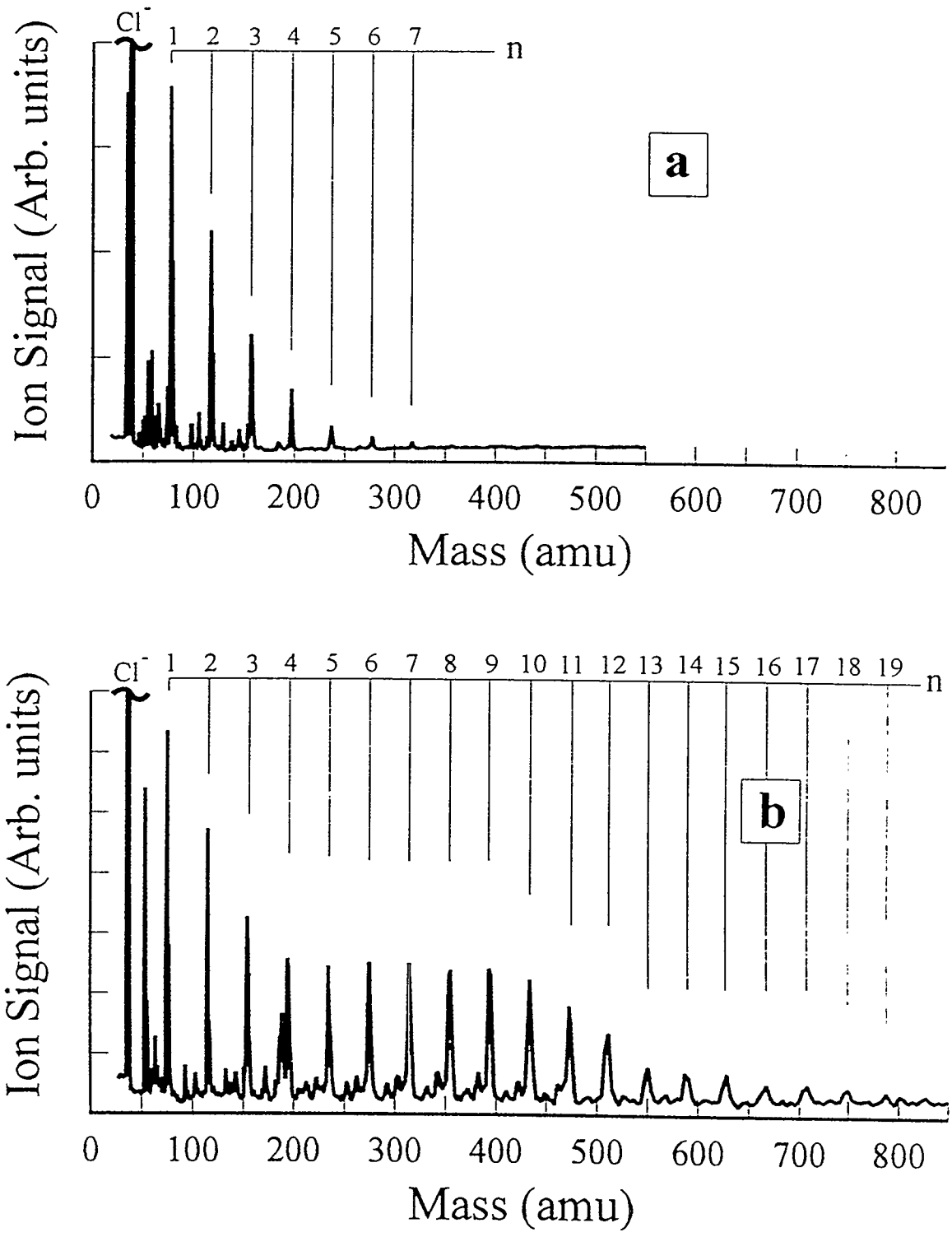

Figure 2.8. The mass spectra of Ar_nCl^-: (a) Without the second skimmer near the pulsed valve. (b) With the double-skimmer setup shown in Figure 2.7. The impurity peak at 53 amu is $Cl^-(H_20)$.

2.2.2 Electron deceleration field

We mentioned above our finding that the application of a weak, positive electric field to the electron extraction plates results in enhancement of the observed ZEKE signal. Here we describe the details of the implementation of this scheme.

The extraction field is applied to of a series of seven molybdenum plates spaced 1 cm apart. Photodetachment takes place between the first two plates. The field is applied such that there is a potential gradient across the first four of the plates. The voltage for electron deceleration is applied at the same place. A schematic diagram of the circuit used to apply the deceleration voltage is shown in Figure 2.9.

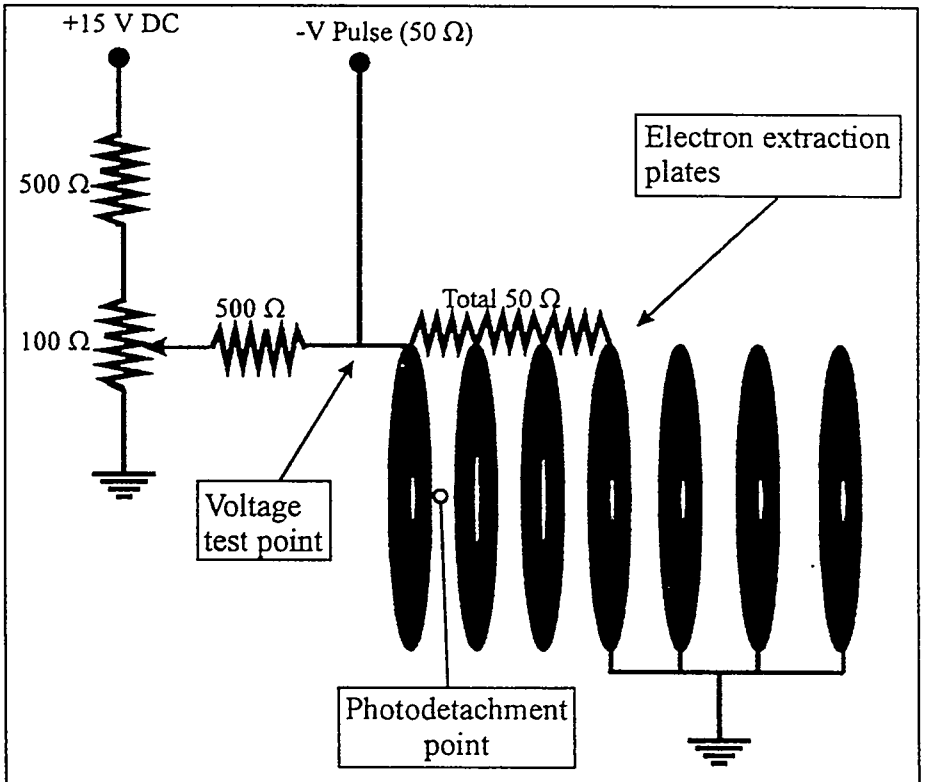

Figure 2.9. The circuit for the electron deceleration field.

This is a simple voltage divider circuit, but a few points should be borne in mind in its use. The values shown for the resistances should not be changed too much, because if the

impedance from the circuit it too high or low it will interfere with the pulsed extraction. The values of the resistances shown were chosen empirically to minimize interference with the extraction pulse. Also this circuit should have its own stable power supply, not used by any other device in the apparatus, to ensure stability and reproducibility of the voltage. Finally, when testing the voltage at the indicated test point with a voltmeter or oscilloscope, one should not leave the probe connected when the experiment is running, because the probe will disrupt the extraction pulse. The DC voltage should be measured with a high impedance probe, rather than with the 50Ω probe of an oscilloscope in order not to alter the voltage from the voltage divider.

An example of the improvement in signal with the electron deceleration field is shown in Figure 2.10. This figure shows the ZEKE spectrum of the Cl^- atom (detaching to the ground $^2P_{3/2}$ state of Cl, without [Fig. 2.10(a)] and with [Fig 2.10(b)] the electron deceleration field circuit in place. In both cases, the spectra are accumulated over 400 laser shots per point. The voltage measured at the test point with a high impedance probe was, 8 mV, corresponding to an electric field of about 3 mV/cm. Both ordinates in the figure are calibrated to the absolute number of electrons collected. The resolution in both cases is 2 cm^{-1} FWHM, but we see a nearly three-fold increase in ZEKE signal when the electron deceleration circuit is used. It is found that with the deceleration field, much longer delay times can be employed. For example, when detaching the Cl^- atom it was possible to see ZEKE signal at delay times of up to 1 μs. However, the resolution seemed to deteriorate at delay times above 500 ns. We find that when using the deceleration field, it is necessary to use higher deceleration voltages when shorter delays are employed to achieve optimal results.

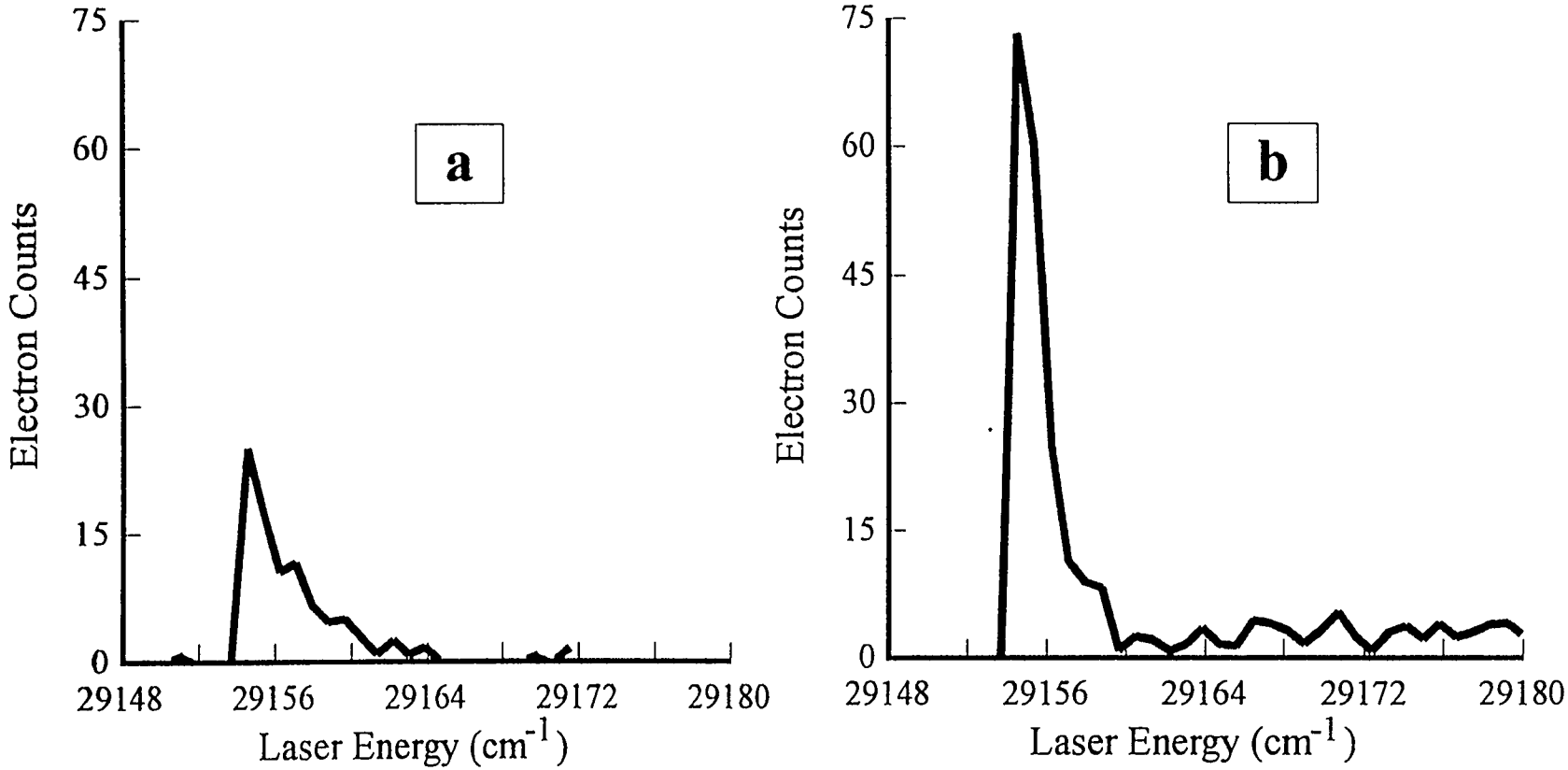

Figure 2.10. A comparison of the ZEKE spectrum of Cl^- (detachment to $^2P_{3/2}$ Cl): (a) Without the electron deceleration field. (b) With an electron deceleration field of *ca.* 3 mV/cm.

The reason for the improved ZEKE signal seen with this arrangement is not known with certainty. One possibility is that the deceleration field separates the detached electrons from the center of mass of the remaining undetached anions to a great enough extent that they are less subject to dispersal by the space charge of the remaining anions. It is also possible that the longer delay times are made possible by the deceleration of the electrons, which--especially light atoms like Cl^---would otherwise travel out of the extraction field within a few hundred ns. The increased electron signal may also be due to the fact that the decelerated electrons experience greater acceleration when the extraction field is applied, because they begin further up on the extraction potential gradient, and hence are better separated and collected with greater efficiency.

2.3. References for Chapter 2

[1] Z. Bacic and R. E. Miller, J. Phys. Chem. **100**, 12945 (1996).

[2] A. Weaver, Ph.D. Thesis, University of California, Berkeley, 1991.

[3] D. W. Arnold, S. E. Bradforth, E. H. Kim, and D. M. Neumark, J. Chem. Phys. **102**, 3493 (1995).

[4] D. W. Arnold, S. E. Bradforth, E. H. Kim, and D. M. Neumark, J. Chem. Phys. **102**, 3510 (1995).

[5] B. J. Greenblatt, M. T. Zanni, and D. M. Neumark, Science **276**, 1675 (1997).

[6] M. T. Zanni, T. Taylor, B. J. Greenblatt, and D. M. Neumark, (to be published).

[7] I. Becker, G. Markovich, and O. Cheshnovsky, Phys. Rev. Lett. **79**, 3391 (1997).

[8] T. Tsukuda, M. A. Johnson, and T. Nagata, Chem. Phys. Lett. **268**, 429 (1997).

[9] S. T. Arnold, J. H. Hendricks, and K. H. Bowen, J. Chem. Phys. **102**, 39 (1995).

[10] G. Markovich, R. Giniger, M. Levin, and O. Cheshnovsky, J. Chem. Phys. **95**, 9416 (1991).

[11] G. Markovich, S. Pollack, R. Giniger, and O. Cheshnovsky, Z. Physik D **26**, 98 (1993).

[12] G. Markovich, S. Pollack, R. Giniger, and O. Cheshnovsky, J. Chem. Phys. **101**, 9344 (1994).

[13] G. Markovich, L. Perera, M. L. Berkowitz, and O. Cheshnovsky, J. Chem. Phys. **105**, 2675 (1996).

[14] K. Müller-Dethlefs, M. Sander, and E. W. Schlag, Z. Naturforsch. Teil A **39**, 1089 (1984).

[15] K. Müller-Dethlefs, M. Sander, and E. W. Schlag, Chem. Phys. Lett. **12**, 291 (1984).

[16] K. Müller-Dethlefs, J. Electron Spectrosc. and Related Phenomena **75**, 35 (1995).

[17] P. Bellomo, D. Farrelly, and T. Uzer, J. Phys. Chem. A **101**, 8902 (1997).

[18] T. N. Kitsopoulos, I. M. Waller, J. G. Loeser, and D. M. Neumark, Chem. Phys. Lett. **159**, 300 (1989).

[19] G. Gantefor, D. M. Cox, and A. Kaldor, J. Chem. Phys. **94**, 854 (1990).

[20] G. Drechsler, C. Bassmann, U. Boesl, and E. W. Schlag, Z. Naturforsch. **49A**, 1256 (1994).

[21] C. Bassmann, U. Boesl, D. Yang, G. Drechsler, and E. W. Schlag, Intl. J. Mass Spec. Ion Proc. **159**, 153 (1996).

[22] E. P. Wigner, Phys. Rev. **73**, 1002 (1948).

[23] W. C. Lineberger and B. W. Woodward, Phys. Rev. Lett. **25**, 424 (1970).

[24] H. Hotop, T. A. Patterson, and W. C. Lineberger, Phys. Rev. A **8**, 762 (1973).

[25] D. Feldmann, Chem. Phys. Lett. **47**, 338 (1977).

[26] H. Hotop and W. C. Lineberger, J. Chem. Phys. **58**, 2379 (1973).

[27] K. R. Lykke, K. K. Murray, and W. C. Lineberger, Phys. Rev. A **43**, 6104 (1991).

[28] D. Calabrese, A. M. Covington, J. S. Thompson, R. W. Marawar, and J. W. Farley, Phys. Rev. A **54**, 2797 (1996).

[29] T. F. O'Malley, Phys. Rev. **137**, A1668 (1964).

[30] T. Lenzer, M. Furlanetto, N. Provinka, and D. M. Neumark, (to be published).

[31] K. J. Reed, A. H. Zimmerman, H. C. Andersen, and J. I. Brauman, J. Chem. Phys. **64**, 1368 (1976).

[32] J. R. Smith, J. B. Kim, and W. C. Lineberger, Phys. Rev. A **55**, 2036 (1997).

[33] T. N. Kitsopoulos, Ph.D. Thesis, University of California, 1991.

[34] C. C. Arnold, Ph.D. Thesis, University of California, 1994.

[35] M. B. Bakker, J. Phys. E **7**, 364 (1974).

[36] R. Spohr, P. M. Guyon, W. A. Chupka, and J. Berkowitz, Rev. Sci. Instrum. **42**, 1872 (1971).

[37] C. J. Chick, Y. Zhao, T. N. Kitsopoulos, and D. M. Neumark, J. Chem. Phys. **97**, 6121 (1992).

[38] T. Lenzer, I. Yourshaw, M. Furlanetto, and D. M. Neumark, (to be published).

[39] I. Yourshaw, T. Lenzer, and D. M. Neumark, (unpublished results).

[40] O. F. Hagena, Surface Science **106**, 101 (1981).

[41] S. DePaul, D. Pullman, and B. Friedrich, J. Phys. Chem. **97**, 2167 (1993).

Chapter 3. Zero electron kinetic energy (ZEKE) spectroscopy of the $KrBr^-$, $XeBr^-$, and $KrCl^-$ anions*

Abstract

Three rare-gas halide (RgX^-) anions, $KrBr^-$, $XeBr^-$, and $KrCl^-$, and the corresponding neutral, open-shell van der Waals complexes are studied with anion zero electron kinetic energy (ZEKE) spectroscopy. The spectra for each system reveal well-resolved progressions in the low frequency vibrations of the anion and one or more of the three neutral electronic states accessed by photodetachment, providing a detailed spectroscopic probe of the Rg-X^- and Rg-X interaction potentials. In the case of $KrBr^-$, transitions to all three of the "covalent" neutral electronic states (the $X\,^2\Sigma_{1/2}$, $I\,^2\Pi_{3/2}$, and $II\,^2\Pi_{1/2}$ states) were observed. For $XeBr^-$, transitions to the $X\,^2\Sigma_{1/2}$ and $II\,^2\Pi_{1/2}$ neutral states were observed. For $KrCl^-$, only the $X\,^2\Sigma_{1/2}$ state could be studied. From our data, we construct model potentials for the anion and each observed neutral state, and these are compared with other experimental and theoretical potentials.

* Submitted to J. Chem. Phys. in slightly different form with co-authors Thomas Lenzer, Georg Reiser, and Daniel M. Neumark.

3.1. Introduction

The characterization of the forces between weakly interacting species has attracted a great deal of experimental and theoretical attention in recent years. The interaction potentials between closed shell neutral species have been characterized in considerable detail as a result of this effort.[1,2] However, less is known about the interactions between open and closed shell atoms, or about those between ions and neutrals. In this work we describe new experimental results involving the latter two types of interactions. We report the results of studies of the rare gas-halide atom complexes $KrBr^-$, $XeBr^-$, and $KrCl^-$ using anion zero electron kinetic energy (ZEKE) spectroscopy. In these experiments we obtain spectroscopic information on both the neutral and negatively charged complex and derive accurate potentials for both the anion and neutral species. This work is a continuation of our earlier ZEKE studies of the $ArBr^-$, KrI^-, and ArI^- complexes,[3,4] and is part of an ongoing effort to obtain ZEKE spectra of the complete series of rare gas halides.

The rare gas halide (RgX^-) species are of interest because the $Rg{\cdot}X^-$ interaction potentials determine the transport properties of halide ions in rare gases; these are important in understanding plasmas and gas discharges. Prior to the work reported here, the only previous experimental results on the $KrBr^-$, $XeBr^-$ and $KrCl^-$ anions came from ion mobility studies,[5,6] from which potentials can be obtained by iterative fitting or direct inversion. Interaction potentials had also been derived within the framework of theoretical[7,8] and semi-empirical[9-11] models. The work described here provides a direct spectroscopic probe of these species.

The rare gas-halogen (RgX) complexes are important for their use in excimer lasers, in which lasing takes place between electronically excited, strongly bound charge transfer states and the repulsive wall of the weakly bound covalent ground states.[12] Excimer emission has also provided spectroscopic information on the charge-transfer and covalent states. In the cases of KrBr and KrCl, emission from the RgX charge transfer

states to the ground state (the B→X band) is broad and relatively unstructured,[12] as is typical of bound-free transitions. However, recent emission studies of the B→X band in XeBr reveal extensive vibrational structure.[13,14] The covalent states of rare gas-halogen neutrals have also been probed in a series of scattering experiments. Information on the RgX species studied here comes from differential cross-section crossed molecular beam experiments of Lee and coworkers,[15] which yielded potentials for the KrBr and XeBr complexes, and from integral cross section measurements by Aquilanti and co-workers[16] who have characterized the potential of KrCl.

The neutral interactions are of interest also because they are simple examples of open shell-closed shell interactions. The two spin-orbit states of the ^2P halogen atom interact with the rare gas to give rise to three molecular electronic states.[17,18] The lower $^2P_{3/2}$ state is split by the electrostatic interaction into two components, corresponding to $\Omega = 1/2$ (the $X\,\frac{1}{2}$ state, in the notation used here) and $\Omega = 3/2$ (the $I\,\frac{3}{2}$ state), where Ω is the projection of the total electronic angular momentum along the internuclear axis. The upper $^2P_{1/2}$ halogen state gives rise to the $II\,\frac{1}{2}$ ($\Omega = 1/2$) state in the complex.

Anion ZEKE spectroscopy of rare gas halides probes the van der Waals well region of the covalent states; this complements earlier studies of emission from excimer states. Our experiments also complement the scattering experiments because, whereas the scattering cross-sections contain information about the *absolute* values of the bond length and well depths of the complexes, the ZEKE spectra are sensitive only to the *relative* differences between the anion and neutral potentials. However, in the ZEKE spectra one can observe vibrationally-resolved photodetachment transitions to the various neutral electronic states, whereas in the crossed beam experiments the contributions of the $X\,\frac{1}{2}$ and $I\,\frac{3}{2}$ states to the experimental signal are not clearly separated and must be extracted by an appropriate data inversion procedure. Also, in the crossed beam experiments involving Br or I atoms, the $II\,\frac{1}{2}$ electronic state arising from the upper $^2P_{1/2}$ spin-orbit state of the halogen atom is generally not probed because the population of this

state is negligible. In the ZEKE experiments, well-resolved spectra of the $II\frac{1}{2}$ states of the KrBr and XeBr systems are seen, and accurate potentials can be derived for these states for the first time.

The anion potentials derived here are a significant improvement over previously available potentials. While our ground state potentials for KrBr and XeBr are essentially the same as those derived from scattering and excimer emission experiments, our excited states potentials represent improvements over previous work, particularly for the $II\frac{1}{2}$ state. In the case of KrCl, our spectra confirm the neutral potentials previously deduced from the scattering experiments.

This article is organized as follows: In Section 3.2 we describe the experimental setup for anion production and ZEKE spectroscopy. In Section 3.3 we present the ZEKE spectra of $KrBr^-$, $XeBr^-$ and $KrCl^-$, and assign the observed electronic and vibrational structure. Section 3.4 deals with the construction of model potentials for fitting the vibrational structure and rotational contours of the ZEKE spectra. Finally, we compare our potentials with other experimental and theoretical results in Section 3.5.

3.2. Experimental

ZEKE spectroscopy was originally developed by Müller-Dethlefs *et al.* for photoionization of neutrals.[19-21] It was first applied to the study of anions by Neumark and co-workers.[22] The anion ZEKE apparatus used here has been described in detail elsewhere.[23-25] A brief description follows.

$KrBr^-$ and $XeBr^-$ anions are produced by expanding a mixture of 0.2% CF_2ClBr / 10-30% Kr (or Xe) / balance He into vacuum through a 0.5 mm aperture in a pulsed valve (General Valve Corp.). The expansion is crossed near the pulsed valve with a 1 keV electron beam produced with a thoria-coated iridium filament (Electron Technology). Halide anions are produced by dissociative attachment and other secondary processes, and clusters form as the supersonic expansion cools. $KrCl^-$ anions are produced by

passing the Kr/He mixture over a reservoir containing CCl_4 at room temperature. Backing pressures were typically 20-80 psi.

The anions pass through a skimmer into a differentially pumped region and are accelerated to 1 keV into a 1 m collinear time-of-flight mass spectrometer. The $KrCl^-$ results were obtained using an additional skimmer in the source chamber placed very close to the beam valve.[26] The clusters separate according to mass, and the species of interest is irradiated with a pulse from an excimer-pumped dye laser (Lambda Physik) operating at a repetition rate of 30 Hz. After a 200-500 ns delay, the electrons are extracted coaxially with the ion beam using a pulsed electric field and detected approximately 1 m away with a microchannel plate detector. The electrons are detected in a 35-100 ns gate, so that as the laser wavelength is scanned, only electrons with nearly zero kinetic energy relative to the anion packet are detected. The resulting spectral resolution is about 1-2 cm^{-1} for atomic anions. The peaks observed in this work are somewhat broader because of unresolved rotational structure.

In order to study the $X\,\frac{1}{2}$ and $I\,\frac{3}{2}$ states, DMQ laser dye was used for KrBr and XeBr and PTP dye was used for KrCl. The laser pulse energy was about 20 mJ/pulse for KrBr and XeBr, and about 3-10 mJ/pulse for KrCl. These spectra were averaged over 1000-2000 laser shots/point. For the $II\,\frac{1}{2}$ states of KrBr and XeBr, light from Rhodamine 640 dye was frequency-doubled with a KDP crystal, yielding laser pulse energies of *ca.* 2-4 mJ/pulse. The electron signal was normalized to the ion signal and to the laser pulse energy. Spectra for the $II\,\frac{1}{2}$ states were averaged over about 8000 laser shots per point. When using DMQ and PTP dyes, the laser wavelength was calibrated using a Fe-Ne hollow cathode lamp. An iodine cell was used to calibrate the fundamental wavelength when Rhodamine 640 dye was frequency-doubled. The spectra were smoothed with a 5-point, second order Savitzky-Golay algorithm,[27] which had a negligible effect on the relative peak intensities.

3.3. Results

3.3.1. KrBr

The ZEKE spectra of $KrBr^-$ are shown in Figure 3.1. We observe two band systems, shown in Figures 3.1(a) and 3.1(b), separated by approximately the spin-orbit constant of Br (3685 cm^{-1}). The lower energy band system in Figure 3.1(a) results from transitions to the $X\,\frac{1}{2}$ and $I\,\frac{3}{2}$ states, and the higher energy system in Figure 3.1(b) is due to the $II\,\frac{1}{2}$ state.

Assignment of the vibrational and electronic features in Figure 3.1 is facilitated by our earlier studies of the ArI^-, $ArBr^-$ and KrI^- spectra.[3] Specifically, the anion vibrational frequencies are expected to be considerably larger than the neutral frequencies, and this enables one to distinguish among the three types of neutral←anion vibrational transitions (υ'-υ") that contribute to the spectra: vibrational progressions in the neutral originating from a single anion vibrational level υ", Δυ=0 sequence band transitions from a series of vibrational levels of the anion, and Δυ≠0 hot band transitions from vibrationally excited anion levels.

Fig. 3.1(a) is dominated by one peak, labeled **1**, with a set of smaller peaks, a_1, b_1, and c_1, spaced by about 15 cm^{-1} toward lower energy. A second, weaker progression is seen at higher energies than peak **1** with a characteristic peak spacing of 20 cm^{-1}. We assign peak **1** to the origin (0-0) transition from the anion to the $X\,\frac{1}{2}$ state. Peaks a_1, b_1 and c_1 are assigned to Δυ=0 sequence band transitions from vibrationally excited anion states, i.e. the 1-1, 2-2, and 3-3 transitions. The dominance of Δυ=0 transitions shows that the anion geometry is very similar to the neutral $X\,\frac{1}{2}$ state geometry.

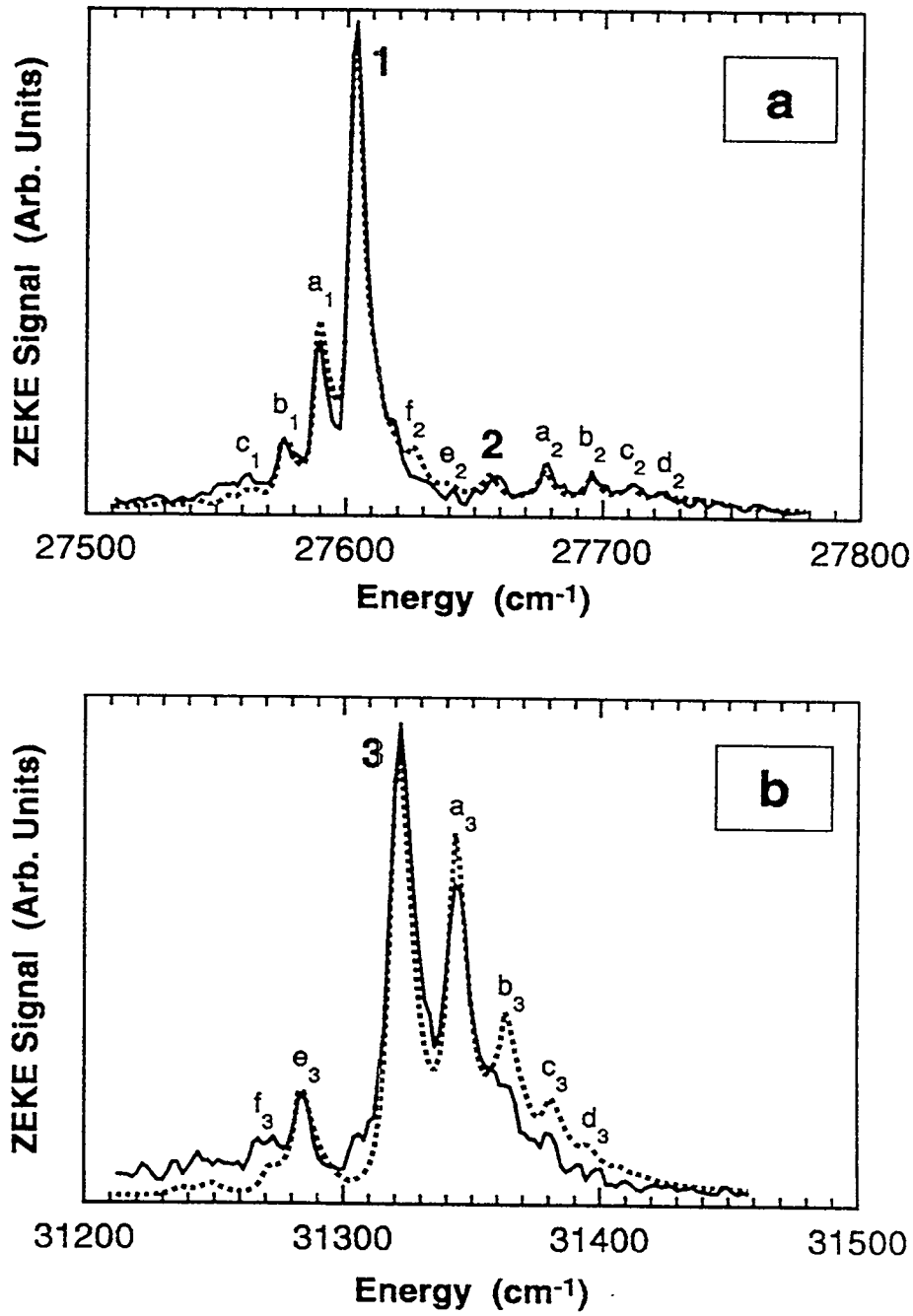

Figure 3.1. Experimental and simulated ZEKE spectra of $KrBr^-$. The solid lines are the experimental spectra, and the dotted lines are the spectra calculated from the model potentials described in the text. **(a)** $X\,\frac{1}{2}$ and $I\,\frac{3}{2}$ states (halogen atom $^2P_{3/2}$ asymptote). **(b)** $II\,\frac{1}{2}$ state (halogen atom $^2P_{1/2}$ asymptote).

Peak **2**, the lowest energy member of the second progression, is assigned to the 0-0 transition to the $I\frac{3}{2}$ state. This assignment is made in part because it gives the best fit to a model potential (See Analysis section, below). Peaks a_2, b_2, c_2, and d_2 are assigned to the (υ'-0) vibrational progression with υ'=1-4 originating from the anion υ"=0 level. The extent of this progression indicates that the $I\frac{3}{2}$ state bond length is significantly shifted from the anion geometry. Peak e_2 is assigned to the 1-1 sequence band of the $I\frac{3}{2}$ state. Peak f_2, 40 cm^{-1} to the red of peak **2** is assigned to the $I\frac{3}{2}$ 0-1 hot band transition, plus several overlapping bands from the $X\frac{1}{2}$ state.

In the $II\frac{1}{2}$ state spectrum, Fig. 3.1(b), we see the progression **3**, a_3, b_3, c_3, and d_3, with a characteristic spacing of about 20 cm^{-1}, and a smaller peak, e_3, 37.2 cm^{-1} below peak **3**. We assign peak **3** to the 0-0 transition to the $II\frac{1}{2}$ state, and the series a_3, b_3, c_3, d_3 to the (υ'-0) progression with υ'=1-4. Peak e_3 corresponds to the 0-1 hot band transition, and gives an accurate value for the anion vibrational frequency.

The complete set of peak positions and assignments is given in Table 3.1.

Table 3.1. Peak Assignments (υ'-υ") for $KrBr^-$ ZEKE spectra (Figure 3.1). Energies are in cm^{-1}.

State	Peak	Position	Relative Energy	Assignment
*X*1/2	**1**	27602.9	0	0←0
	a_1	27588.4	-14.5	1←1
	b_1	27576.2	-26.7	2←2
	c_1	27561.0	-41.9	3←3
*I*3/2	**2**	27657.0	0	0←0
	a_2	27678.4	21.4	1←0
	b_2	27695.3	38.3	2←0
	c_2	27710.7	53.7	3←0
	d_2	27723.0	66.0	4←0
	e_2	27641.7	-15.3	1←1
	f_2	27619.0 (shoulder)	-38.0	0←1
*II*1/2	**3**	31321.7	0	0←0
	a_3	31343.5	21.8	1←0
	b_3	31363.2	41.5	2←0
	c_3	31380.6	58.9	3←0
	d_3	31398.3	76.6	4←0
	e_3	31284.5	-37.2	0←1
	f_3	31274.7	-47.0	1←2

3.3.2. XeBr

The ZEKE spectra of $XeBr^-$ are shown in Figures 3.2(a) and 3.2(b). Our assignment of the peaks proceeds in a fashion similar to the assignment of the KrBr spectrum. Again there are two band systems separated approximately by the Br spin-orbit constant. The lower energy system in Figure 3.2(a) is dominated by a single peak, **1**, with a set of peaks c_1, d_1 and e_1 spaced at *ca.* 17 cm^{-1} intervals toward lower energy. We also observe a pair of small peaks, a_1 and b_1, 26.5 cm^{-1} and 50.2 cm^{-1} to the blue of peak **1**, respectively. As above we assign peak **1** to the origin transition to the $X\,\frac{1}{2}$ state, and the peaks c_1, d_1 and e_1 to the sequence bands 1-1, 2-2 and 3-3, respectively. Peaks a_1

and b_1 correspond to the 1-0 and 2-0 transitions, and are consistent within 2.5 cm^{-1} with the peak spacings calculated from the spectroscopic constants determined by Tellinghuisen and coworkers in their excimer emission study.[14] As above, the dominance of the 0-0 transition shows that the anion bond length is apparently quite close to that of the $X\,\frac{1}{2}$ state. However, in contrast to the $KrBr^-$ spectrum, transitions to the $I\,\frac{3}{2}$ state are not seen in Figure 3.2(a).

The more congested $II\,\frac{1}{2}$ state spectrum in Figure 3.2(b) reveals two vibrational progressions. Peaks **3**, a_3, b_3, c_3, d_3 and e_3 are spaced by 20 cm^{-1} toward higher energy, peaks **3**, f_3 and g_3 are spaced by about 42 cm^{-1} toward lower energy. Based on this change in peak spacing, peaks **3**-e_3 are assigned to a progression arising from the ground anion vibrational state with the origin at peak **3**. Peaks f_3 and g_3 are assigned to the 0-1 and 0-2 hot band transitions, respectively. The $XeBr^-$ peak positions and assignments are given in Table 3.2.

Table 3.2. Peak assignments for $XeBr^-$ ZEKE spectra (Figure 3.2). Energies are in cm^{-1}.

State	Peak	Position	Relative Energy	Assignment
*X*1/2	**1**	27890.0	0	0←0
	a_1	27916.5	26.5	1←0
	b_1	27940.2	50.2	2←0
	c_1	27873.0	-17.0	1←1
	d_1	27855.1	-34.9	2←2
	e_1	27841.9	-48.1	3←3
*II*1/2	**3**	31623.6	0	0←0
	a_3	31647.6	24.0	1←0
	b_3	31667.1	43.5	2←0
	c_3	31687.5	63.9	3←0
	d_3	31704.7	81.1	4←0
	e_3	31719.8	96.2	5←0
	f_3	31583.2	-40.4	0←1
	g_3	31539.8	-83.8	0←2

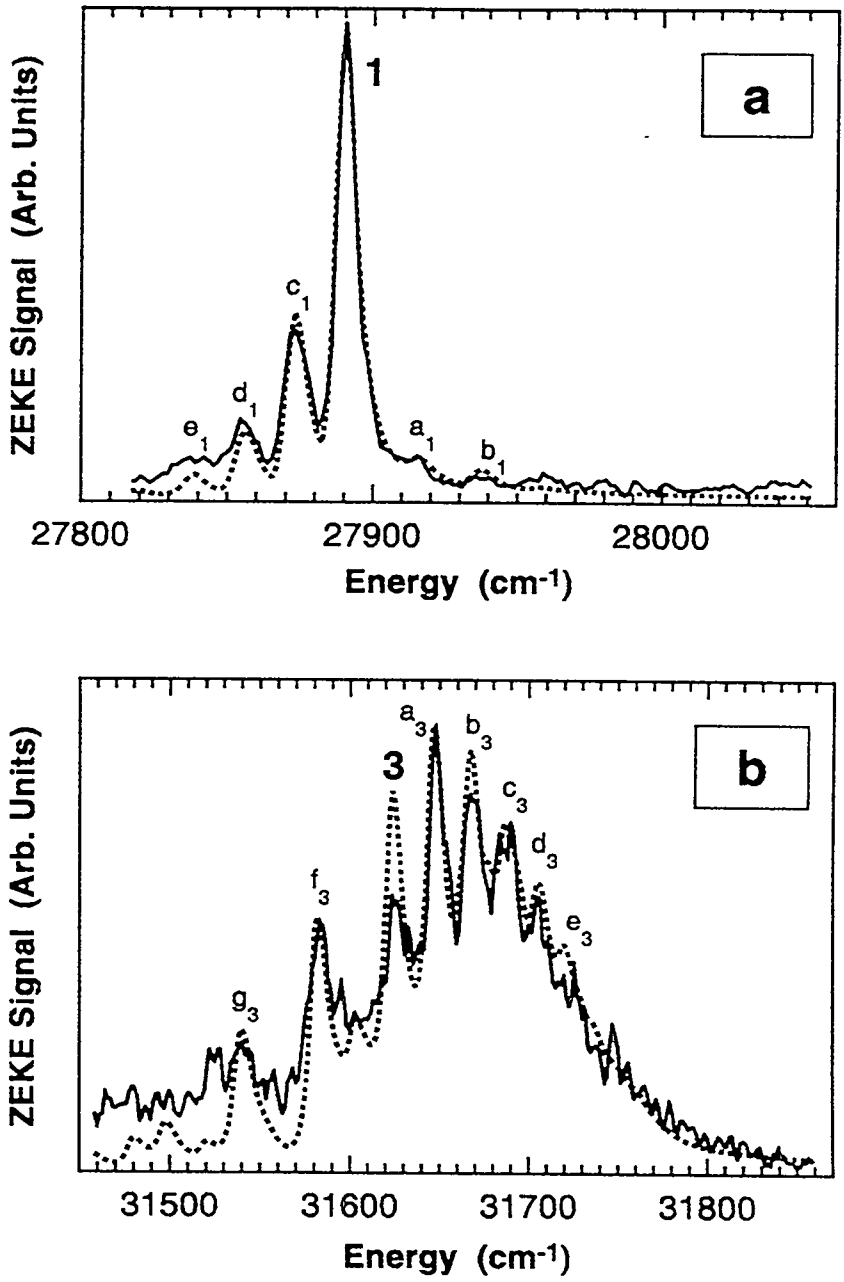

Figure 3.2. Experimental and simulated ZEKE spectra of $XeBr^-$. The solid lines are the experimental spectra, and the dotted lines are the spectra calculated from the model potentials described in the text. (**a**) $X\,\frac{1}{2}$ state (halogen atom $^2P_{3/2}$ asymptote). The $I\,\frac{3}{2}$ state cannot be seen. (**b**) $II\,\frac{1}{2}$ state (halogen atom $^2P_{1/2}$ asymptote).

3.3.3. KrCl

The ZEKE spectrum of $KrCl^-$ is shown in Figure 3.3. The largest peak, labeled 1, is assigned to the origin transition to the $X\,\frac{1}{2}$ state. Peaks **1**, c_1, d_1, e_1 and f_1 are spaced approximately 26 cm^{-1} toward lower energy. The latter four peaks are assigned to $\Delta\upsilon=0$ sequence band transitions, with additional contributions from the hot band transitions listed in Table 3.3. Also, peaks g_1 and h_1 can be assigned to the overlapping hot band transitions given in Table 3.3. The partially resolved peaks a_1 and b_1 to the blue of **1** are assigned to the (υ'-0) progression, yielding a frequency of 29 cm^{-1} for the $X\,\frac{1}{2}$ state. We were not able to observe the $I\,\frac{3}{2}$ state or the $II\,\frac{1}{2}$ state for this system. The peak positions and assignments are shown in Table 3.3.

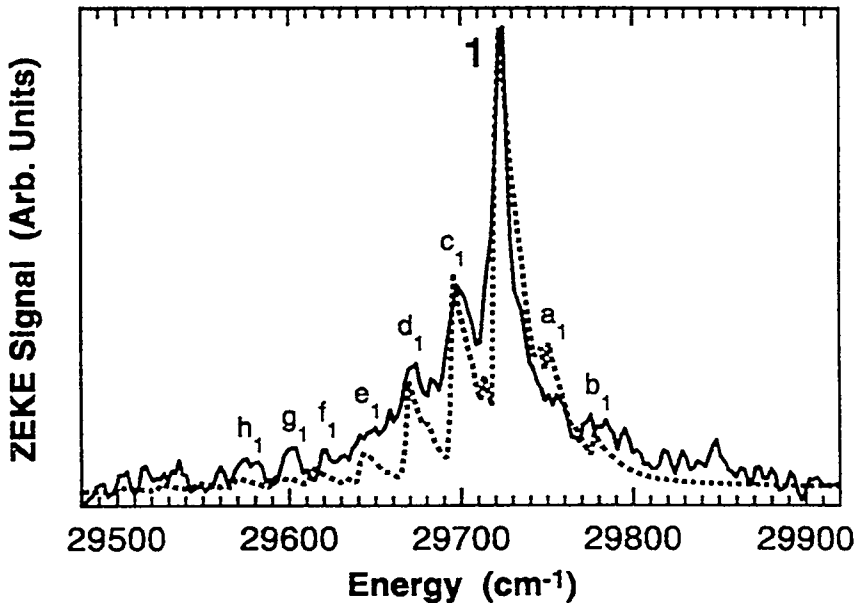

Figure 3.3. Experimental and simulated ZEKE spectrum of the $X\,\frac{1}{2}$ state (halogen atom $^2P_{3/2}$ asymptote) of $KrCl^-$. The solid line is the experimental spectrum, and the dotted line is the spectrum calculated from the model potentials described in the text.

Table 3.3. Peak assignments for KrCl ZEKE spectrum (Figure 3.3). Energies are in cm^{-1}. Assignments in parentheses contribute less than 20% of the total peak intensity. The assignment listed first contributes the most peak intensity.

State	Peak	Position	Relative Energy	Assignment
*X*1/2	**1**	29724.5	0	0←0 (2←1)
	a_1	29753.3	28.8	1←0 3←1
	b_1	29784.2	59.7	2←0
	c_1	29698.1	-26.4	1←1 (5←3) (3←2)
	d_1	29673.9	-50.6	2←2 (0←1)
	e_1	29645.4	-79.1	3←3
	f_1	29621.3	-103.2	4←4 0←2
	g_1	29601.6	-122.9	1←3 3←4
	h_1	29575.4	-149.1	2←4 4←5

3.4. Analysis

For each of the three species, the energy of peak **1** yields an accurate electron affinity: 27603(3) cm^{-1} for KrBr, 27890(3) for XeBr, and 29725(5) cm^{-1} for KrCl. These values are larger than the corresponding electron affinities of Br and Cl, which are 27129.170 cm^{-1} and 29138.3 cm^{-1}, respectively.[28,29] The larger electron affinities for the complexes show that the RgX^- dissociation energies are greater than the RgX dissociation energies, and that $XeBr^-$ is more strongly bound than $KrBr^-$. Also, from the vibrational assignments in Tables 3.1-3.3 we directly obtain vibrational frequencies for the anion and neutral states.

To gain further insight into these complexes, we construct model potentials for the anion and neutral RgX complexes to simulate the experimental ZEKE spectra. The spectra are fit by choosing model anion and neutral potentials, and calculating the Franck-Condon factors, which, assuming a Boltzmann distribution of anion vibrational population, are used to produce a simulated spectrum to be compared with the experimental spectrum. The potential parameters and vibrational temperature are then adjusted in a trial-and-error fashion to produce the best agreement between the experimental and simulated spectra. The vibrational eigenvalues are calculated from the potentials using a discrete variable representation procedure[30] based on a basis set of Morse potential eigenfunctions.[31] See Appendix A for the details of this calculation.

We use the flexible, piecewise Morse-Morse-switching function-van der Waals (MMSV) potential form. This is the same potential form used by Lee and co-workers for the RgX neutral potentials[15] and in our previous work.[3] For the neutral, this potential has the reduced form, with $f(x) = V(R)/\varepsilon$ and $x = R/R_m$:

$$\begin{aligned} f(x) &= e^{2\beta_1(1-x)} - 2e^{\beta_1(1-x)}, && 0 < x \le 1, \\ &= e^{2\beta_2(1-x)} - 2e^{\beta_2(1-x)} \equiv M_2(x), && 1 < x \le x_1, \\ &= SW(x)M_2(x) + [1 - SW(x)]W(x), && x_1 < x < x_2, \\ &= -C_{6r}x^{-6} - C_{8r}x^{-8} \equiv W(x), && x_2 \le x < \infty, \end{aligned} \tag{3.1}$$

where the switching function is given by

$$SW(x) = \frac{1}{2}\left[\cos\frac{\pi(x - x_1)}{(x_2 - x_1)} + 1\right], \tag{3.2}$$

and

$$C_{6r} = \frac{C_6}{\varepsilon R_m^6}, \qquad C_{8r} = \frac{C_8}{\varepsilon R_m^8} \tag{3.3}$$

Here, ε is the potential well depth and R_m is the bond length. C_6 is the induced dipole-induced dipole dispersion coefficient, and C_8 represents the induced dipole-induced quadrupole dispersion coefficient. Higher dispersion terms are neglected, as is the small induction term, varying as R^{-8}, arising from the halogen permanent quadrupole moment.

The anion potentials are of the same form, except that the dispersion terms are replaced by

$$f(x) = -B_{4r}x^{-4} - B_{6r}x^{-6} \equiv W(x), \qquad x_2 \le x < \infty \tag{3.4}$$

with

$$B_{4r} = \frac{B_4}{\varepsilon R_m^4}, \qquad B_{6r} = \frac{B_6}{\varepsilon R_m^6}. \tag{3.5}$$

and

$$B_4 = \tfrac{1}{2}q^2\alpha_d^{Rg}, \qquad B_6 = \tfrac{1}{2}q^2\alpha_q^{Rg} + C_6 \tag{3.6}$$

Here, q is the halide charge and B_4 is the coefficient of the dominant term in the long range RgX^- potential, reflecting the dipole induced in the rare gas atom by the halide charge. The B_6 term arises from quadrupole induction and dipole dispersion terms. α_d^{Rg} and α_q^{Rg} are the dipole and quadrupole polarizabilities of the rare gas, respectively.

The dispersion coefficients C_6 and C_8 are estimated using the formulas of Koutselos *et al.*[32] These formulas involve the dipole and quadrupole polarizabilities of each interacting atom, and an effective number of electrons, N, characteristic of each atom. N is empirically determined from the like-atom C_6 coefficients.[33-35] In the case of the halide atoms, the values of N are assumed to be the same as those of the isoelectronic rare gas atoms.

In calculating the dispersion coefficients for the neutral RgX complexes one must account for the open shell nature of the halogen atoms which results in anisotropic polarizabilities. The anisotropy of the dipole polarizability has been calculated for the Cl atom, neglecting spin-orbit effects, to be 14% relative to the average over all M_L states.[36] The halogen in the Σ state of an RgX complex, with the unpaired electron oriented along the internuclear axis, thus has a smaller polarizability and smaller dispersion interaction than in the Π state, where the unpaired electron is perpendicular to the axis. Bartolotti *et al.*[37] have calculated the anisotropy of the quadrupole polarizability of the Cl atom. However, in this calculation, the value given for the quadrupole polarizability of the Ar atom is 18% higher than the accurate value of Thakkar *et al.*[38] Therefore, the Cl quadrupole polarizabilities have been scaled down by this amount. This gives an anisotropy of 16% for α_q. Because the anisotropy of α_d of Br has not been calculated, it was assumed to be the same as that of Cl. Likewise, because calculations of α_q are not available for Br, these were estimated using the "hydrogenic relationship" discussed by Sastri *et al.*:[39]

$$\alpha_q \cong 1.570\alpha_d^{3/2} \tag{3.7}$$

The anisotropy of α_q was also assumed to be the same for Br as for Cl. To find the dispersion coefficients for states including spin-orbit effects, we note that the $I\frac{3}{2}$ state has pure Π character, while at long range the $X\frac{1}{2}$ state is a mixture of 2/3 Σ and 1/3 Π character, and the $II\frac{1}{2}$ state has 1/3 Σ and 2/3 Π character.[18] We assume that these mixing coefficients are constant for all regions of the potential where dispersion is the dominant interaction (i.e. $x > x_2$ in Eq. (1)).

The polarizabilities and effective numbers of electrons used here can be found in Table 3.4. The C_6 and C_8 coefficients for the various interactions are given with the other potential parameters, discussed below, in Tables 3.5-3.7. The C_6 values are fairly close to those of Lee and co-workers,[15] but the C_8 coefficients are in general larger because Lee and co-workers approximated C_8 with the values from the isoelectronic rare gas

pairs. Because the ZEKE spectra are not sensitive to the very long range part of the potential, B_4, B_6, C_6 and C_8 were kept fixed at the calculated values during the fitting procedures.

Table 3.4. Dipole and quadrupole polarizabilities and effective numbers of electrons used to calculate dispersion and induction coefficients. In atomic units.

Atom	Corresponding Neutral Spinless State	α_d	α_q	N
Cl	Σ	13.3[a]	72.0[b]	4.2[c]
	Π	15.3[a]	84.9[b]	4.2[c]
Br	Σ	18.7[d]	131[e]	6.2[c]
	Π	21.5[d]	154[e]	6.2[c]
Cl^-	---	28.1[c]	---	5.404[f]
Br^-	---	36.4[c]	---	6.309[f]
Kr	---	16.79[g]	99.296[h]	6.309[i]
Xe	---	27.16[g]	223.29[h]	7.253[i]

References for Table 3.4

(a) Ref. 36.

(b) Values from Ref. 37, scaled by a factor of 0.822 as explained in the text.

(c) Ref. 33.

(d) Derived from the spherically averaged value given in Ref. 33, assuming the same anisotropy for Br as for Cl.

(e) Calculated from α_d of Br using the "hydrogenic relationship" $\alpha_q \cong 1.570\alpha_d^{3/2}$ given in Ref. 39.

(f) Calculated from the C_6 values of the corresponding isoelectronic rare gases from Ref. 34, using the Slater-Kirkwood formula (see, for example, Ref. 35).

(g) Ref. 34.

(h) Ref. 38.

(i) Calculated from the C_6 values from Ref. 34.

Since the ZEKE spectra do not give information about the absolute values of ε and R_m, we have used results from previous experiments to guide our choice of these parameters. For KrBr, we fix R_m and ε of the $X\,\frac{1}{2}$ state at the values determined in the scattering experiments of Lee and co-workers.[15] For KrCl the values determined by Aquilanti and co-workers for the $X\,\frac{1}{2}$ states are used.[16] For XeBr, R_m and ε are taken from the $X\,\frac{1}{2}$ state potential of Tellinghuisen and coworkers,[13,14] which they obtained by combining their vibrationally resolved $B\,\frac{1}{2} \rightarrow X\,\frac{1}{2}$ emission spectrum with the repulsive wall of the potential of Lee and co-workers.[15] The emission spectrum independently provides a more precise well depth ε than could be determined from the scattering experiments alone: Clevinger and Tellinghuisen cite an uncertainty of 0.8% for their value of ε, significantly more precise than the uncertainty cited for the scattering results (~5%). However the uncertainty in R_m is essentially the same as in Lee's potential (~10%).

To determine ε for the anions and the remaining electronic states we then use the relationships implied by Figure 3.4, on the following page, namely:

$$\varepsilon_{an} = \nu_{00}(X\,\tfrac{1}{2}) + \omega_0^{an} + \varepsilon_X - \omega_0^{X} - EA \qquad (3.8)$$

$$\varepsilon_I = \varepsilon_X - \Delta_{X\text{-}I} - \omega_0^{X} + \omega_0^{I} \qquad (3.9)$$

$$\varepsilon_{II} = \varepsilon_X + \Delta_{so} - \Delta_{X\text{-}II} - \omega_0^{X} + \omega_0^{II} \qquad (3.10)$$

where $\nu_{00}(X\,\frac{1}{2})$ is the origin of the $X\,\frac{1}{2}$ state, ω_0^{an}, ω_0^{X}, etc. represent zero point energies, EA is the electron affinity of the halogen atom, $\Delta_{X\text{-}I}$ is the $X\,\frac{1}{2}$ - $I\,\frac{3}{2}$ state splitting (between υ=0 levels), and $\Delta_{X\text{-}II}$ is the $X\,\frac{1}{2}$ - $II\,\frac{1}{2}$ state splitting.

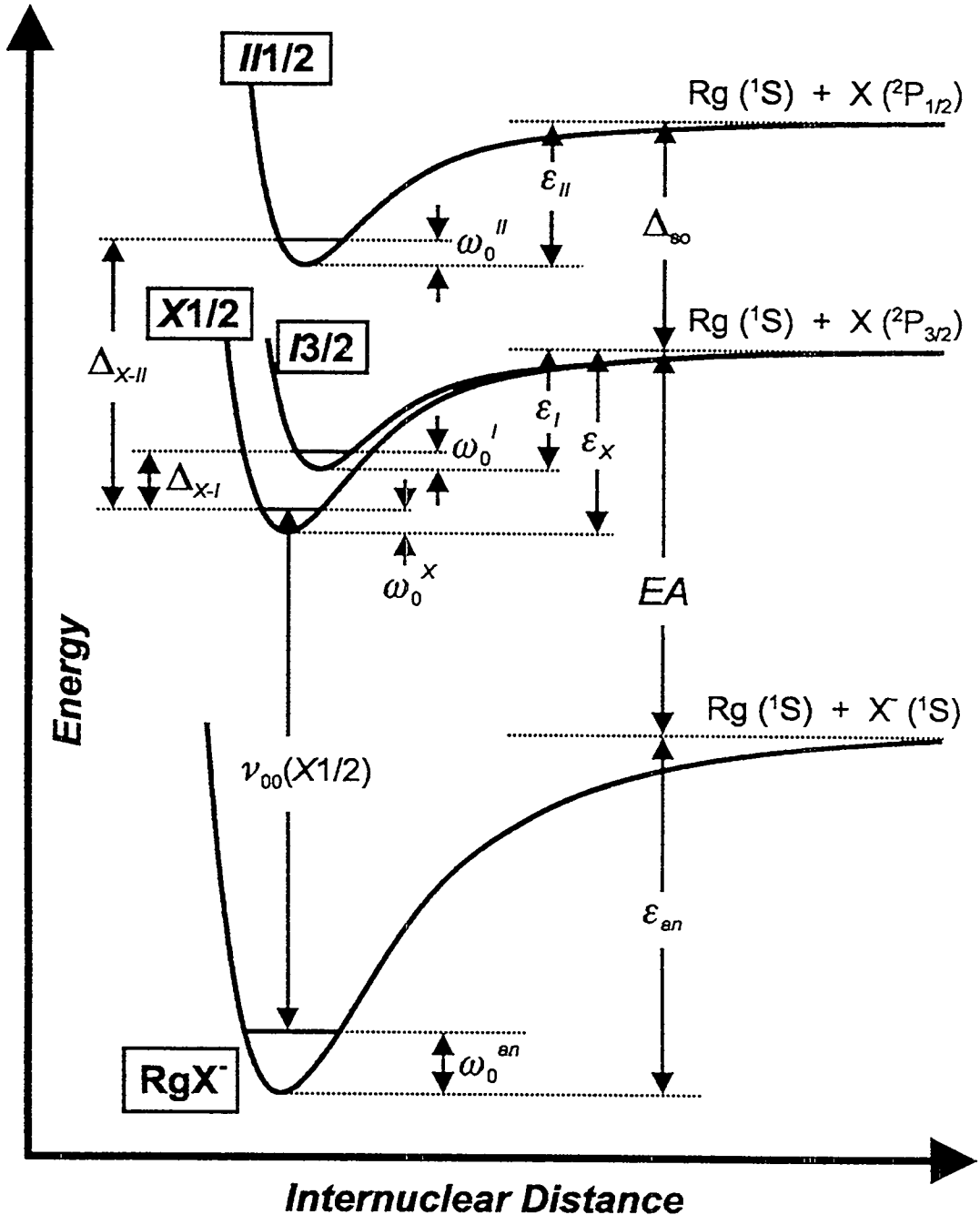

Figure 3.4. Schematic potential energy level diagram, showing the energetic relations among the atomic and molecular anion and neutral electronic states.

Once ε is fixed for the $I\frac{3}{2}$, $II\frac{1}{2}$ and anion states, R_m is found for these potentials by first adjusting R_m of the anion to best reproduce the observed peak intensities of the $X\frac{1}{2}$ state portion of the spectrum. When R_m is known for the anion, R_m for the $X\frac{1}{2}$ and $II\frac{1}{2}$ states can then also be found by means of the Franck-Condon simulation.

For KrBr the initial values of the $X\frac{1}{2}$ and $I\frac{3}{2}$ state potential parameters β_1, β_2, x_1 and x_2 were taken to be the same as Lee's values.[15] The β_1 parameter was kept fixed at the initial value because of the observation by Lee and co-workers that the slope of the repulsive part of the potential, with this β_1 values, agrees well with the slope determined from analysis of the excimer emission.[15] The remaining parameters, β_2, x_1 and x_2 were then adjusted to reproduce the peak spacings seen in the ZEKE spectra.

In the case of XeBr, the β_1, β_2, x_1 and x_2 parameters of the $X\frac{1}{2}$ state were adjusted to best fit the RKR turning points determined by Tellinghuisen and co-corkers.[14] With this potential form, it was possible to reproduce the RKR turning point energies to within 3.5 cm^{-1}. The vibrational spacings for the first nine levels of the resulting MMSV potential are within 0.2 cm^{-1} of those calculated from Tellinghuisen's spectroscopic constants, with the exception of the υ=0 to υ=1 spacing, which differed by 0.4 cm^{-1}. This level of agreement was judged to be sufficient for the purposes of this work. Because of the accuracy of the Tellinghuisen potential, no adjustments were made to the $X\frac{1}{2}$ state MMSV parameters during the fitting procedure.

For KrCl, the shape of the $X\frac{1}{2}$ state potential was estimated by choosing the MMSV parameters to reproduce the $X\frac{1}{2}$ state potential of Aquilanti and co-workers,[16] who used a different representation of the potentials. This was not modified during the fitting because of the absence of sufficient detail in the ZEKE spectrum. Therefore, in the KrCl simulation only the anion parameters are adjusted.

Once the potentials are established by the Franck-Condon fitting procedure, a rotational simulation is performed to fit the observed asymmetric peak shapes. In this

procedure, a set of rotational lines are calculated for each vibrational band, and these are convoluted with the asymmetric ZEKE instrumental line shape. The intensity of the ZEKE electron signal, $I(E)$, due to an individual line is represented by:

$$I(E) = \frac{a\left(\frac{E-E_0}{\Gamma}\right) + b\left(\frac{E-E_0}{\Gamma}\right)^3}{1 + c\left(\frac{E-E_0}{\Gamma}\right)^2 + d\left(\frac{E-E_0}{\Gamma}\right)^4}, \qquad E \geq E_0 \qquad (3.11)$$

$$= 0, \qquad E < E_0$$

with a = 4.3, b = 0.19, c = 4.2, and d = 2.3, and where E-E_0 is the energy above the threshold, E_0, of the line in cm^{-1}, and Γ is the full width at half maximum (FWHM) in cm^{-1}. The line shape parameters are obtained by a non-linear least squares fit to the ZEKE spectra of Br^-. This form differs from that used previously[3] and is a more accurate representation of the true ZEKE line shape. Readers are referred to our previous work[3] and to Appendix B for further details of the rotational fitting procedure. As in previous work, the rotational temperature was assumed to be 40 K.

The simulated spectra are shown as dotted lines superimposed on the experimental spectra in Figures 3.1-3.3. For the $KrBr^-$ and $XeBr^-$ spectra, the anion vibrational temperature for the $II\,\frac{1}{2}$ state differs slightly from the lower energy state(s) because the spectra were taken with different source conditions. The best-fit potential parameters are given in Tables 3.5-3.7, and the potentials are plotted in Figure 3.5, on the following pages.

Fitting the XeBr and KrCl spectra is fairly straightforward, because the peak assignments are readily apparent by inspection of the spectra. However, for KrBr the fitting procedure is used as an aid in assigning spectral features, since not all of the assignments are obvious from the spectra. Specifically, although the assignments of the $X\,\frac{1}{2}$ state features are straightforward, the location of the origin of the $I\,\frac{3}{2}$ state is not obvious upon initial inspection. As mentioned in Section 3.3.1, peak **2** of Figure 3.1(a) is assigned to the $I\,\frac{3}{2}$ origin because this allows the best fit with the model potential. Also,

this assignment gives a value of the $X\,\frac{1}{2}$ - $I\,\frac{3}{2}$ state splitting, $\Delta_{X\text{-}I}$, of 54.1 cm^{-1}. This is somewhat larger than $\Delta_{X\text{-}I}$ for ArBr, 38.2 cm^{-1},[3] a result expected due to the stronger Kr-Br interaction. On the other hand, if peak e_2 were chosen as the $I\,\frac{3}{2}$ state origin, $\Delta_{X\text{-}I}$ would essentially be the same as in ArBr, contrary to expectation. This further corroborates our choice of peak **2** as the $I\,\frac{3}{2}$ state origin.

The method for estimating the uncertainties of the potential parameters is discussed at length in our earlier work.[3] Here, we present these estimated uncertainties along with the potential parameters in Tables 3.5-3.7. The anion and neutral potentials are plotted in Figure 3.5. It should be remembered that the uncertainties in ε and R_m are expected to be fairly rigorous, whereas the uncertainties given for the other potential parameters represent lower bounds on the true uncertainties, because a complete multivariate analysis of the correlations amongst these parameters was not performed.

Table 3.5. MMSV potential parameters for KrBr and KrBr, and zero point energies (ω_0) and fundamental vibrational frequencies (ν_{01}) calculated from the potentials. Term values T_0 are referenced to anion ground vibrational state. Estimated uncertainties are given in parentheses.

	*X*1/2	*I*3/2	*II*1/2	Anion
T_{vib} (K) of anion	68. (4.)	68. (4.)	45. (3.)	---
T_0 (cm^{-1})	27602.9 (2.0)	27657.0 (4.0)	31321.7 (2.0)	0
ω_0 (cm^{-1})	12.5	11.7	12.4	19.2
ν_{01} (cm^{-1})	23.8	21.4	22.8	37.2
ε (meV)	19.9 (1.0)	13.1 (0.9)	15.7 (1.0)	79.5 (1.0)
R_m (Å)	3.90 (0.30)	4.15 (0.30)	4.03 (0.30)	3.85 (0.30)
β_1	5.70 (0.40)	7.20 (0.50)	7.00 (0.50)	4.62 (0.30)
β_2	6.72 (0.30)	8.00 (0.30)	7.20 (0.30)	4.62 (0.20)
x_1	1.02 (0.06)	1.05 (0.06)	1.05 (0.06)	1.04 (0.06)
x_2	1.70 (0.20)	1.65 (0.10)	1.85 (0.20)	1.50 (0.10)
C_6 ($eV \cdot Å^6$)	86.6 (13.0)	92.7 (14.0)	89.7 (14.0)	----
C_8 ($eV \cdot Å^8$)	740. (230.)	801. (250.)	770. (240.)	----
B_4 ($eV \cdot Å^4$)	----	----	----	17.91 (2.70)
B_6 ($eV \cdot Å^6$)	----	----	----	165. (41.)

Table 3.6. MMSV potential parameters for XeBr and $XeBr^-$, and zero point energies (ω_0) and fundamental vibrational frequencies (ν_{01}) calculated from the potentials. Term values T_0 are referenced to anion ground vibrational state. Estimated uncertainties are given in parentheses.

	***X*1/2**	***II*1/2**	**Anion**
T_{vib} (K) of anion	70. (4.)	90. (5.)	---
T_0 (cm^{-1})	27890.0 (2.0)	31623.6 (4.0)	0
ω_0 (cm^{-1})	12.3	12.6	21.3
ν_{01} (cm^{-1})	24.1	23.1	42.1
ε (meV)	31.53 (0.25)	25.52 (0.74)	126.92 (0.50)
R_m (Å)	3.82 (0.19)	4.00 (0.22)	3.81 (0.21)
β_1	4.35 (0.30)	6.42 (0.45)	3.50 (0.25)
β_2	7.41 (0.30)	7.00 (0.28)	5.30 (0.21)
x_1	1.01 (0.06)	1.03 (0.06)	1.03 (0.06)
x_2	2.00 (0.24)	1.60 (0.19)	1.60 (0.19)
C_6 (eV•$Å^6$)	128. (20.)	133. (21.)	----
C_8 (eV•$Å^8$)	1260. (400.)	1320. (410.)	----
B_4 (eV•$Å^4$)	----	----	28.98 (4.30)
B_6 (eV•$Å^6$)	----	----	271. (68.)

Table 3.7. MMSV potential parameters for KrCl and KrCl$^-$, and zero point energies (ω_0) and fundamental vibrational frequencies (ν_{01}) calculated from the potentials. Term values T_0 are referenced to anion ground vibrational state. Estimated uncertainties are given in parentheses.

	***X*1/2**	**Anion**
T_{vib} (K) of anion	210. (10.)	---
T_0 (cm^{-1})	29724.5 (2.0)	0
ω_0 (cm^{-1})	16.0	29.3
ν_{01} (cm^{-1})	29.9	55.5
ε (meV)	22.01 (1.00)	95.7 (1.0)
R_m (Å)	3.75 (0.10)	3.83 (0.10)
β_1	5.49 (0.40)	5.70 (0.50)
β_2	5.70 (0.20)	4.40 (0.20)
x_1	1.30 (0.08)	1.30 (0.06)
x_2	1.90 (0.20)	2.50 (0.20)
C_6 (eV•Å^6)	60.8 (9.1)	----
C_8 (eV•Å^8)	473. (71.)	----
B_4 (eV•Å^4)	----	17.91 (2.70)
B_6 (eV•Å^6)	----	138. (35.)

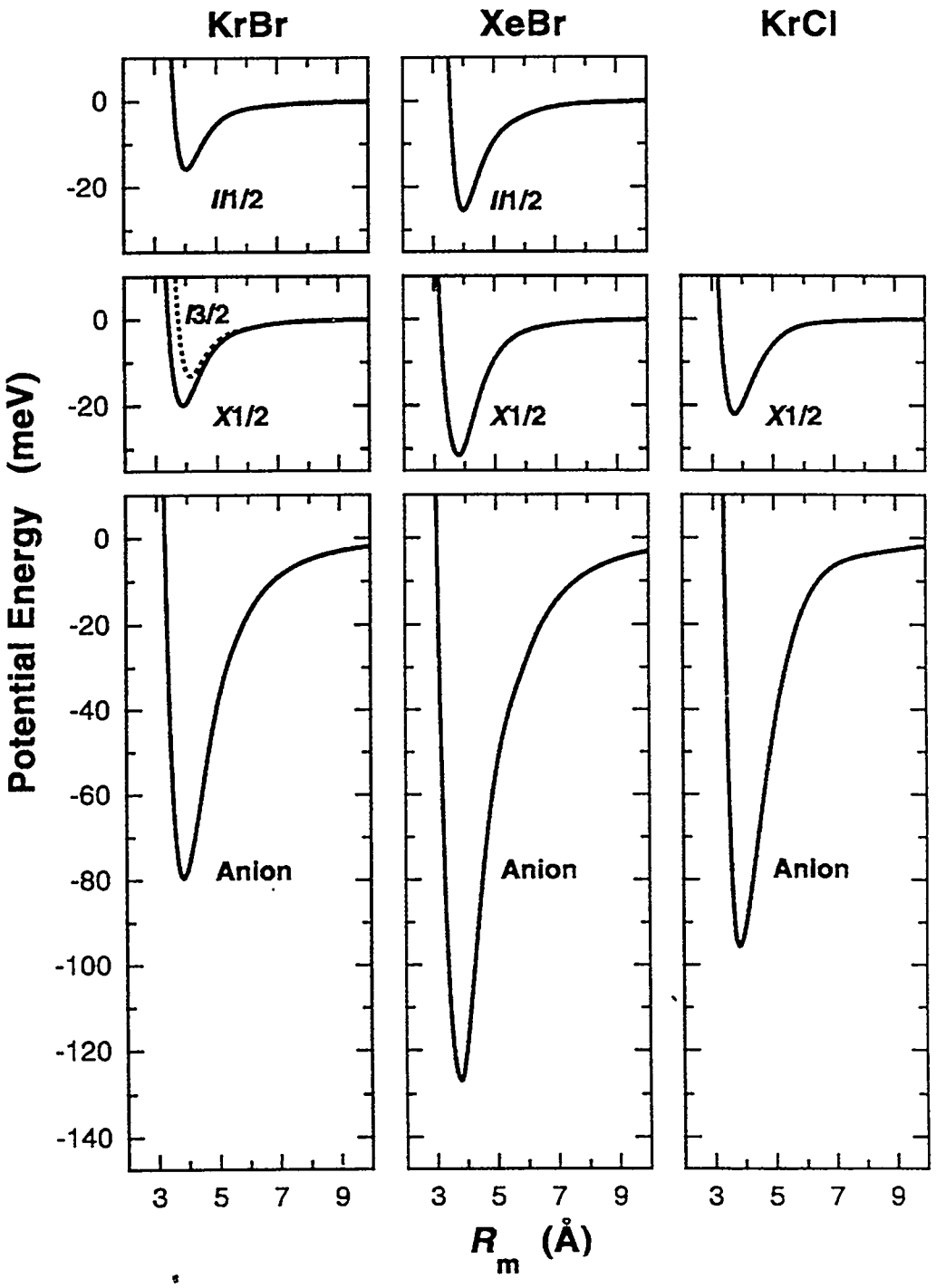

Figure 3.5. Plots of model potentials for the RgX anion and observed neutral states determined from the ZEKE spectra, using the MMSV potential parameters given in Tables 3.5-3.7.

3.5. Discussion

In this section we discuss our results for the neutral and anion RgX potentials and compare them to previously published potentials. The neutral potentials presented here do not differ greatly from earlier potentials. For the KrBr $X\frac{1}{2}$ state, of course, ε and R_m are the same as those from Lee's study,[15] since these were not adjusted in our fitting procedure. Our values of β_2, x_1 and x_2 differ somewhat from Lee's values and result in an improved match with the vibrational spacings in the ZEKE spectra. As mentioned above, β_1 was not adjusted, in order to retain agreement with the repulsive wall slopes from emission studies. For the $I\frac{3}{2}$ state of KrBr, the bond length is somewhat longer and the well depth a bit shallower than Lee's values. The difference in R_m values is well within the stated 10% uncertainty, but the difference in ε is just outside this range at 12%.

Our XeBr $X\frac{1}{2}$ state potential is more or less identical with that determined by Clevenger and Tellinghuisen,[14] differing only because of the limitations of the MMSV potential form, and was not varied during the fit because of the much higher relative accuracy of the emission results. The few features of the XeBr $X\frac{1}{2}$ state observed in the ZEKE spectrum (the progression $\upsilon' = 0, 1, 2 \leftarrow \upsilon'' = 0$, *i.e.*, peaks **1**, a_1 and b_1 in Figure 2) are consistent with the emission results within our experimental uncertainty.

Our $X\frac{1}{2}$ state potential for KrCl is essentially identical to the integral cross-section potential,[16] differing only in the choice of potential form. This is because the ZEKE spectra do not contain enough information to significantly improve on the potential obtained from the scattering experiments.

We obtain significantly more new information about the anion potentials. The trends in anion binding energies are similar to those seen in our previous study.[3] The larger binding energy for $XeBr^-$ compared to $KrBr^-$ is due to the larger polarizability of Xe, and the larger binding energy of $KrCl^-$ vs. $KrBr^-$ results from the smaller R_m and stronger charge-polarizability attraction in $KrCl^-$. For all three anions, the change in R_m upon photodetachment to the $X\frac{1}{2}$ state is very small, even though the anion binding

energy is considerably larger. Apparently the larger radius of the halide in the anions compensates for the stronger binding energy.

In Table 3.8, on the following page, we compare the potential parameters R_m and ε of the anions from the present study with other values from the literature, all of which have been derived through less direct means. It can be seen that the literature values are quite scattered. In comparing our results with the results of Kirkpatrick and Viehland,[6] who obtained potentials via direct inversion of ion mobility data, we find our well depths are systematically shallower and our bond lengths systematically longer. However, except for the $XeBr^-$ well depth, our values lie within or close to the 10% uncertainties cited by those authors. Similar discrepancies with other ZEKE potentials were explored in a recent paper by Kirkpatrick and Viehland, in which they used the ZEKE potentials[3] of ArI^- and $ArBr^-$ to simulate ion mobilities.[40] They found that the ZEKE ArI^- potential satisfactorily reproduced the mobility data, despite significant differences in R_m and ε from the potentials obtained by direct inversion of mobility data. However, the agreement for $ArBr^-$ was not as good. The authors cite the relative insensitivity of the mobility data to well depth to explain these findings.

Potentials derived from the earlier mobility results of McDaniel and co-workers[5] for $KrBr^-$ and $XeBr^-$ are slightly closer to ours, but show the same sign and order of magnitude deviations in ε and R_m. The electron gas calculations of Waldman and Gordon[8] again give systematically larger well depths than ours, although the bond lengths are in reasonable agreement.

Table 3.8. Comparison of ZEKE-determined anion potentials with literature potentials. Uncertainties are given in parentheses as reported in each work cited, if available.

	$KrBr^-$		$XeBr^-$		$KrCl^-$	
	ε (meV)	R_m (Å)	ε (meV)	R_m (Å)	ε (meV)	R_m (Å)
present work	79.5 (1.0)	3.85 (0.30)	126.92 (0.50)	3.81 (0.21)	95.7 (1.0)	3.83 (0.10)
ion mobilities[a]	87.1	3.73	145	3.62	---	---
ion mobilities[b]	88.7 (8.9)	3.579	169 (17)	3.397	102 (10)	3.448
electron gas[c]	93	3.76	159	3.64	---	---
electron gas[d]	---	---	---	---	115	3.48
empirical[e]	98	3.99	142	4.10	105	3.85
empirical[f]	90.1	3.91	130	4.02	95.4	3.79
semi-empirical[g]	92.5	3.70	167	3.62	107	3.55
semi-empirical[h]	75.1	3.87	99.9	4.05	---	---
semi-empirical[i]	85	3.79	146	3.74	108	3.53

References for Table 3.8

(a) Ref. 5.

(b) Ref. 6.

(c) Ref. 8.

(d) Ref. 7.

(e) Ref. 11.

(f) Empirical method of Ref. 11 modified as explained in the text.

(g) Ref. 32.

(h) Ref. 9.

(i) Ref. 10.

The results of Pirani and coworkers in Table 3.8 are obtained by very simple formulas based on empirical polarizability correlations.[11] Comparison to the ZEKE potentials show discrepancies greater than our experimental uncertainties, except for R_m of $KrBr^-$ and $KrCl^-$. Because of the simplicity of this method, and its usefulness for predicting new potentials, it is of interest to "recalibrate" these polarizability correlation formulae using the current and earlier ZEKE results. Fitting the ε and R_m parameters of the current study, and also those of the previous work on KrI^-, $ArBr^-$ and ArI^- we obtain

$$R_m = 1.725 \frac{\alpha_I^{1/3} + \alpha_B^{1/3}}{\left[\alpha_I \alpha_B \left(1 + \frac{1}{\rho}\right)\right]^{0.095}} \text{Å} \tag{3.12}$$

and

$$\varepsilon = 4380 \frac{\alpha_B}{R_m^{\,4}} (1 + \rho) \quad \text{meV} \tag{3.13}$$

with

$$\rho = \frac{\alpha_I \alpha_B}{\left[1 + (2\alpha_I / \alpha_B)^{1/3}\right] \alpha_B^{1/2}} \tag{3.14}$$

Here the anion and neutral polarizabilities, α_I and α_B, are in $Å^3$ and R_m is in Å. The numerical coefficients in Eqs. (3.12) and (3.13) differ somewhat from Pirani's values, 1.767 and 5200,[11] which were obtained using ε and R_m for the Li^+-He and Li^+-Ne interaction potentials as references. The results using our parameters are given in Table VIII; a significant improvement is obtained, although agreement is certainly not perfect. Eqs. (3.12) and (3.13) should be useful in predicting other halide-rare gas interactions; this will be tested in ongoing studies of similar species.

Finally, we should remark on the apparent absence of the $I\frac{3}{2}$ state in the $XeBr^-$ and $KrCl^-$ ZEKE spectra, and the much lower intensity of this state relative to the $X\frac{1}{2}$ state in the $KrBr^-$ ZEKE spectrum. Examination of these and our previous results on

ArI^-, $ArBr^-$, and KrI^- shows an overall trend in which the $I\frac{3}{2}$ transition is weaker for smaller halides and larger rare gas atoms. This may reflect variations in the transition moments, or perhaps the explanation lies in the differences between the s-wave partial detachment cross sections, since only those photoelectrons ejected with orbital angular momentum l=0 contribute to the ZEKE signal.[22]

3.6. Concluding Remarks

In this article, we have presented the ZEKE spectra of the RgX^- complexes $KrBr^-$, $XeBr^-$, and $KrCl^-$. We have obtained accurate electron affinities for these systems. Model anion and neutral potentials were constructed by Franck-Condon simulations of the spectra. In cases where comparison is possible, the neutral potentials are in reasonable agreement with the potentials from scattering experiments, with some minor adjustments in the well region for KrBr and XeBr. The anion potentials constructed from the data are, we believe, the most accurate experimental determinations available for these systems so far.

We have recently obtained results for Ar_nCl^- and Xe_nI^- clusters. Analysis of these spectra will yield further insight into the pair potentials and many-body interactions that govern bonding and structure in these species.

3.7. Acknowledgments

This research is supported by the Air Force Office of Scientific Research under Grant No. F49620-97-1-0018. Georg Reiser and Thomas Lenzer thank the Deutsche Forschungsgemeinschaft for postdoctoral fellowships.

3.8. References for Chapter 3

[1] G. C. Maitland, M. Rigby, E. B. Smith, and W. A. Wakeham, *Intermolecular Forces* (Oxford University Press, Oxford, 1981).

[2] R. A. Aziz, in *Inert Gases*, edited by M. L. Klein (Springer Verlag, 1984), pp. 5-86.

[3] Y. Zhao, I. Yourshaw, G. Reiser, C. C. Arnold, and D. M. Neumark, J. Chem. Phys. **101,** 6538 (1994).

[4] I. Yourshaw, Y. Zhao, and D. M. Neumark, J. Chem. Phys. **105,** 351 (1996).

[5] D. R. Lamm, R. D. Chelf, J. R. Twist, F. B. Holleman, M. G. Thackston, F. L. Eisele, W. M. Pope, I. R. Gatland, and E. W. McDaniel, J. Chem. Phys. **79,** 1965 (1983).

[6] C. C. Kirkpatrick and L. A. Viehland, Chem. Phys. **98,** 221 (1985).

[7] Y. S. Kim and R. G. Gordon, J. Chem. Phys. **61,** 1 (1974).

[8] M. Waldman and R. G. Gordon, J. Chem. Phys. **71,** 1325 (1979).

[9] J. W. Wilson, J. H. Heinbockel, and R. A. Outlaw, J. Chem. Phys. **89,** 929 (1988).

[10] S. H. Patil, J. Chem. Phys. **89,** 6357 (1988).

[11] D. Cappelletti, G. Liuti, and F. Pirani, Chem. Phys. Lett. **183,** 297 (1991).

[12] P. J. Hay and W. R. Wadt, Annu. Rev. Phys. Chem. **30,** 311 (1979).

[13] J. O. Clevenger and J. Tellinghuisen, Chem. Phys. Lett. **231,** 515 (1994).

[14] J. O. Clevenger and J. Tellinghuisen, J. Chem. Phys. **103,** 9611 (1995).

[15] P. Casavecchia, G. He, R. K. Sparks, and Y. T. Lee, J. Chem. Phys. **75,** 710 (1981).

[16] V. Aquilanti, D. Cappelletti, V. Lorent, E. Luzzatti, and F. Pirani, J. Phys. Chem. **97,** 2063 (1993).

[17] V. Aquilanti, G. Liuti, F. Pirani, and F. Vecchiocattivi, J. Chem. Soc. Faraday Trans. 2 **85,** 955 (1989).

[18] H. Haberland, Z. Phys. A **307,** 35 (1982).

[19] K. Müller-Dethlefs, M. Sander, and E. W. Schlag, Z. Naturforsch. Teil A **39,** 1089 (1984).

[20] K. Müller-Dethlefs, M. Sander, and E. W. Schlag, Chem. Phys. Lett. **12,** 291 (1984).

[21] K. Müller-Dethlefs and E. W. Schlag, Annu. Rev. Phys. Chem. **42,** 109 (1991).

[22] T. N. Kitsopoulos, I. M. Waller, J. G. Loeser, and D. M. Neumark, Chem. Phys. Lett. **159,** 300 (1989).

[23] C. J. Chick, Y. Zhao, T. N. Kitsopoulos, and D. M. Neumark, J. Chem. Phys. **97,** 6121 (1992).

[24] T. N. Kitsopoulos, Ph.D. Thesis, University of California, 1991.

[25] C. C. Arnold, Ph.D. Thesis, University of California, 1994.

[26] T. Lenzer, I. Yourshaw, M. Furlanetto, and D. M. Neumark, to be published .

[27] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes*, 2nd ed. (Cambridge University Press, Cambridge, 1992).

[28] C. Blondel, P. Cacciani, C. Delsart, and R. Trainham, Phys. Rev. A **40,** 3698 (1989).

[29] R. Trainham, G. D. Fletcher, and D. J. Larson, J. Phys. B **20,** L777 (1987).

[30] J. C. Light, I. P. Hamilton, and J. V. Lill, J. Chem. Phys. **82,** 1400 (1985).

[31] E. M. Greenawalt and A. S. Dickinson, J. Mol. Spectrosc. **30,** 427 (1969).

[32] A. D. Koutselos, E. A. Mason, and L. A. Viehland, J. Chem. Phys. **93,** 7125 (1990).

[33] E. A. Mason and E. W. McDaniel, *Transport Properties of Ions in Gases* (Wiley, New York, 1988).

[34] A. Kumar and W. J. Meath, Mol. Phys. **54,** 823 (1985).

[35] J. N. Wilson, J. Chem. Phys. **43,** 2564 (1965).

[36] K. Andersson and A. J. Sadlej, Phys. Rev. A **46,** 2356 (1992).

[37] L. J. Bartolotti, L. Ortiz, and Q. S. Xie, Int. J. Quant. Chem. **49,** 449 (1994).

[38] A. J. Thakkar, H. Hettema, and P. E. S. Wormer, J. Chem. Phys. **97,** 3252 (1992).

[39] M. V. K. Sastri, P. L. Narasimhulu, and K. D. Sen, J. Chem. Phys. **80,** 584 (1984).

[40] L. A. Viehland and C. C. Kirkpatrick, Chem. Phys. **202,** 285 (1996).

Chapter 4. Observation of many body effects in the zero electron kinetic energy (ZEKE) and threshold photodetachment spectra of Ar_nBr^- (n=2-9) and Ar_nI^- (n=2-19)*

Abstract

The anion zero electron kinetic energy (ZEKE) spectra of the van der Waals clusters $Ar_{2-3}Br^-$ and $Ar_{2-7}I^-$ have been measured, and partially discriminated threshold photodetachment (PDTP) experiments have been performed on $Ar_{4-9}Br^-$ and $Ar_{8-19}I^-$. Adiabatic electron affinities (EAs) have been determined from these results. The separation of the halogen $^2P_{3/2}$ asymptotic states was measured for the $Ar_{2-3}Br$ and $Ar_{2-7}I$ neutral clusters, and the separations between the $^2P_{1/2}$ and $^2P_{3/2}$ asymptotic states was determined for $Ar_{2-3}Br$ and $Ar_{2-3}I$. Model potentials were constructed, using the pair potentials determined from previous work on the diatomic rare gas-halide atom complexes, as well as various non-additive terms, and the cluster minimum energy structures were determined using a simulated annealing procedure. A simple first-order degenerate perturbation theory model of the neutral cluster potentials was found to agree well with the $^2P_{3/2}$ asymptotic electronic state separations observed in the ZEKE spectra. The halogen spin-orbit splittings in the $Ar_{2-3}Br$ and $Ar_{2-3}I$ clusters were found to be slightly smaller than those of the free halogen atoms. The binding energies calculated from a model additive potentials were found to be inconsistent with the experimental electron affinities. Model potentials including many-body induction effects, three-body "exchange

---

* Originally published in slightly different form in J. Chem. Phys. **105**, (1996), with co-authors Yuexing Zhao and Daniel M. Neumark.

quadrupole" effects and a triple-dipole dispersion term were found to agree well with the experimental results. Many-body induction was found to be the dominant non-additive effect. The exchange quadrupole effect--i.e., the interaction of the exchange induced electron charge distribution distortion among argon atoms with the halide charge-- was also found to be important.

4.1. Introduction

In most studies of weakly interacting atoms or molecules, pairwise additivity of the potentials is assumed. Given pair potentials, V_{ij}, between atoms i and j, the pairwise additive approximation to the total potential of N interacting atoms is

$$V_{pair} = \sum_{i<j}^{N} V_{ij}\left(\left|\mathbf{r}_i - \mathbf{r}_j\right|\right) \tag{4.1}$$

Here $\mathbf{r}_i$ and $\mathbf{r}_j$ represent the positions of atoms i and j. If the atoms had closed valence shells, and if no deformation of the atomic charge distributions were induced by the interactions, then pairwise additivity would hold exactly.[1] However, if the deformation of the charge distributions due to the interatomic interactions (e.g. dispersion, induction or exchange) is considered, the assumption of pairwise additivity breaks down.[1] This assumption can also break down if one of the atoms has an open valence shell. Then it is necessary to consider the electronic states of the open-shell atom which arise from the simultaneous presence of all the other atoms; the potential energy surfaces of these states cannot, in general, be obtained by simply adding the pair potentials in the sense of Equation (4.1). In either case it is necessary to extend Equation (4.1) to include non-additive, or many-body, effects:

$$\begin{aligned} V_{many-body} &= V_{pair} + V_{non-add} \\ &= V_{pair} + \sum_{i<j<k}^{N} V_{ijk}\left(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k\right) + \ldots + \sum_{i<j<k\cdots z}^{N} V_{ijk\cdots z}\left(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k \cdots \mathbf{r}_z\right). \end{aligned} \tag{4.2}$$

Non-additive effects are believed to play a significant role in determining the properties of bulk matter. For example, the binding energies of rare gas solids (Ne, Ar, Kr and Xe) measured experimentally are about 7-10% smaller than the binding energies calculated from accurate pair potentials.[2] However, there has been some controversy

about the precise nature of the non-additive effects involved.[2,3] Furthermore, it is in general difficult to extract detailed information about non-additive effects from measurements of bulk properties.[3]

Cluster studies represent an alternative approach to learning about non-additive effects. By probing the spectroscopy and/or energetics of a cluster as a function of its size, and comparing the results with predictions based on additive forces alone, one can obtain considerable insight into the various non-additive components of the interaction potential.[3] To this end, we present in this paper the anion zero electron kinetic energy (ZEKE) spectra of the $Ar_{2-3}Br^-$ and $Ar_{2-7}I^-$ van der Waals clusters, and partially discriminated threshold photodetachment (PDTP) spectra of $Ar_{4-9}Br^-$ and $Ar_{8-19}I^-$. We also present the results of calculations with model potentials involving various non-additive terms, in an effort to understand the experimentally observed electron affinities (EAs) and electronic structure. Our results probe non-additive effects in both the cluster anion and the open-shell neutral cluster resulting from photodetachment. This work is an extension of our previous ZEKE study of the diatomic rare gas-halide atom complexes,[4] and previous ZEKE work on the I^-CO_2 complex.[5]

In order to extract information on many-body effects from experimental studies of clusters, the pair potentials must be known more accurately than the magnitude of the many-body effect. Furthermore, the experiment must provide information about the "true" potential that can be compared with the results of calculations with model additive and non-additive potentials. This information may consist of spectroscopically measured vibrational frequencies, rotational constants, etc. In this case, accurate dynamical calculations are needed to extract this information from the model potentials for

comparison with experiment. Alternately, some experiments allow a more direct measurement of the cluster binding energies, in which case comparison with model potentials is much more straightforward.

Non-additive effects can affect the rotational, vibrational, and electronic spectroscopy of a cluster. Much of the recent interest in this field has focused on high resolution spectroscopy of van der Waals clusters. For example, pure rotation spectra of Ne_2Kr and Ne_2Xe have been observed using fourier transform microwave spectroscopy.[6] The structural information and nuclear hyperfine coupling constants determined from these spectra show evidence of non-additivity. There have also been a number of near and far infrared studies of molecular chromophores in rare gas clusters.[3,7,8] In many cases it is difficult to extract meaningful information about many-body forces from spectroscopic studies because the intermolecular pair potentials are often not well enough characterized, in that the uncertainty in the pair potentials is comparable to the magnitude of the many-body effects. There has, however, been recent experimental and theoretical progress in determining intermolecular pair potentials accurately enough to learn about three body interactions in the Ar_2HCl,[3,8,9,10a] Ar_2HF,[7,9,10c] and Ar_2DCl[3,10b] systems. In work more closely related to the results presented here, the electronic spectroscopy of $Ar_{1-4}Hg$ has been studied with multi-photon ionization,[11] and Ar_nBa clusters have been studied by laser induced fluorescence.[12] Only the $Ar_{1-4}Hg$ study was mass-selective. From this work, progress has been made in identifying "non-additive" effects in the excited electronic state of these clusters with open-shell chromophores.[11,12]

It is challenging to extract information on non-additive effects from direct spectroscopic measurements such as those mentioned above. Even when mass selectivity

can be obtained, non-trivial dynamical calculations are needed to extract the vibrational and rotational structure information from a many-body model potential in order to compare it with the experimental spectrum. It is desirable, therefore, to experimentally measure the binding energies (*BE*s) of clusters, because *BE*s can be readily obtained from many-body model potentials by simple methods. Moreover, there is generally an intuitive connection between a particular non-additive term and the cluster binding energy, in the sense that one can usually predict by inspection if the binding energy will increase or decrease when a given many-body term is added to a model potential. However, in most cases, *BE*s of clusters cannot be directly obtained from experimental spectra. Exceptions include the pump-probe experiments of Janda and coworkers on $Ar_{2-3}Cl_2$[13a] and $HeBr_2$,[13b] and the stimulated emission pumping experiments on the carbazole-Ar system by Leutwyler and coworkers.[14]

Anion photoelectron spectroscopy (PES) of clusters has proved useful in providing more direct information about the relative binding energies of anion and neutral clusters. It also has the advantage of mass selectivity. Examples include the work of Markovich *et al.* on $X^-(H_2O)_n$ (X^- = Cl^-, Br^- and I^-),[15] Bowen and coworkers on O^-Ar_n,[16] and Arnold *et al.* on $X^-(CO_2)_n$ and $X^-(N_2O)_n$.[17] The theoretical calculations of Berkowitz *et al.*[18] in conjunction with the PES spectra of Markovich *et al.* [15] have demonstrated the importance of non-additive inductive effects in $Br^-(H_2O)_n$ clusters.

However, there are two problems with trying to extract information on non-additive forces from these studies. First, the pair potentials for the relevant neutral and ionic species are not in general known very accurately; this is particularly true for clusters involving molecular solvents. Secondly, the resolution of conventional anion PES is

typically in the range of 80 cm^{-1} (Ref. 17)- 400 cm^{-1} (Ref. 15), depending on the type of energy analyzer used. Because of this limited resolution, anion PES experiments can only be sensitive to the largest non-additive effects, such as inductive effects in the anion clusters.

The anion ZEKE technique used in the present experiments on Ar_nBr^- and Ar_nI^- combines the advantage of mass-selectivity with much higher resolution (*ca.* 2-3 cm^{-1} for atomic systems) than PES experiments. This resolution allows accurate measurement of electron affinities, as well as spectroscopic observation of the electronic structure of the neutral Ar_nX clusters. The $Ar\text{-}X^-$ and Ar-X pair potentials are known accurately from our previous work on the diatomic species.[4] Thus, by employing simulated annealing procedures to determine the binding energies and neutral electronic structure from model potentials, we can directly compare our experimental results with the pairwise additive predictions, and explore the effects of various many-body corrections to the additive potentials. From this comparison, we can obtain a detailed picture of non-additive effects in Ar_nX^- and Ar_nX clusters.

This paper is organized as follows. In Section 4.2 we briefly describe the experimental apparatus and techniques. In Section 4.3, we present the anion ZEKE and PDTP spectra, determine the experimental *EA*s, assign the electronic structure observed in the ZEKE spectra, and briefly discuss the observed vibrational structure. In Section 4.4, we describe the methods and present the results of calculations of the cluster *EA*s and neutral electronic structure from model additive and non-additive potentials, and compare them with the experimental results. In Section 4.5, we summarize, considering what we

can and cannot conclude about many-body interactions on the basis of our results, and suggest future avenues for experimental and theoretical research.

4.2. Experiment

Zero electron kinetic energy (ZEKE) spectroscopy was first developed for photoionization of neutrals by Müller-Dethlefs *et al.*[19] and applied to negative ion photodetachment by Neumark and co-workers.[20] The experimental apparatus has been described in detail elsewhere.[20] Briefly, Ar_nX^- clusters are produced by expanding a mixture of approximately 0.1-0.5% Freon (CF_3I or CF_2ClBr, PCR Co.) in a *ca.* 25% argon/ 75% helium mixture through a pulsed valve (General Valve Series 9) with a 0.5 mm diameter orifice. Backing pressures are typically 60-80 psi. The expansion is crossed with a 1 keV electron beam. Halide anions are formed by dissociative attachment of low energy secondary electrons and undergo clustering in the continuum flow region of the free expansion. The molecular beam is collimated with a skimmer, accelerated to 1 keV, and mass-selected with a 1 m long collinear time-of-flight mass spectrometer.[20c,21] The mass selected ions then enter a differentially pumped detection region, and are irradiated with a pulse from an excimer pumped dye laser (Lambda Physik). For the ground states of Ar_nI, BBQ, PBBO, Exalite 398, QUI and DMQ laser dyes (Exciton) were used. For the $Ar_{2-3}I$ excited state scans rhodamine 610 dye was frequency doubled with a KDP crystal. For the Ar_nBr clusters, DMQ and PTP dyes were used for the ground states; rhodamine 640 was doubled with a KDP crystal for the excited state $Ar_{2-3}Br$ spectra. The power of the undoubled light was typically 7-20 mJ per pulse at the interaction region. The frequency doubled laser power was about 2 mJ per pulse. The laser wavelength was

calibrated from 337-400 nm with the Ne lines observed by the optogalvanic effect in a Fe-Ne hollow cathode lamp. The fundamental wavelength of the frequency doubled light was calibrated in the region 600-640 nm with an iodine absorption cell.

Two modes of electron detection were used in the present studies: the high resolution ZEKE mode, and the lower resolution partially discriminated threshold photodetachment (PDTP) mode. In the ZEKE mode, the photodetached electrons are extracted collinearly by a weak (2-5 V/cm) electric field after a 300-500 ns delay, and deflected to an off-axis microchannel detector. Detection is gated to provide temporal filtering. A series of apertures between the detachment point and detector provide spatial discrimination. This combination of spatial and temporal filtering discriminates against high energy electrons, so that as the laser wavelength is scanned, only photoelectrons with nearly zero kinetic energy are detected. The resolution of the instrument is about 2-3 cm^{-1} for atomic systems.[20] However, in the spectra of molecules, the peaks are broadened by unresolved rotational structure. For the systems studied in this paper, the observed peaks were at least 8 cm^{-1} wide (FWHM).

In the PDTP mode[20b], there is no delay between the laser pulse and electron extraction, retaining only spatial filtering as in the "steradiancy detector" first described by Baer *et al.*[22] This results in some discrimination against electrons with energies greater than about 150 cm^{-1}, and leads to peaks about 200 cm^{-1} wide in the present case. However, the thresholds, and hence the electron affinities, can be determined more accurately than this, to within approximately ± 50 cm^{-1}. Because nearly all of the electrons are collected, this mode of operation has the advantage of much higher sensitivity than the ZEKE mode.

The ZEKE spectra were averaged over several thousand laser shots per point taken in several separate scans. The PDTP spectra were averaged over 300-1000 laser shots per point. All spectra were normalized to the ion signal and laser power.

No obvious "magic numbers" were seen in the mass spectra. The ion signal smoothly decreased in intensity with increasing cluster size in the mass spectra of both the Ar_nBr^- and Ar_nI^- clusters.

4.3. Results

4.3.1. $Ar_{2-3}I^-$, $Ar_{2-3}Br^-$

The ZEKE spectra of Ar_2Br^- and Ar_3Br^- are shown in Figure 4.1, along with the spectrum of the diatomic $ArBr^-$ complex, reproduced from Reference 4. ZEKE spectra of ArI^-, Ar_2I^- and Ar_3I^- are displayed in Figure 4.2. All the spectra have two sets of features, separated by approximately the spin orbit splitting of the halogen atoms: 3685 cm^{-1} for Br and 7603.15 cm^{-1} for I.[23] We assign the lower energy set of features to electronic states arising from the ground $^2P_{3/2}$ state of the halogen atom, and the higher energy features to $^2P_{1/2}$ asymptotic states. The ground state manifolds of the $Ar_{2-3}I$ clusters are dominated by two sharp, intense peaks, labeled *X* and *I*, separated by about 40-65 cm^{-1}. In the $Ar_{2-3}Br$ spectra, both features are also present, but peak *I* is less intense and distinct than in the Ar_nI spectra.

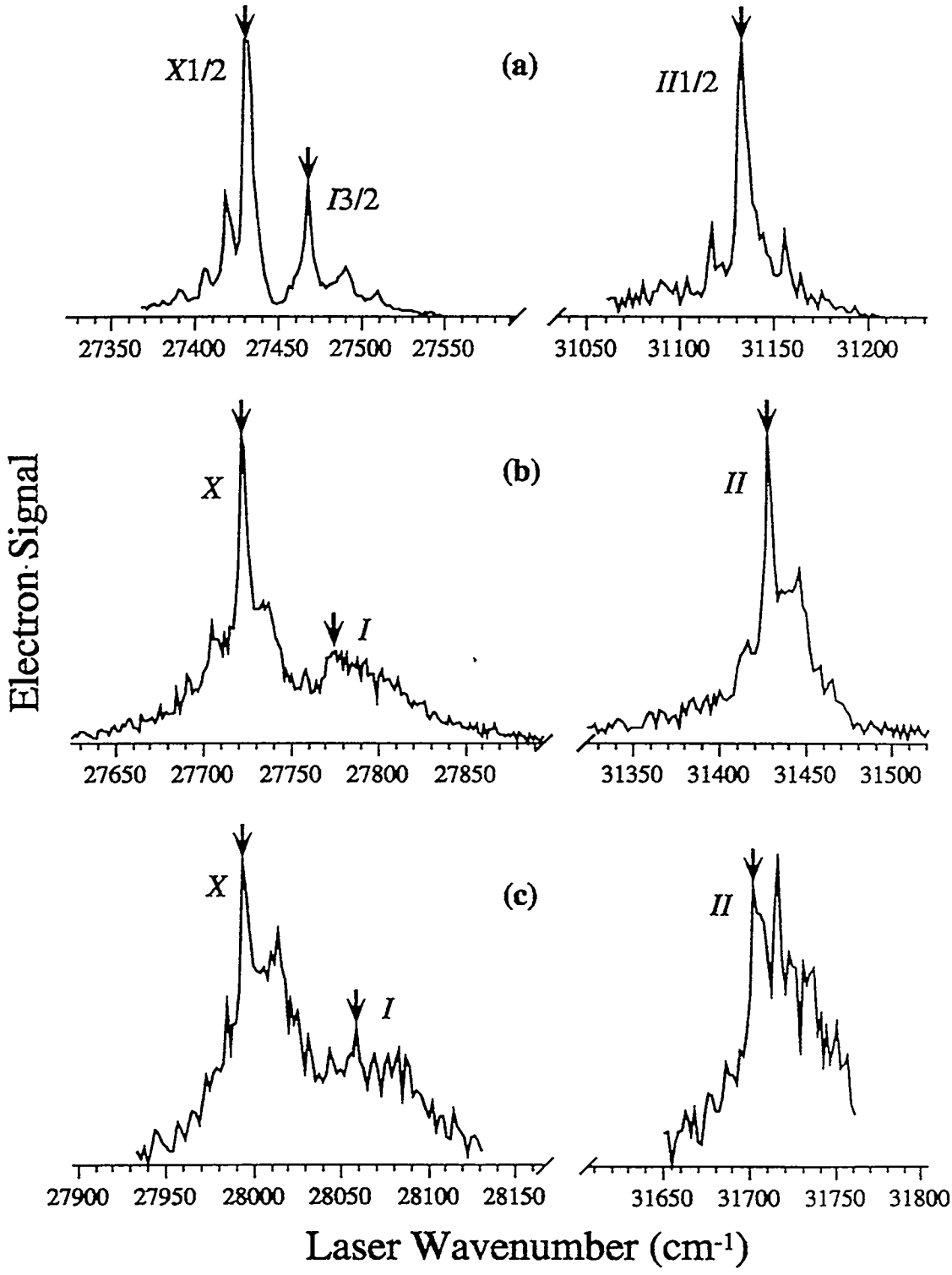

Figure 4.1. Zero electron kinetic energy (ZEKE) spectra of (a) $ArBr^-$, (b) Ar_2Br^-, and (c) Ar_3Br^-. The arrows indicate the neutral electronic state origins.

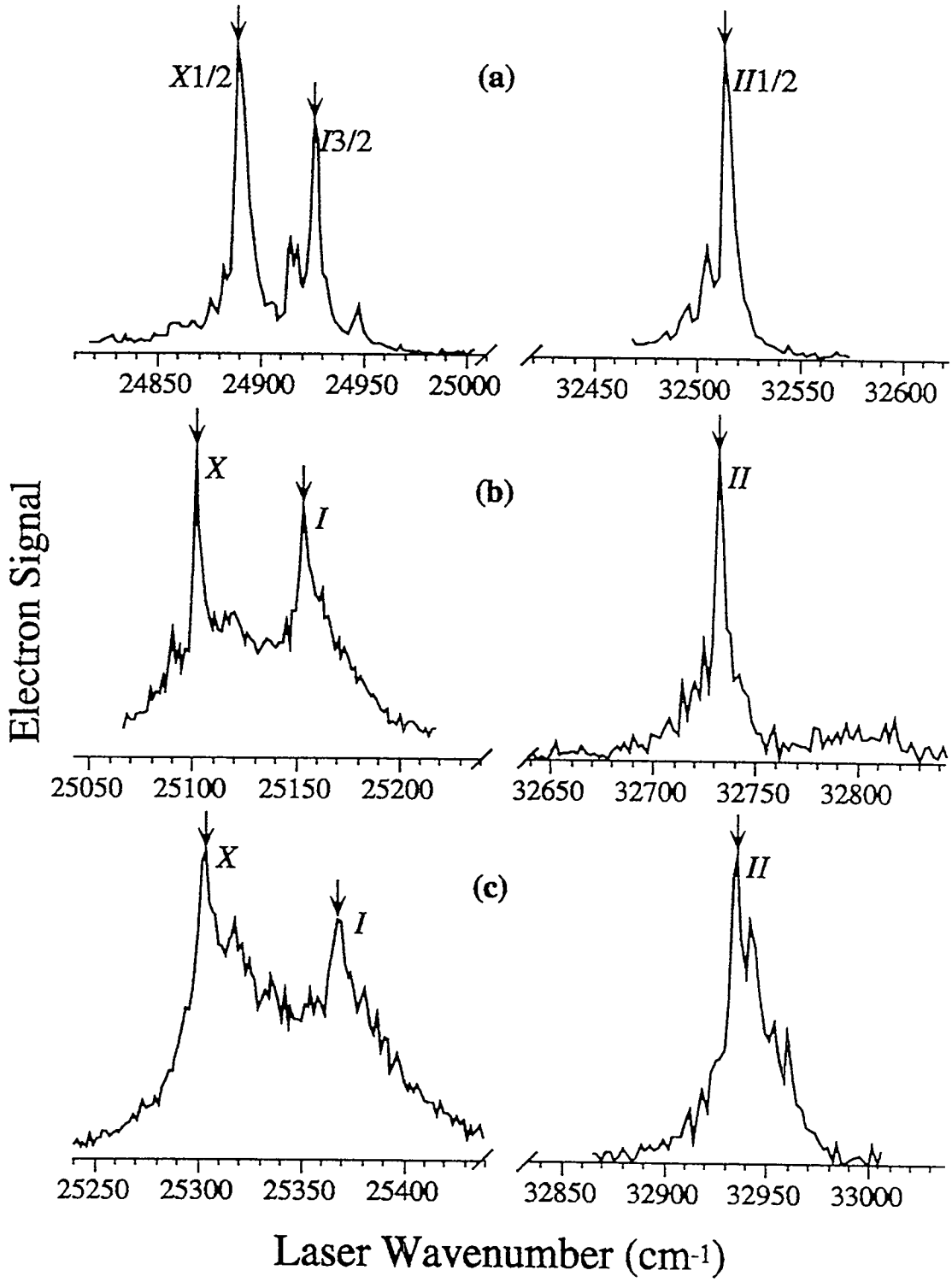

Figure 4.2. ZEKE spectra of (a) ArI^-, (b) Ar_2I^-, and (c) Ar_3I^-. The arrows indicate the neutral electronic state origins.

In the previous work on the diatomic species[4] the corresponding features were assigned to the origins of the two electronic states that correlate to the halogen $^2P_{3/2}$ asymptote, referred to as the $X\frac{1}{2}$ ($j_a = \frac{1}{2}$, $\Omega = \frac{1}{2}$) and $I\frac{3}{2}$ ($j_a = \frac{3}{2}$, $\Omega = \frac{3}{2}$) states, in Hund's case (c) notation.[24] The feature labeled $II\frac{1}{2}$ in the diatomic spectra was assigned to the origin of the $II\frac{1}{2}$ state ($j_a = \frac{3}{2}$, $\Omega = \frac{1}{2}$), which correlates to the halogen $^2P_{1/2}$ asymptote. We expect an analogous set of three doubly degenerate electronic states to be present in the polyatomic clusters. The lower $^2P_{3/2}$ halogen state is split into two doubly degenerate states by the weak interaction with the argon atoms. We refer to these states as the *X* and *I* states, by analogy with the diatomic case, dropping the Ω designation, as this is no longer a good quantum number in the polyatomic case. Note that here the "*X* state" always refers to the lowest energy state at the equilibrium geometry regardless of the symmetry of the cluster.

The *X* and *II* state origins are blue shifted relative to the corresponding atomic lines by several hundred cm^{-1}. The blue shift increases as the number of argon atoms increases. This demonstrates that the anionic clusters are more strongly bound than the neutral species.

In the Ar_2Br^- spectrum [Figure 4.1(b)] we see some partially resolved peaks to the red and blue of the *X* state origin. There is also a long "tail" to the blue of the *I* state origin. We attribute all these features to transitions to or from vibrationally excited states. Based on our previous interpretation of the diatomic $ArBr^-$ spectrum,[4] it is likely that the features to the red of the *X* state origin are due to hot-band or sequence band transitions from vibrationally excited anion states. Likewise, the features to the blue of the *X* state

origin may be transitions to vibrationally excited neutral ground states and/or hot-band transitions to the *I* state. The vibrational progressions are not as well resolved as in the diatomic spectra. The observed structure is probably due to many overlapping transitions involving more than one vibrational mode. This spectral congestion appears to be more severe to the blue of the *I* state origin, possibly indicating a larger geometry change between the *I* state and the anion than between the *X* state and the anion, as was seen in $ArBr^-$.[4] The peaks to the red and blue of the *II* state origin may similarly be understood as sequence band or hot-band transitions, and transitions to vibrationally excited neutral states, respectively.

The spectrum of Ar_3Br^- [Figure 4.1(c)] appears even more congested. There is a distinct peak 20 cm^{-1} to the blue of the *X* state origin, in addition to numerous poorly resolved features. Again there appears to be an extended unresolved progression to the blue of the *I* state origin. The *II* state has two prominent peaks, separated by 14 cm^{-1}, plus some other indistinct peaks to the blue. It is not clear which of the two peaks is in fact the *II* state origin.

In the case of Ar_2I^- [Figure 4.2(b)], the vibrational structure is somewhat less well resolved than in Ar_2Br^-. There is a clear feature 11 cm^{-1} to the red of the *X* state origin, as well as some poorly resolved structure between the origins of the *X* and *I* states. There is a tail to the blue of the *I* state origin. The spectrum of the *II* state is rather sparse, with some peaks 10-30 cm^{-1} to the red of the origin due to sequence or hot bands, and a slight shoulder to the blue. The lack of any extended progression indicates that the anion-*II* state transition is quite vertical.

The spectrum of Ar_3I^- [Figure 4.2(c)] shows clearer vibrational resolution than Ar_2I^-. There are two peaks spaced by 8 and 32 cm^{-1} from the *X* state origin. However, the sequence band structure to the red of the origin is not resolved. The *II* state of Ar_3I displays three distinct peaks 7, 17 and 24 cm^{-1} to the blue of the origin, as well as some less distinct sequence band structure to the red.

The partially resolved vibrational structure seen in these spectra is of considerble interest and will be considered further in future publications. In this paper, we are primarily concerned with the accurate electron affinities and state splittings yielded by these spectra.

4.3.2. $Ar_{4-7}I^-$

The ZEKE spectra of the $Ar_{4-7}I^-$ clusters are shown in Figure 3. For these clusters we studied only the lower ($^2P_{3/2}$ asymptotic) states. In the Ar_4I spectrum [Figure 4.3(a)] the origins of the *X* and *I* electronic states are distinct. In the Ar_5I^- spectrum [Figure 4.3(b)] the peak corresponding to the *X* state origin is quite broad, and the *I* state appears as an unresolved shoulder. The *I* state also appears relatively less intense than in the $Ar_{1-4}I$ spectra. Based on the profile of this shoulder, we can only estimate the position of the *I* state origin to ±20 cm^{-1}.

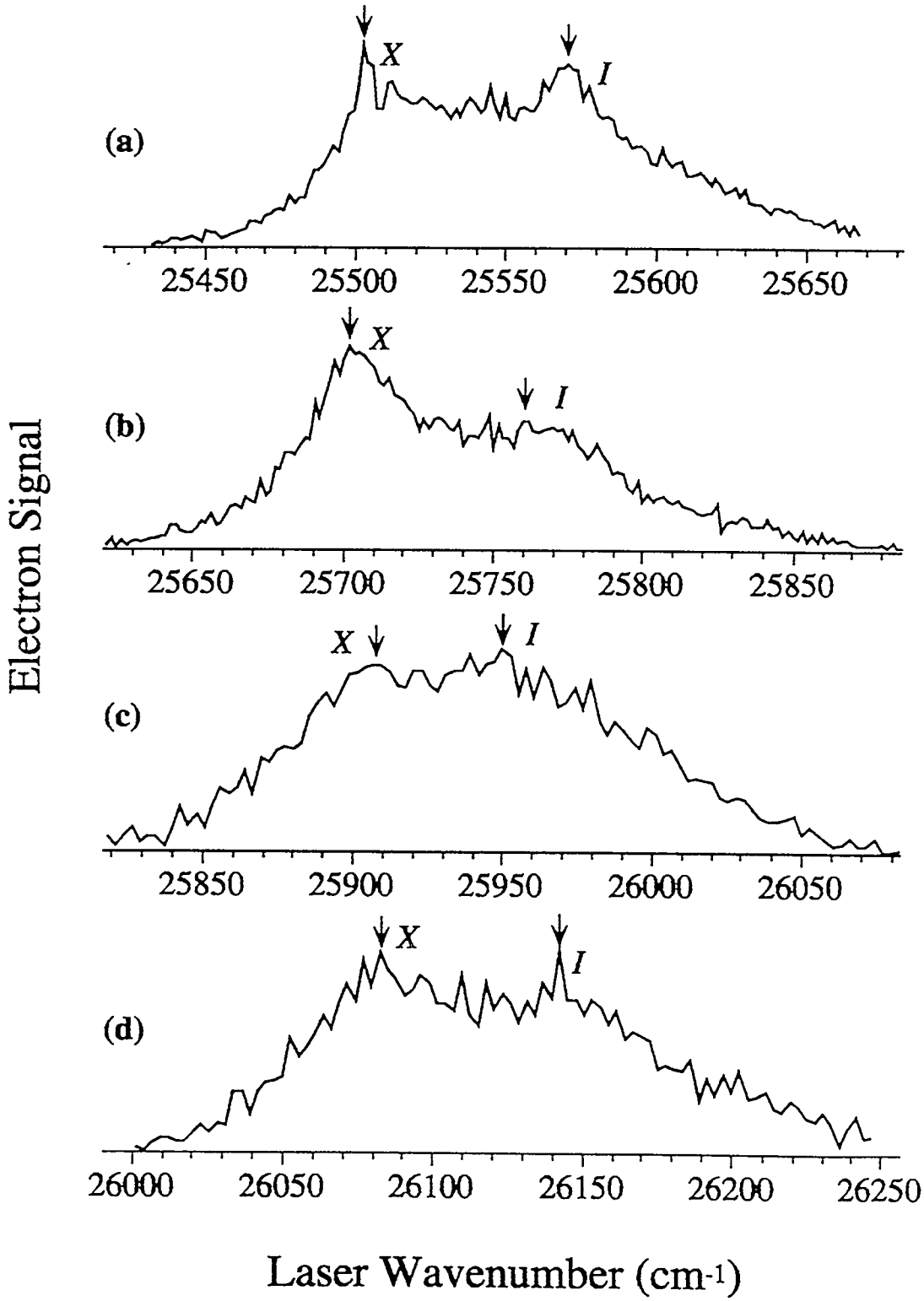

Figure 4.3. ZEKE spectra of (a) Ar_4I^-, (b) Ar_5I^-, (c) Ar_6I^-, and (d) Ar_7I^-. The arrows indicate the neutral electronic state origins.

The spectra of Ar_6I and Ar_7I [Figures 4.3(c) and 4.3(d)] are more congested. The positions of the *X* and *I* state origins can be estimated, as indicated by the arrows in the figures, but it is not possible to discern any reproducible vibrational structure. As we go from Ar_4I to Ar_5I, the separation between the *X* and *I* states appears to decrease. Although the exact splitting is difficult to discern from the Ar_6I^- spectrum, it appears in this case that the *X-I* splitting again decreases somewhat from Ar_5I. However, the splitting appears to increase again in the Ar_7I spectrum, in which the two states are better resolved than in the Ar_6I spectrum.

We attempted to observe ZEKE spectra of Ar_8I^- and larger clusters, but obtained only unstructured spectra with no reproducible features.

4.3.3. Partially Discriminated Threshold Photodetachment Spectra

Because of the increasing spectral congestion with increasing cluster size and the difficulty of producing sufficient quantities of large clusters with our source, it was not possible to perform the ZEKE experiment on clusters with n>7 in the case of Ar_nI^-, and n>3 for Ar_nBr^-. In the PDTP mode of operation it is possible to work with much smaller quantities of anions, because nearly all of the photoelectrons near the detachment threshold are collected. Therefore, only the PDTP experiment was performed for $Ar_{4-9}Br^-$ and $Ar_{8-19}I^-$.

The PDTP spectra of $Ar_{4-9}Br^-$ are shown in Figure 4.4, and those of $Ar_{8-19}I^-$ in Figure 4.5 on the following pages.

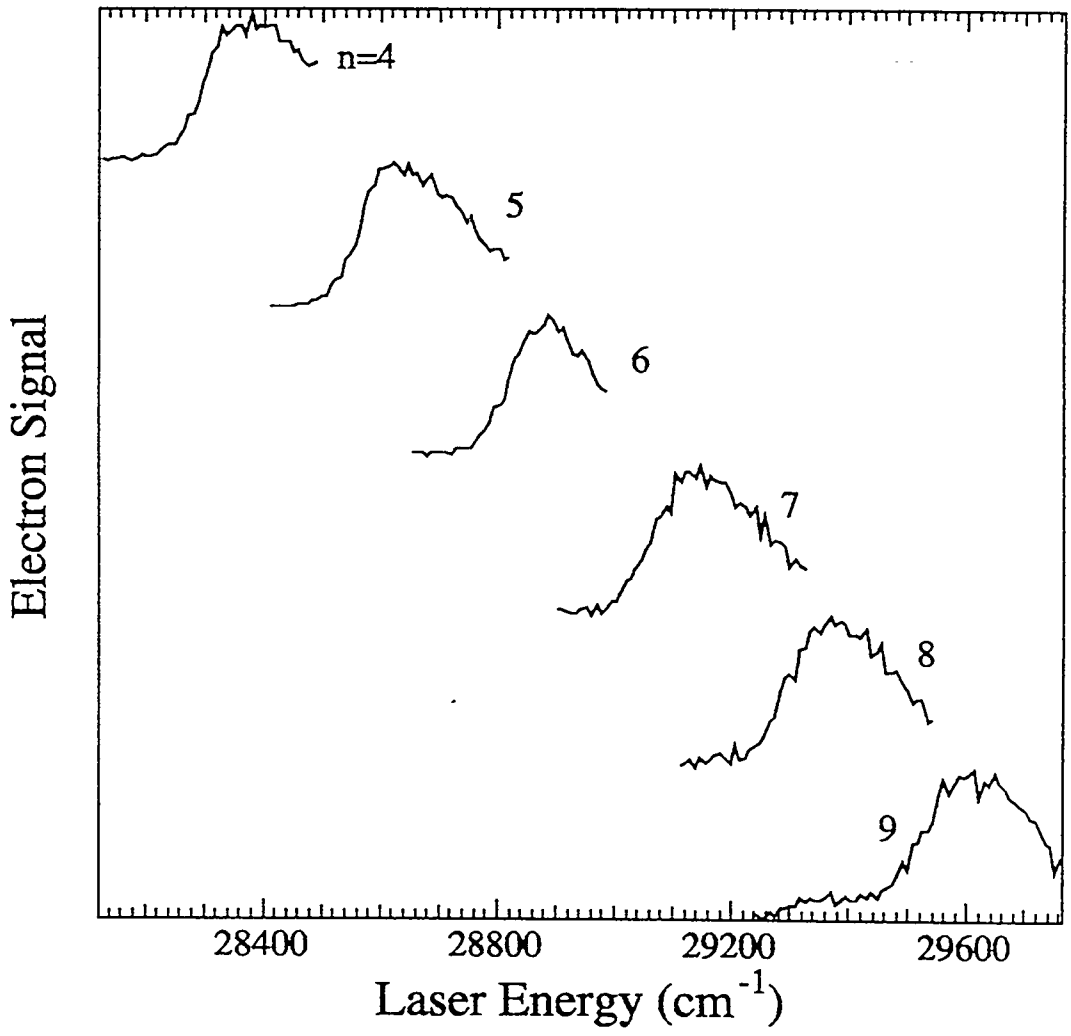

Figure 4.4. Partially discriminated threshold photodetachment (PDTP) spectra of $Ar_{4-9}Br^-$

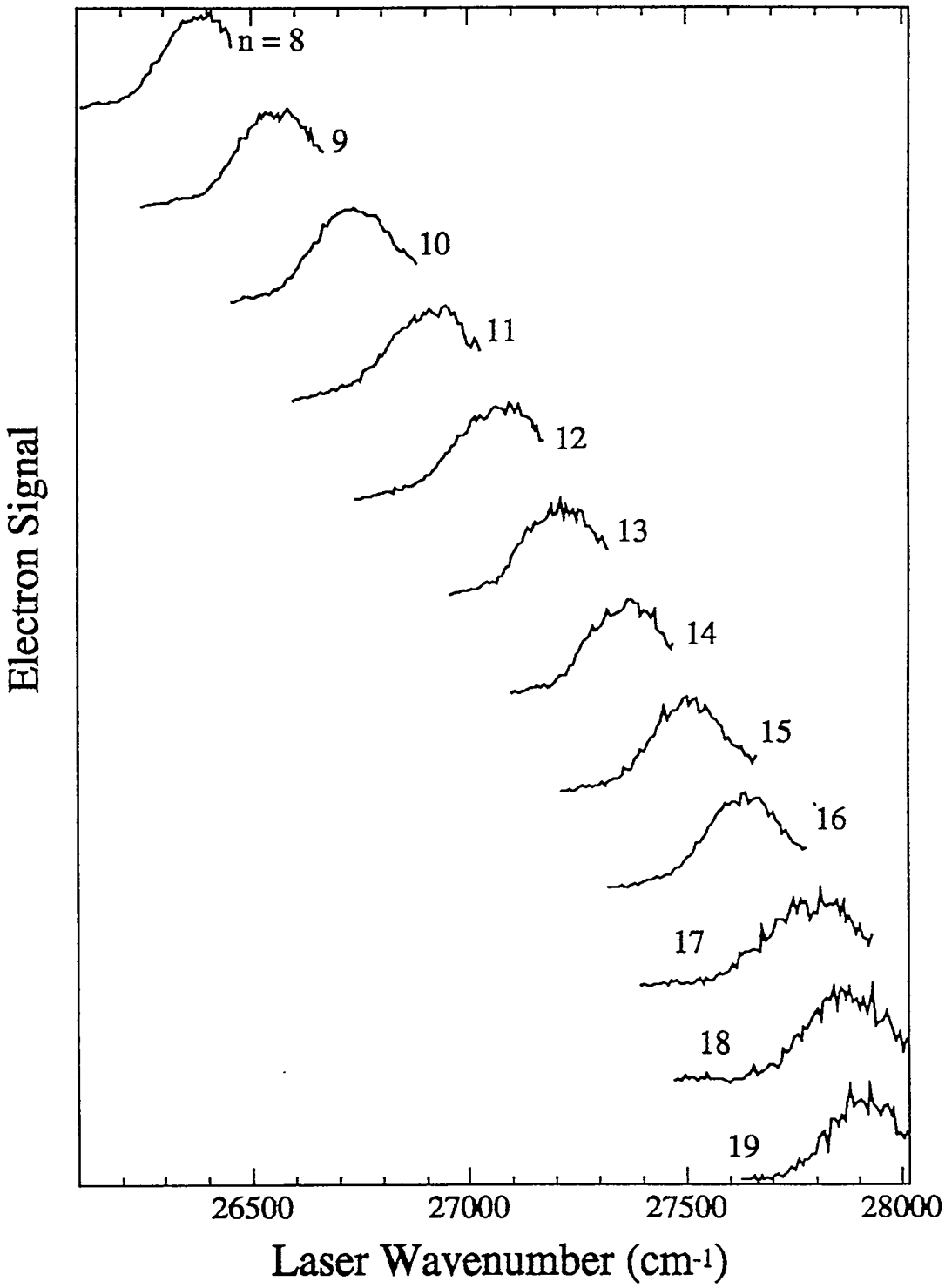

Figure 4.5. PDTP spectra of $Ar_{8-19}I^-$.

The peaks are all about 200 cm^{-1} wide (FWHM), with a rising edge of about 100 cm^{-1}. In order to estimate the adiabatic *EA*s from these thresholds, we compared the PDTP and ZEKE spectra of Ar_5I^-. The adiabatic *EA* of Ar_5I^- from the ZEKE spectrum corresponds to the 25% point on the rising edge of the PDTP threshold. Because the shape of the PDTP spectra is relatively unchanging for the larger clusters, we estimated the adiabatic *EA*s of the larger Ar_nBr^- and Ar_nI^- clusters from the 25% point of the partially discriminated cross sections. A reasonable estimate of the uncertainties in the *EA*s determined in this way is ± 50 cm^{-1}, corresponding approximately to the width of the rising edge of the PDTP thresholds.

The adiabatic electron affinities determined from the ZEKE and PDTP spectra of Ar_nBr and Ar_nI are shown in Tables 4.1 and 4.2, respectively. The origins of the *I* and *II* states, when observed, are also shown, as well as the neutral electronic state splittings $\Delta_{X\text{-}I}$ and $\Delta_{X\text{-}II}$. The stated uncertainties in the results obtained from ZEKE spectra were determined by considering the width and reproducibility of the peaks, as well as the scan step size.

Table 3.1. Experimental adiabatic electron affinities, excited state origins, and electronic state splittings for Ar_nBr. All energies are in cm^{-1}. Uncertainties are in parentheses.

n	*EA* (*X* State Origin)	*I* State Origin	Δ_{X-I}	*II* State Origin	Δ_{X-II}
0	27129.2[a]	-	-	30814[a,b]	3685[b]
1	27429.6 (3.0)	27467.4 (1.6)	37.8(2.3)	31132.3 (1.6)	3702.7 (3.4)
2	27722.4 (3.0)	27775.5 (5.0)	53.1(5.5)	31427.8 (2.2)	3705.4 (3.7)
3	27994.6 (3.0)	28059.4 (1.6)	64.8(2.3)	31702.8 (2.2)	3708.2 (3.7)
4	28266 (50)	-	-	-	-
5	28532 (50)	-	-	-	-
6	28778 (50)	-	-	-	-
7	29029 (50)	-	-	-	-
8	29258 (50)	-	-	-	-
9	29491 (50)	-	-	-	-

Table 4.2. Experimental adiabatic electronic affinities, excited state origins, and electronic state splittings for Ar_nI.

n	*EA* (*X* State Origin)	*I* State Origin	Δ_{X-I}	*II* State Origin	Δ_{X-II}
0	24673.3[a]	-	-	32276.5a,b	7603.15[b]
1	24888.3 (3.0)	24925.5 (1.5)	37.2 (2.2)	32512.6 (2.2)	7624.3 (3.7)
2	25100.9 (3.0)	25152.9 (3.0)	52.0 (3.4)	32731.2 (2.2)	7630.3 (3.7)
3	25303.0 (3.0)	25368.0 (4.5)	65.0 (5.1)	32936.4 (2.2)	7633.4 (3.7)
4	25502.2 (3.0)	25571 (10)	69 (10)	-	-
5	25702 (10)	25762 (10)	60 (14)	-	-
6	25907 (15)	25950 (15)	43 (21)	-	-
7	26083 (15)	26163 (10)	60 (18)	-	-
8	26247 (50)	-	-	-	-
9	26413 (50)	-	-	-	-
10	26582 (50)	-	-	-	-
11	26753 (50)	-	-	-	-
12	26904 (50)	-	-	-	-
13	27079 (50)	-	-	-	-
14	27226 (50)	-	-	-	-
15	27375 (50)	-	-	-	-
16	27488 (50)	-	-	-	-
17	27617 (50)	-	-	-	-
18	27717 (50)	-	-	-	-
19	27794 (50)	-	-	-	-

[a] H. Hotop and W.C. Lineberger, J. Phys. Chem. Ref. Data **14**, 731 (1985).

[b] C.E. Moore, *Atomic Energy Levels*, v. I, Circ. Natl. Bur. Std. 467 (1949).

4.4. Analysis and Discussion

Our goal in this section is to compare the experimentally observed electron affinities and electronic structure with predictions from model potentials. The schematic energy level diagram shown in Figure 6.6 on the next page relates the experimental observables, i.e. the adiabatic *EA*s and neutral electronic state splittings, to the model anion and neutral potentials. The observed *X-I* state splitting is then calculated from the model potential using

$$\Delta_{X-I} = \varepsilon_X - \omega_0^X - \varepsilon_I + \omega_0^I, \tag{4.3}$$

where ε_X and ε_I are the classical binding energies, and ω_0^X and ω_0^I are the zero point energies of the *X* and *I* states, respectively. Similarly, the *X-II* state splitting is given by

$$\Delta_{X-II} = \varepsilon_X - \omega_0^X - \varepsilon_{II} + \omega_0^{II} + \Delta, \tag{4.4}$$

where Δ is the atomic spin-orbit splitting. Using similar notation for the anion binding energy and zero point energy, the calculated adiabatic *EA* of a cluster is given by

$$EA(\mathrm{Ar_n Br}) = EA(\mathrm{Br}) + \varepsilon_a - \omega_0^a - \varepsilon_X + \omega_0^X, \tag{4.5}$$

where *EA*(Br) is the electron affinity of a bare bromine atom (3.3636 eV[25]). An analogous equation holds for Ar_nI clusters, with *EA*(I) = 3.0591 eV.[26]

Notice from Equations (4.3) and (4.4) that a comparison of the neutral potentials with the experimental electronic state splittings is possible without any knowledge of the anion potential. Thus to some extent the neutral and anion potentials can be compared with experiment independently of each other.

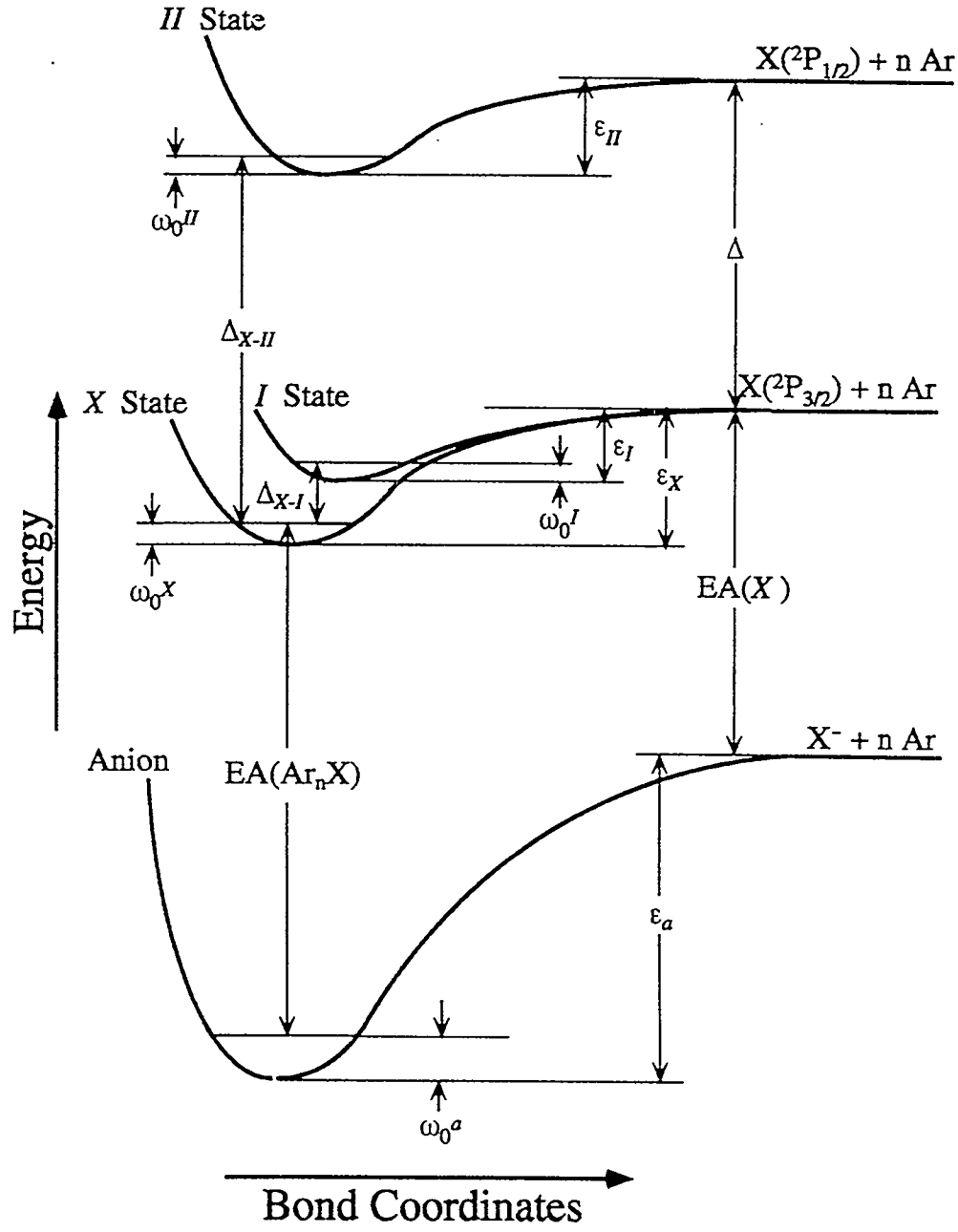

Figure 4.6. Schematic energy level diagram of the Ar_nX^- anion and Ar_nX neutral electronic states.

In order to use (4.3)-(4.5), we need model potential functions, and methods to determine the minimum energy cluster geometries and to calculate the zero point energies. These are described below. We demonstrate that the experimentally observed cluster properties are not consistent with pairwise additive potentials, and then consider various non-additive corrections to the potentials.

4.4.1. Pair Potentials

The pair potential of the ArBr neutral has been determined in scattering experiments by Lee *et al.*[27] The scattering experiments characterized only the neutral $X\frac{1}{2}$ and $I\frac{3}{2}$ state potentials. Our previous ZEKE results[4] on the ArBr complex provided further refinement of Lee's potential, as well as information on the neutral $II\frac{1}{2}$ state and anion potentials. In the case of ArI, scattering results are not available, so the ZEKE spectrum is the onlye source of information on the ArI diatomic potentials.

The neutral Ar-X potentials are of the Morse-Morse-switching function-van der Waals (MMSV) form. The reduced form of this potential, with $x = r/r_m$ and $f(x) = V(r)/\varepsilon$ is

$$\begin{aligned} f(x) &= e^{2\beta_1(1-x)} - 2e^{\beta_1(1-x)}, && 0 < x \le 1, \\ &= e^{2\beta_2(1-x)} - 2e^{\beta_2(1-x)} \equiv M_2(x), && 1 < x \le x_1, \\ &= SW(x)M_2(x) + [1 - SW(x)]W(x), && x_1 < x < x_2, \\ &= -C_{6r}x^{-6} - C_{8r}x^{-8} \equiv W(x), && x_2 \le x < \infty, \end{aligned} \tag{4.6}$$

where ε is the well depth, r_m is the bond length, and the switching function is given by

$$SW(x) = \frac{1}{2}\left[\cos\frac{\pi(x - x_1)}{(x_2 - x_1)} + 1\right], \tag{4.7}$$

The reduced dimensionless coefficients C_{6r} and C_{8r} are related to the usual dispersion coefficients C_6 and C_8 by

$$C_{6r} = \frac{C_6}{\varepsilon r_m^6}, \qquad C_{8r} = \frac{C_8}{\varepsilon r_m^8}. \tag{4.8}$$

The anion potentials have the same form, except that the van der Waals portion is replaced by a function including charge-induced dipole (r^{-4}), and charge-induced quadrupole and dispersion (r^{-6}) terms:

$$f(x) = -B_{4r}x^{-4} - B_{6r}x^{-6} \equiv W(x), \qquad x_2 \le x < \infty, \tag{4.9}$$

with

$$B_{4r} = \frac{B_4}{\varepsilon r_m^4}, \qquad B_{6r} = \frac{B_6}{\varepsilon r_m^6}. \tag{4.10}$$

Further details about the construction of the Ar-Br and Ar-I pair potentials are given in Reference 4. The MMSV potential parameters used in this work are given in Table 4.3, below.

Table 4.3. MMSV pair potential parameters of argon halides.

	ArI				ArBr			
	$X\,^1\Sigma_{1/2}^+$	$I\,^3\Pi_{1/2}$	$II\,^1\Pi_{1/2}$	Anion	$X\,^1\Sigma_{1/2}^+$	$I\,^3\Pi_{1/2}$	$II\,^1\Pi_{1/2}$	Anion
ε (meV)	18.8	13.9	16.0	45.8	16.5	11.5	14.0	54.4
r_m (Å)	3.95	4.18	4.11	4.07	3.73	3.94	3.89	3.78
β_1	7.15	7.25	6.90	5.70	6.80	7.72	6.70	5.10
β_2	6.18	6.30	6.40	4.45	6.50	7.10	6.35	4.45
x_1	1.01	1.04	1.04	1.08	1.02	1.012	1.01	1.065
x_2	1.62	1.62	1.64	1.62	1.59	1.63	1.58	1.66
C_6 (eV•Å^6)	98.4	98.4	98.4	-	65.2	70.2	68.8	-
C_8 (eV•Å^8)	715	715	715	-	379	379	379	-
B_4 (eV•Å^4)	-	-	-	12.8	-	-	-	12.5
B_6 (eV•Å^6)	-	-	-	162	-	-	-	120.5

Some of the parameters used here have been modified slightly from those published previously.[4] The reason for this is that in the previous work, the well depths of the three neutral electronic states and the anion were related to each other using the relationships implied by Figure 4.6, with the zero point energies used in these relations assumed to be equal to half of the observed vibrational fundamental frequencies. A slightly more accurate procedure, used to obtain the parameters in Table 4.3, is to calculate the actual zero point energies from the model potentials. The well depth parameters are then iteratively adjusted in order to satisfy Equations. (4.3)- (4.5), as well as to fit the observed spectra. The well depths obtained in this way differ from those given previously by no more than 2-3 cm^{-1}, which is within the uncertainties stated in Reference 4.

It is important here to consider the uncertainties in the pair potential parameters. In the case of the ArBr potentials, the scattering experiments provide information on the absolute values of the well depths and bond lengths for the $X\,\frac{1}{2}$ state. On the other hand, the ZEKE spectra, although quite sensitive to the *relative* bond lengths and well depths between the anion and neutral states, are not very sensitive to the *absolute* values of these parameters. Therefore, the r_m and ε parameters for the $X\,\frac{1}{2}$ state of ArBr were fixed at the values of Lee *et al.*[27] The parameters for the anion and remaining neutral states were then adjusted to be consistent with the relations implied by Fig. 4.6, as well as to reproduce the ZEKE spectrum. The uncertainties in r_m and ε stated in Reference 27 are ±0.2 Å and ±9 cm^{-1}, respectively, so the absolute uncertainties of r_m and ε for the anion and the other neutral states are of about the same order. However, the *relative* uncertainties in r_m and ε between the anion and neutral are significantly smaller than this.

For example, because the uncertainty in the *EA* obtained from the ZEKE spectrum is ±3 cm^{-1}, the difference ε_a - ε_X is known with about this same uncertainty. (See Equation 2). Similarly, the relative uncertainties in r_m are found to be about 1-2% (0.04-0.08 Å), based on the fit to the ZEKE spectra.

For ArI, for which scattering experiments have not been performed, modified versions of the polarizability correlation formulae of Pirani *et al*[28] were used to estimate r_m and ε for the $II\frac{1}{2}$ state of ArI, as described in Reference 4. Then the remaining neutral and anion potentials were adjusted to fit the ZEKE spectrum. The estimated *absolute* uncertainties in r_m and ε of ArI are ±18 cm^{-1} for ε, and ±0.2 Å for r_m. However, the same considerations about the *relative* uncertainties among the anion and neutral states also apply for ArI. The relative uncertainties in ε and r_m are ±3 cm^{-1} and ±0.04-0.08 Å, respectively.

To model the Ar-Ar pair interaction, the accurate Hartree-Fock Dispersion (HFD-B2) potential of Aziz and Slaman[29] was used. For this potential r_m = 3.7565 Å and ε = 99.5465 cm^{-1}. For the detailed form and other parameters of this well-known potential, see Reference 29.

The pairwise additive approximations to the Ar_nBr^- and Ar_nI^- binding energies were found by minimizing the additive potentials, using the simulated annealing procedure to be described below, from:

$$\varepsilon_a = \min\left(V_{ArX} + V_{ArAr}\right), \tag{4.11}$$

with $V_{ArX} = \sum_i V_{i0}\left(\left|\mathbf{r}_i - \mathbf{r}_0\right|\right)$, and $V_{ArAr} = \sum_{i<j} V_{ij}\left(\left|\mathbf{r}_i - \mathbf{r}_j\right|\right)$, where the sums run over the Ar atoms, $\mathbf{r}_i$ is an Ar atom position, and $\mathbf{r}_0$ is the halide position. The calculation of the

neutral potentials is more complex because of the open shell nature of the halogen atom, and is discussed in Section 4.4.5, below.

4.4.2. Simulated Annealing Method

We use a simple molecular dynamics simulated annealing procedure to determine the minimum energy cluster geometries. The simulated annealing program used here was adapted from a molecular dynamics program written by Li and Martens.[30] The procedure used is as follows:

(1) Random initial atomic positions are generated. The initial positions lie within a 6-15 Å box, depending on the size of the cluster, and are subject to the constraint that no two atoms may be closer than a certain cutoff distance, usually 3.5 Å. The latter condition ensures that the cluster starts out in an attractive region of the potential surface so that dissociation does not occur.

(2) The classical equations of motion are solved for about 5 ps, using a Gear predictor-corrector algorithm started with a 16 step Runge-Kutta algorithm[30]. The step size is 5 fs.

(3) Kinetic energy is removed by rescaling the atomic velocities. When starting with random positions, the kinetic energy is removed very quickly, so that the velocities and kinetic energies are essentially reset to zero with each rescaling. This rapid quenching was found to be necessary to prevent evaporation.

(4) Steps (2) and (3) are repeated until a minimum is found. This typically requires 100-250 ps.

(5) Beginning with the minimum configuration found by the above procedure, kinetic energy is added, constrained so that the translational energy of the cluster center-of-mass and its angular momentum are zero. To prevent evaporation, the initial kinetic energy was usually set to not more than 25-33% of the total well depth. Then steps (2) and (3) are repeated, but with kinetic energy removed much more gradually, by rescaling the velocities by a factor[31]

$$\left[1+\frac{\tau_{scale}}{\tau_{const}}\left(\frac{KE_{targ}}{KE_{avg}}-1\right)\right]^{1/2}$$

every 5 ps. Here τ_{scale} is the time between rescalings, τ_{const} is a time constant--typically 50 or 100 ps--KE_{avg} is the average kinetic energy, and KE_{targ} is a target kinetic energy, set to a very small value in order to find a minimum. The entire procedure typically requires 5-10 ns.

(5) Finally, the minimum energy configuration is located more precisely using a simple gradient minimization routine.[32]

The entire annealing procedure was repeated 5-20 times for each cluster to ensure that the global minimum was found. In this process, low-lying local minima were often also found. In order to locate higher lying local minima, an interval of 250 fs or less between rescaling steps is used in steps (2)- (4), to prevent equilibration of the cluster as kinetic energy is removed.

For further details about simulated annealing and the computer program used to implement it, see Appendix C.

4.4.3. Zero Point Energy Calculation

Once the minimum energy configurations and classical binding energies are found, it is necessary to know the zero point energies in order to use Equations (4.3)- (4.5). The model potentials are analytical functions of the nuclear Cartesian coordinates, allowing the zero point energies to be estimated by the following procedure. The normal coordinates of the clusters were found in terms of linear combinations of Cartesian displacement coordinates, using standard techniques.[33] Then each of the $3N$-6 single-mode vibrational Schrödinger equations was solved using a simple one-dimensional discrete variable representation (DVR) procedure.[5,34,35,36] In this way, the anharmonicity of the potential is approximately accounted for, although interactions between normal modes are neglected. The total zero point energy was then obtained by adding up the single mode values. The zero point calculation was limited to the portion of the potential in the vicinity of the minimum structure, so that any splittings due to tunneling are not reproduced.

For details of the computer program used to calculate the zero point energies, refer to Appendix C, Sections C5 and C6.

4.4.4. Anion Minimum Energy Geometries

The minimum energy geometries found using pairwise additive potentials for $Ar_{2\text{-}19}I^-$ are shown in Figure 4.7. Similar structures were found for $Ar_{2\text{-}9}Br^-$. The calculated anion binding energies and zero point energies are given in Tables 4.4 and 4.5, on the following pages.

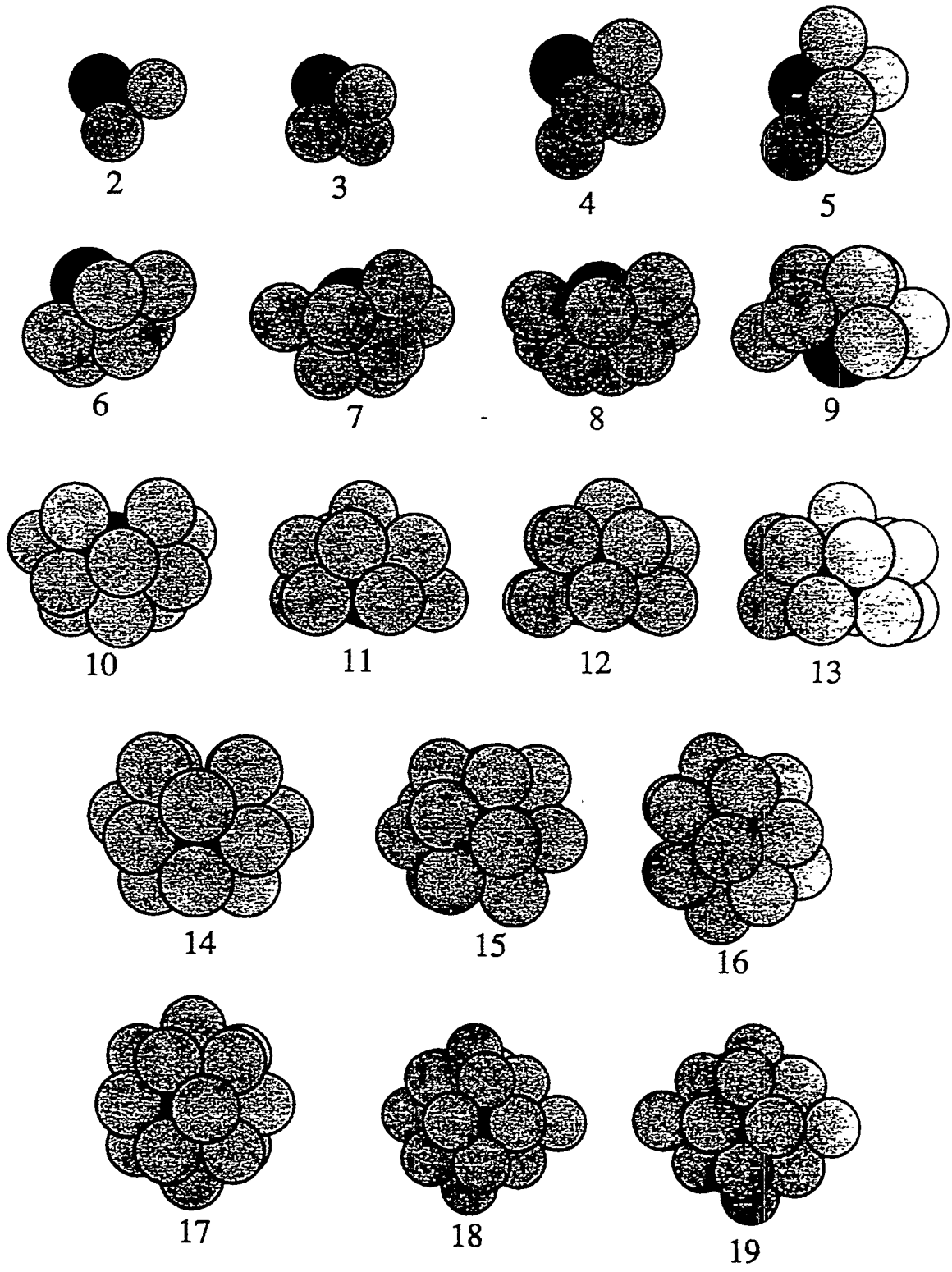

Figure 4.7. Minimum energy structures of $Ar_{2\text{-}19}I^-$ clusters found using pairwise additive potentials.

Table 4.4. Results of calculations with pairwise additive Ar_nBr^- anion potentials, and "matrix-additive" Ar_nBr neutral potentials. All energies are in cm^{-1}.

n	ε_a	ε_x	ε_I	ε_{II}	ω_0^a	ω_0^X	ω_0^I	ω_0^{II}	$\Delta_{X\text{-}I}$	$\Delta_{X\text{-}II}$	EA_{add}
0	0	0	0	0	0	0	0	0	0	3685	27129.2
1	438.8	133.1	92.8	112.9	20.7	15.6	13.5	13.4	38.2	3703.0	27429.8
2	977.1	349.4	295.4	319.6	56.8	42.3	45.0	44.6	56.6	3717.1	27742.3
3	1614.9	662.6	599.9	628.7	107.4	85.1	89.6	88.6	67.3	3722.4	28059.2
4	2260.0	977.3	916.6	945.0	157.8	127.6	135.2	132.9	68.4	3722.6	28382
5	2911.3	1293.6	1243.7	1267.3	207.5	168.7	181.1	176.4	62.3	3719.0	28708
6	3659.7	1696.2	1677.2	1686.0	267.5	211.7	248.5	230.0	55.8	3713.6	29037
7	4318.6	2042.9	2001.2	2014.8	315.5	268.9	289.7	272.9	62.5	3717.1	29358
8	5069.8	2465.0	2413.9	2432.4	371.1	315.8	337.1	323.6	72.4	3725.4	29679
9	5815.2	2880.0	2807.1	2837.7	421.3	360.3	375.7	370.1	88.2	3737.1	30003

Table 4.5. Results of calculations with pairwise additive Ar_nI^- anion potentials, and "matrix-additive" Ar_nI neutral potentials. All energies are in cm^{-1}.

n	ε_a	ε_X	ε_I	ε_{II}	ω_0^a	ω_0^X	ω_0^I	ω_0^{II}	$\Delta_{X\text{-}I}$	$\Delta_{X\text{-}II}$	EA_{add}
0	0	0	0	0	0	0	0	0	0	7603.15	24673.3
1	369.4	151.6	112.1	129.0	17.3	14.6	12.3	13.2	37.2	7624.4	24888.3
2	838.4	385.4	332.0	355.5	51.0	41.2	42.4	42.4	54.5	7634.3	25116.5
3	1406.8	718.1	653.1	682.5	98.4	83.2	85.8	85.4	67.6	7640.9	25346.8
4	1982.0	1051.5	986.0	1016.3	145.8	125.1	130.1	128.7	70.5	7642.0	25583
5	2562.3	1385.6	1328.5	1355.1	192.7	165.8	174.9	171.3	66.2	7639.2	25823
6	3229.2	1799.2	1767.6	1782.7	249.4	208.3	233.9	222.5	57.3	7633.8	26062
7	3815.2	2151.3	2112.5	2126.9	295.3	257.8	279.6	264.3	60.6	7634.0	26300
8	4469.2	2569.3	-	-	343.5	300.4	-	-	-	-	26530
9	5094.8	2963.2	-	-	379.9	334.7	-	-	-	-	26760
10	5731.8	3369.0	-	-	436.9	390.0	-	-	-	-	26990
11	6366.8	3764.8	-	-	476.1	428.1	-	-	-	-	27227
12	7044.3	4199.3	-	-	539.6	483.9	-	-	-	-	27463
13	7796.7	4706.4	-	-	586.8	530.4	-	-	-	-	27707
14	8519.3	5159.0	-	-	645.1	572.8	-	-	-	-	27961
15	9280.6	5686.6	-	-	710.5	641.6	-	-	-	-	28198
16	9914.3	6143.3	-	-	777.5	693.7	-	-	-	-	28360
17	10561.3	6589.0	-	-	850.4	770.0	-	-	-	-	28565
18	11102.2	7061.3	-	-	913.0	804.9	-	-	-	-	28606
19	11648.0	7498.0	-	-	979.3	880.8	-	-	-	-	28725

For $Ar_{2-3}X^-$ (X = Br or I), there is only one minimum, in which all atoms are in contact with each other. Linear (Ar_2X^-) or planar (Ar_3X^-) geometries are not stable with additive potentials. Furthermore, the *X-I* state splittings expected for linear and planar geometries are not consistent with those observed experimentally.

In the minimum energy structures of larger clusters (Ar_nI^- $4 \leq n \leq 17$, and Ar_nBr^- $4 \leq n \leq 9$), all the Ar atoms contact the central halide atom. This type of structure is energetically favorable because each Ar-X^- "bond" is about four times stronger than an Ar-Ar "bond." For Ar_4X^-, one local minimum isomer is seen (C_{3v} point group) which has one Ar atom in contact with the other three argons but not with the halide. In Ar_4I^-, its energy is about 200 cm^{-1} higher than that of the global minimum. The analogous Ar_4Br^- isomer lies 256 cm^{-1} above the global minimum. These energy differences correspond approximately to one Ar-X^- "bond". Ar_5X^- has two local minima with approximately the same separations from the global minimum as in the Ar_4X^- clusters.

The clusters with $6 \leq n \leq 17$ show two types of local minima. In one type, the Ar atoms are all in contact with the halide--as in the global minimum--but have fewer Ar-Ar "bonds." These typically differ in energy from the global minimum by approximately the magnitude of an Ar-Ar "bond," i.e. about 100 cm^{-1}. The other type, seen already for $n<6$, are structures in which one or more Ar atoms are not in direct contact with the halide. This type of isomer usually differs in energy from the global minimum by approximately the energy of one or more Ar-X^- "bonds."

For Ar_nI^-, rare gas atoms continue to fit around the halide without significant crowding up to n=15. At n=16 there is some crowding, so that the Ar-I^- contribution to the potential is reduced. $Ar_{17}I^-$ constitutes a "closed" solvent shell (at 0 K). It consists of a

capped pentagonal bipyramid structure (D_{5h}), with the axial Ar atoms significantly further from the halide than the others. Subsequent Ar atoms are added outside the first solvent shell. In the case of Ar_nBr^-, we did not observe the closing of the solvent shell since we did not perform calculations for n>9.

4.4.5. Neutral Open-Shell Potentials

Because of the anisotropy of the open-shell halogen atom in the neutral clusters, the potentials cannot in general be obtained by simply adding the Ar-X pair potentials. This is clear from the observed spectra. For example, in the diatomic ArI molecule an *X-I* splitting of 37 cm^{-1} is observed.[4] If the potentials were simply additive, one would predict an *X-I* splitting of 74 cm^{-1} for Ar_2I. The observed $\Delta_{X\text{-}I}$ in Ar_2I is 52 cm^{-1}. The simple additive prediction is well outside experimental uncertainty.

This "non-additivity" of the open-shell potentials has been discussed by Lawrence and Apkarian,[37] whose explanation we follow here. The non-additivity can most easily be understood if we momentarily neglect the effect of spin-orbit coupling. In this case there are two electronic states of the diatomic complex corresponding to the two possible orientations of the singly occupied halogen p-orbital relative to the argon atom.[24,38] A $^2\Sigma$ state arises when the singly occupied p-orbital lies along the internuclear axis, and a doubly degenerate $^2\Pi$ state corresponds to the singly occupied p-orbital lying perpendicular to the internuclear axis. However, if the cluster contains additional Ar atoms, Λ is no longer a good quantum number if the polyatomic cluster is not linear. Consider, for example, the case of Ar_2I. The singly occupied halogen p-orbital will not, in general, lie either parallel or perpendicular to either of the Ar-I internuclear axes.

Therefore the Ar-I interaction potentials in Ar_2I will not be the same as the potentials of either the $^2\Sigma$ or $^2\Pi$ diatomic states, but--in the first approximation--may be considered to be linear combinations of the diatomic potentials. Thus, in order to obtain the potentials of Ar_2I, and larger open-shell clusters, from the diatomic potentials, our concept of pairwise additivity must be *extended* to include this mixing of the diatomic electronic states. We describe how this is done in more detail below.

A simple first-order perturbation theory treatment of the interaction of an open-shell atom with several closed-shell (rare gas) atoms in terms of the diatomic potentials has been developed by various workers.[39,40,41,42] These methods have been used to study open-shell atoms in rare gas matrices, clusters, and on surfaces.[12,37,43,44,45] Our implementation here most closely resembles that of Lawrence and Apkarian,[37] who studied the emission spectra of I atoms in Xe and Kr matrices. The theory is briefly as follows.

The Ar_n-X interaction is modeled by an effective potential depending on the rare gas coordinates and on the coordinates of the "hole" in the singly occupied halogen p-orbital in an arbitrary space-fixed frame:

$$H = \sum_k V_{Ar_nX}(\mathbf{r}, \mathbf{R}_k) + H_{SO}. \qquad (4.12)$$

Here, the sum is over the rare gas atoms, $\mathbf{r}$ is the coordinate of the "hole," $\mathbf{R}_k$ are the rare gas coordinates relative to the halogen nucleus, and H_{SO} is the spin-orbit interaction Hamiltonian.

The potential V_{Ar_nX} is then expanded in Legendre polynomials in $\hat{\mathbf{r}} \cdot \hat{\mathbf{R}}_k$. We are ultimately interested in the matrix elements of H' in a p-orbital basis, and only the first two even terms of the expansion contribute to these. Hence, we write

$$H' = \sum_k \left[V_0(r, R_k) + V_2(r, R_k) P_2\left(\hat{\mathbf{r}} \cdot \hat{\mathbf{R}}_k\right) \right] + H_{SO} . \tag{4.13}$$

In the diatomic case (one Ar atom), the expectation values of these two expansion coefficients, $V_0(R)$ and $V_2(R)$, can be shown, using the relations given by Haberland[24] and Aquilanti *et al.*,[38] to be related to the spectroscopic diatom potentials by

$$V_0(R) = \tfrac{1}{3}\left[V_{X\frac{1}{2}}(R) + V_{I\frac{1}{2}}(R) + V_{II\frac{1}{2}}(R) \right] \tag{4.14}$$

and

$$V_2(R) = \tfrac{5}{3}\left[V_{X\frac{1}{2}}(R) + V_{II\frac{1}{2}}(R) - 2\, V_{I\frac{1}{2}}(R) \right]. \tag{4.15}$$

Here, the zero for each potential is set at the potential asymptote. ($^2P_{1/2}$ for $V_{I3/2}$, and $^2P_{3/2}$ for $V_{X1/2}$ and $V_{II1/2}$). In deriving these equations it is assumed that the spin-orbit constant, Δ, is independent of R.

With some effort, one can show that for a cluster with many Ar atoms, the perturbation Hamiltonian **H'** is given by a 6x6 matrix:[37,42]

$$\mathbf{H'} = \sum_k V_0\left(R_k\right) \cdot \mathbf{1} + V_2\left(R_k\right) \cdot \mathbf{M}\left(\mathbf{R}_k\right), \tag{4.16}$$

where $\mathbf{M}(\mathbf{R}_k)$ is a 6x6 Hermitian matrix involving the argon atom coordinates. The detailed form of the matrix **H'** has been given, in the $|J, m_J\rangle$ basis, by Lawrence and Apkarian.[37] Diagonalization of **H'** yields three doubly degenerate eigenvalues, corresponding to the potentials of the X, I and II states.

In our implementation, an analytical form for the eigenvalues was found using the Maple V program. This allowed the eigenvalues to be calculated approximately 10 times faster than by numerical diagonalization and saved considerable computer time. The potentials, V_{Ar_nX}, are then referred to their own asymptotes by adding $\frac{1}{3}\Delta$ to the X and I

state potentials, and subtracting $\frac{2}{3}\Delta$ from the *II* state potential. The total potential of the cluster is then obtained by adding the Ar-Ar potentials in a pairwise fashion. The well depths are found by minimizing these potentials using the simulated annealing and gradient minimization procedures described in Section 4.4.2. For the *X* state, for example,

$$\varepsilon_X = \min\left(V_{Ar_nX} + V_{ArAr}\right), \tag{4.17}$$

with V_{ArAr} the same as in Equation (4.11).

There are several assumptions implicit in this treatment of the open shell potentials. First, the basis set is limited to p-orbitals; excited orbitals of the halogen or rare gas atoms are not included. Thus, many-body effects due to polarization of the halogen atom or charge-transfer are neglected. Also, we assume that the spin-orbit constant, Δ, is independent of the internuclear separations, as well as independent of the number of rare gas atoms in the cluster. To verify the former assumption, Δ was calculated as a function of *R* for ArBr and ArI, using the relations given by Haberland[24] and Aquilanti *et al.*[38], and the three diatomic potential energy curves determined from the ZEKE spectra. The calculated Δ does not vary more than 1 meV (0.1%) for ArI and not by more than 5 meV (1%) for ArBr for *R* greater than the zero crossing point. The assumption that Δ is independent of the number of argon atoms is more questionable, as we will see below.

The above method of calculating the adiabatic potential surfaces was used directly in the simulated annealing procedure for the smaller clusters ($n \leq 6$). For the larger clusters, the annealing was first performed using the anion potentials described above, and then the system was allowed to relax (to optimize the geometry) on each of the neutral surfaces. In most cases, the anion and neutral have approximately the same global minimum

configurations. There are some exceptions. For instance, the global minimum isomer of the Ar_5Br^- anion has all five of the Ar atoms in contact with the Br^- atom, but this geometry corresponds to a local minimum of the neutral *X* state surface. In such cases, the neutral minimum corresponding to the *anion* global minimum was always used to compute the "adiabatic" EAs and neutral electronic state splittings.

The results of the calculation of the neutral binding energies, zero point energies, $\Delta_{X\text{-}I}$, and $\Delta_{X\text{-}II}$ are presented in Tables 4.4 and 4.5, for $Ar_{2\text{-}9}Br$ and $Ar_{2\text{-}19}I$, respectively. It is interesting to note that for all n>1, the zero point energy of the *I* state is greater than that of the *X* state, contrary to intuition. This seems to be due to the steep repulsive wall of the $I\frac{1}{2}$ diatomic state, which causes the antisymmetric modes to be more steeply curved in the *I* than in the *X* state. The result is an increase in $\Delta_{X\text{-}I}$ over what would be calculated if the zero point energies were neglected.

We can compare the *X-I* splittings calculated using Equation (4.3) with the experimental results without reference to the anion potential. This comparison is shown in Figure 4.8. In the cases where the two states are well resolved, the agreement with experiment is quite satisfactory.

For n=2 and 3 the splitting between the *X* and *II* states may also be compared with experiment using Equation (4.4), as shown in Figure 4.9.

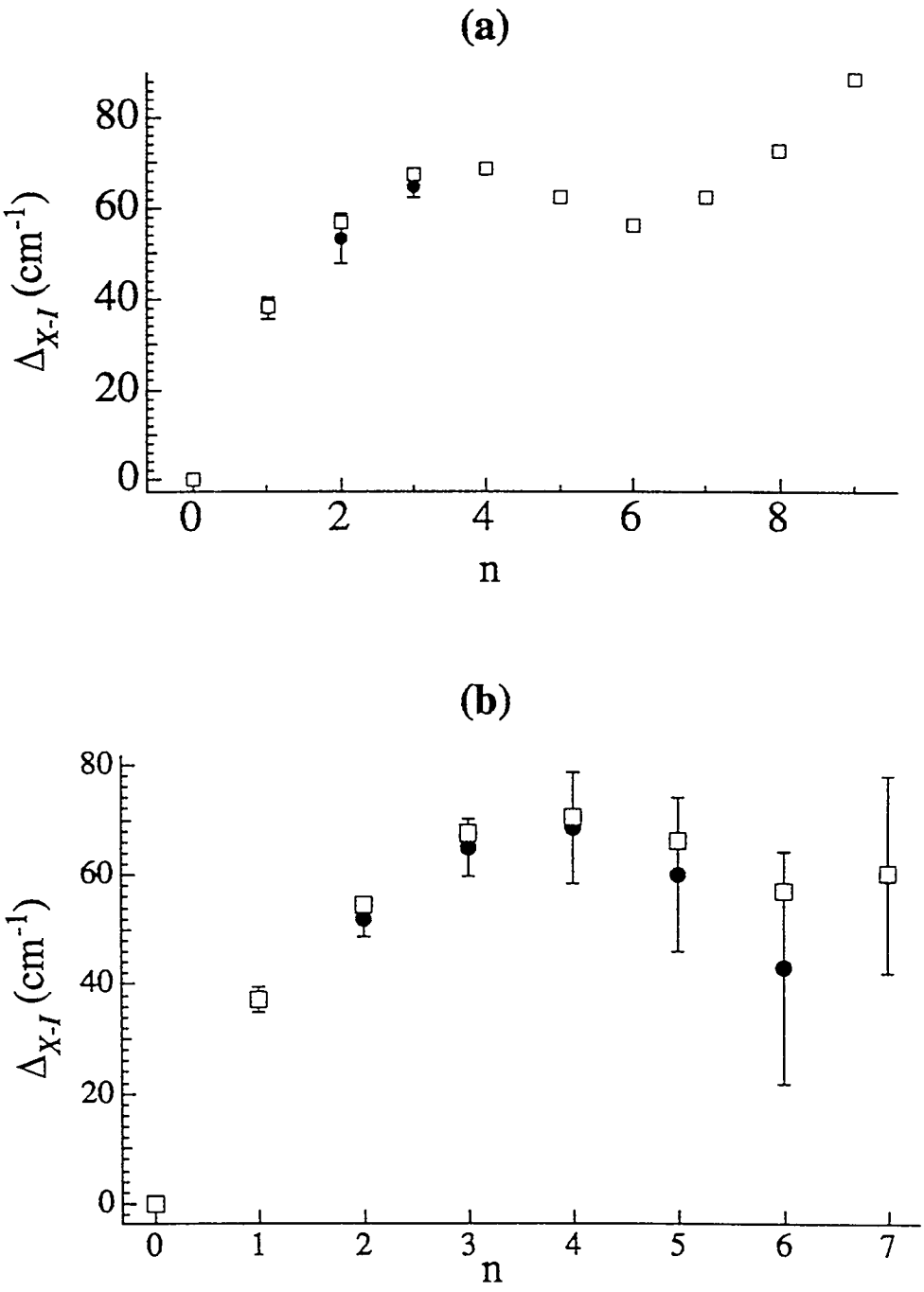

Figure 4.8. Comparison of experimental and calculated *X-I* state splittings for (a) Ar_nBr and (b) Ar_nI. Solid circles: experimental. Open squares: calculated as described in Section 4.4.5.

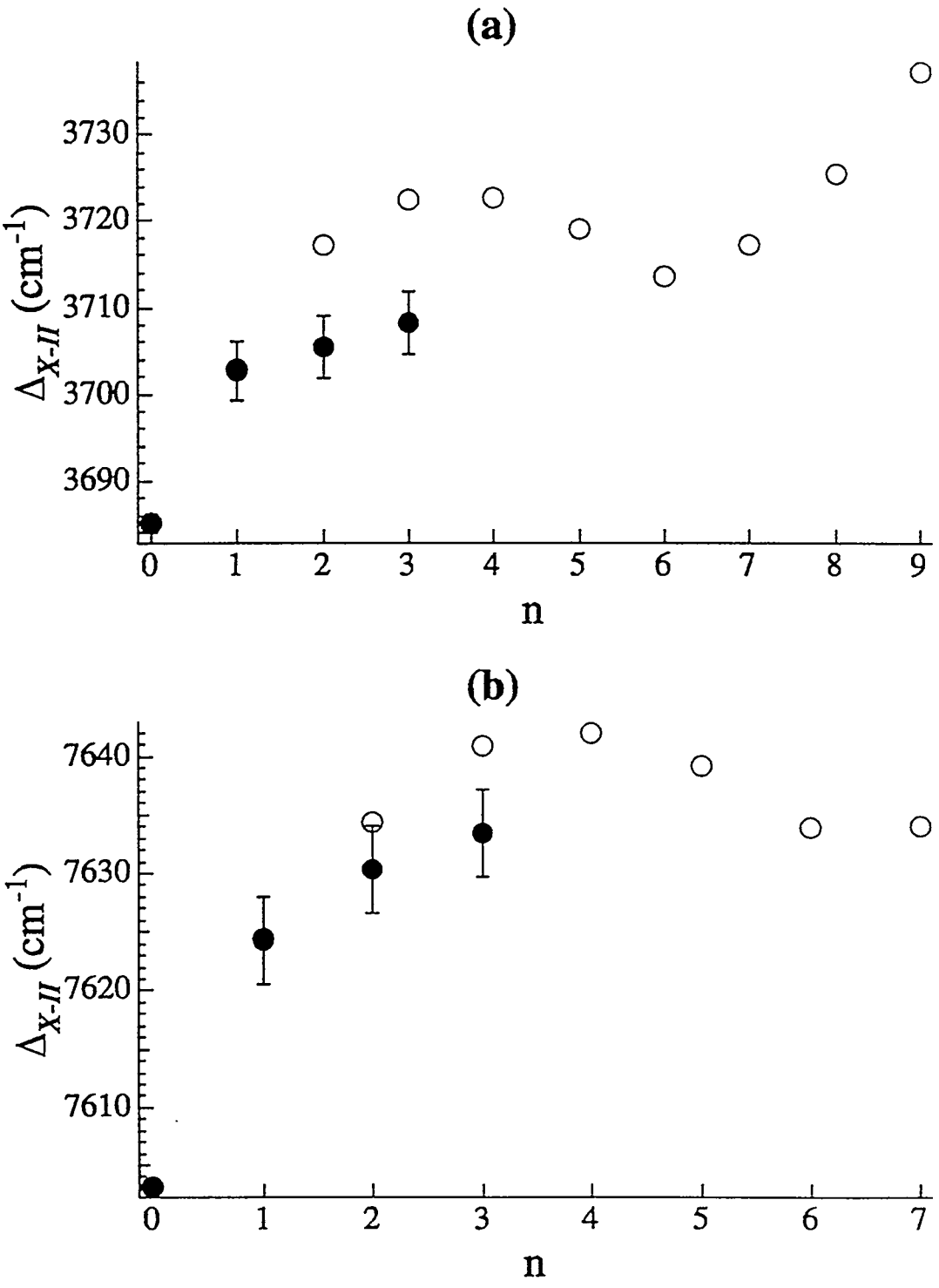

Figure 4.9. Comparison of experimental and calculated *X-II* state splittings for (a) Ar_nBr, and (b) Ar_nI. Solid circles: experimental. Open circles: calculated as described in Section 4.4.5.

For both $Ar_{2-3}Br$ [Figure 4.9(a)] and $Ar_{2-3}I$ [Figure 4.9(b)], the theoretical Δ_{X-II} is greater than the experimental value by about 5-15 cm^{-1}. The agreement is somewhat worse for $Ar_{2-3}Br$ than for $Ar_{2-3}I$. This discrepancy could mean that the atomic spin-orbit splitting, Δ, is not independent of the number of Ar atoms, as was assumed above. It is known that the spin-orbit splitting of atoms in rare gas matrices is different from that of the free atoms. For example, Lawrence and Apkarian found that the I atom spin-orbit splitting is decreased by about 3% or 5% in Xe or Kr matrices, respectively.[37] We observe a smaller decrease of Δ in the small clusters studied here: about 0.06%-0.1% in $Ar_{2-3}I$ and 0.3%-0.4% in $Ar_{2-3}Br$.

Generally speaking, the open shell interactions described in this section are non-additive, in the sense that they are of the form of the additional terms in Eq. 4.2. However, they are not true many-body effects because they can be obtained directly from the pair potentials, and do not introduce additional interactions between the Ar atoms in contrast to the effects described in Section 4.4.7, below. As pointed out by Sando and coworkers,[42] the open-shell potentials in the n>1 clusters can be considered to be additive as matrices rather than as scalars, and we will refer to these interactions as "matrix additive" effects in the rest of this discussion.

4.4.6. Electron Affinities Calculated from Additive Potentials

The adiabatic *EA*s calculated from Equation (4.5) using the additive anion potentials [Equation (4.11)] and "matrix additive" neutral potentials [Equation (4.17)] are given in Tables 4.4 and 4.5. These are compared with the experimental *EA*s in Figures 4.10 and 4.11.

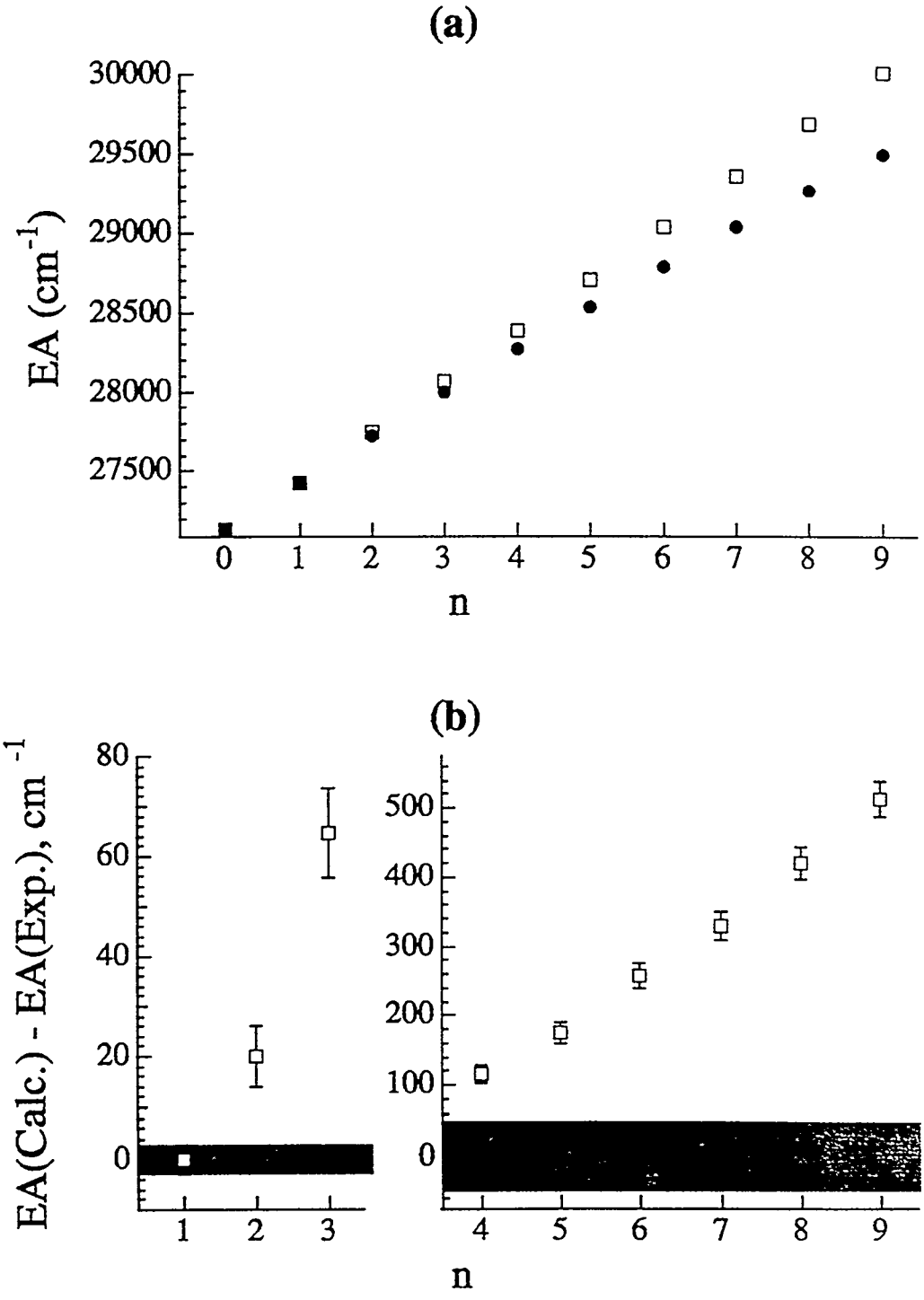

Figure 4.10. Comparison of experimental Ar_nBr electron affinities (*EA*s) with those calculated from the pairwise additive model. (a) *EA* as a function of n. Solid circles: experimental *EA*s. Open squares: additive calculation. (b) Difference between calculated and experimental *EA*s as a function of n. The shaded region represents the experimental uncertainty. The error bars represent the uncertainty in the calculated *EA*s.

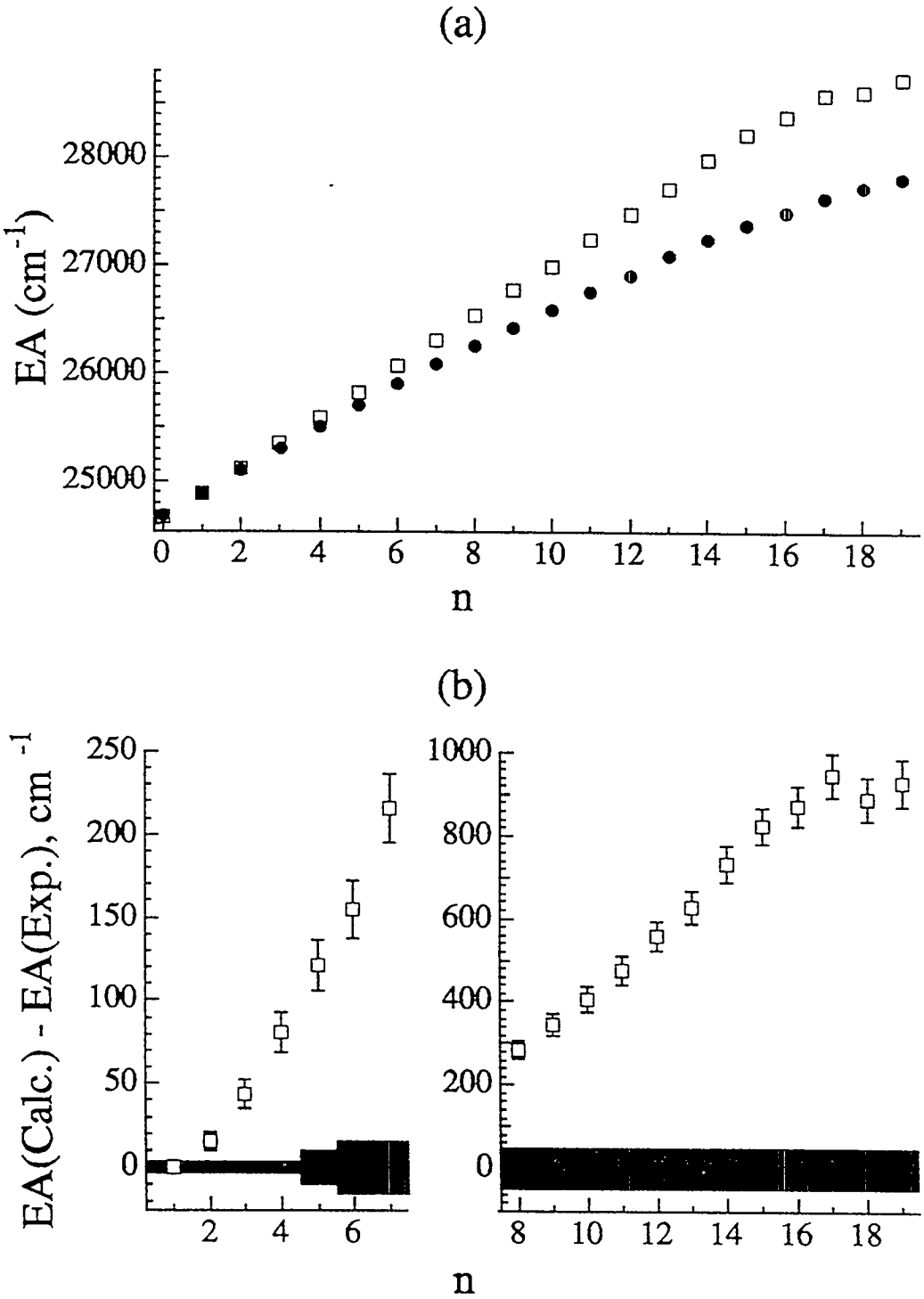

Figure 4.11. Comparison of experimental Ar_nI *EA*s with those calculated from the pairwise additive model. (a) *EA* as a function of n. Solid circles: experimental *EA*s. Open squares: additive calculation. (b) Difference between calculated and experimental *EA*s plotted as a function of n. The shaded region represents the experimental uncertainty. The error bars represent the uncertainties in the calculated *EA*s.

First, notice that in both $Ar_{2-9}Br$ and $Ar_{2-19}I$ the calculated *EA*s are significantly larger than the experimental results. For $Ar_{17-19}I$, the calculated *EA*s are almost 1000 cm^{-1} larger than the experimental values. Furthermore, the calculated *EA*s for $n \leq 17$ are nearly linear as a function of n. There is a slight positive curvature due to the nonadditivity of the neutral *X* state, and for $n>17$ the plot becomes flat in the case of Ar_nI^-. On the other hand, the experimental *EA*s display a significant negative curvature when plotted versus n. In Ar_nI, the flattening out at n=17 is not observed. Clearly the model potentials, as described so far, are not consistent with experiment.

Before we consider many-body effects in the anion, let us first rule out other possible explanations for this inconsistency. We first consider the propagation of the uncertainties in the pair potentials. The theoretical error bars shown in Figures 4.10(b) and 4.11(b) were estimated by assuming an uncertainty of ± 3 cm^{-1} in the quantity $\varepsilon_a - \varepsilon_X$ for the pair potentials, as discussed above, and multiplying this by the number of Ar-X nearest neighbors. The uncertainty in the Ar-Ar potential, and that due to "relaxation" of the geometry is neglected. The shaded areas in the Figures represent the experimental uncertainties. The theoretical and experimental uncertainty regions show no overlap for $n>2$. If a *much* more conservative estimate of the uncertainties is desired, we can consider the individual uncertainties in the diatomic well depths, i.e. 9 cm^{-1} for ArBr and $ArBr^-$, and 18 cm^{-1} for ArI and ArI^-. Even in this case, the experimental and theoretical error ranges overlap only for Ar_2Br and $Ar_{2-3}I$.

Furthermore, because the trends in the size dependence of the observed *EA*s are so different from those of the calculated *EA*s, it does not seem possible to modify the pair

potentials so as to simultaneously account for all the experimental *EA*s. Any modification of the pair potentials would result in the same more or less linear trend in theoretical *EA*s.

One might also ask whether the population of local minima affects the trends in the experimental *EA*s. We can rule this out for $Ar_{2-3}X$, for which there is only one possible minimum geometry. For n=4 and 5 the only local minima give calculated *EA*s much *lower* than the experimental result. For some of the larger clusters there may be local minima that would be consistent with the experimental *EA*s. However, we know from the diatomic spectra[4] that the vibrational temperatures in the beam are on the order of 50 K. In light of this, a significant population of larger clusters occupying local minima several hundred cm^{-1} above the global minimum seems unlikely. For this reason, and because it is not possible to account for the observed *EA*s of the small clusters with alternate minima, it is very unlikely that population of local minima could be the sole explanation for the observed trends in the *EA*s.

Next we consider various non-additive terms in the potentials.

4.4.7. Many-Body Interactions

Non-additive (or many-body) interactions fall into three categories: those present in both the anion and the neutral, those unique to the neutral, and those unique to the anion. Many-body interactions present in both anion and neutral include dispersion (Axilrod-Teller) and exchange interactions. Interactions unique to the neutral include the many-body effects due to the open shell nature of the halogen atom, which have already been discussed in Section 4.4.5. Many-body effects unique to the anion are those involving the charge on the halide atom. These include non-additive induction effects, and

the interaction of the halide charge with multipole moments caused by exchange and dispersion interactions between pairs of argon atoms.

Since the experimental observable, the *EA*, depends on the *difference* between the anion and neutral potentials [see Equation (4.5)], we expect the many-body effects unique to the anion to be the most important in explaining the observed trends.

We will consider each non-additive effect in turn, incorporating it into our simulated annealing procedure to test its effect on cluster energetics at the minimum energy geometry.

4.4.7.1. Triple-Dipole Interaction

The leading term in the non-additive dispersion energy, the triple-dipole interaction, was first derived by Axilrod and Teller[46], and independently by Muto.[47] The form of the triple-dipole potential is, for three atoms i, j and k,

$$V_{ddd} = C_9 \frac{\left(3\cos\theta_i \cos\theta_j \cos\theta_k + 1\right)}{R_{ij}^3 R_{jk}^3 R_{ik}^3}, \tag{4.18}$$

where θ_i is the interior angle $\angle jik$, R_{ij} is the internuclear distance between atom i and atom j, and C_9 is a constant depending only on the identities of the three atoms. C_9 can be calculated using semi-empirical methods[48], or by fitting to *ab initio* calculations.[49] However, because such results are not available for the Ar_nX or Ar_nX^- systems considered here, we use the approximation to C_9 discussed by various authors,[48b,50]

$$C_9 = \frac{3}{2}\alpha_i\alpha_j\alpha_k \frac{\eta_i\eta_j\eta_k\left(\eta_i + \eta_j + \eta_k\right)}{\left(\eta_i + \eta_j\right)\left(\eta_j + \eta_k\right)\left(\eta_i + \eta_k\right)}, \tag{4.19}$$

where α_i and η_i are, respectively, the dipole polarizability and average excitation energy of atom i.

A simple approximation to η_i is, in atomic units,[50,51]

$$\eta_i = \left(\frac{N_i}{\alpha_i}\right)^{\frac{1}{2}}. \tag{4.20}$$

Here, N_i is an effective number of electrons for a given atom. Substituting (4.20) into (4.19) gives a three-body analogue of the Slater-Kirkwood formula[51] for the C_6 dispersion coefficient. In Koutselos and Mason's treatment[50a], which we follow here, N_i is treated as an empirical parameter determined from the corresponding C_6 two-body dispersion coefficient for like atoms. Furthermore, the values of N_i for the halide anions for which the C_6 coefficients are not known are assumed to be the same as those of the corresponding isoelectronic rare gases. Some theoretical and empirical justification of the approximations involved in this approach is given by Koutselos and Mason, who estimate an uncertainty of 5%-10% for C_9 coefficients determined in this way.[50a] The parameters N and α as well as the values of C_9 calculated from (4.19) and (4.20) are given in Table 4.6. It should be noted that Equations (4.18) and (4.19) are, strictly speaking, valid only for atoms in S-states.[46b,50a] In extending their use to P-state halogens we are implicitly neglecting the anisotropy of the halogen atom polarizability.

Table 4.6. Atomic dipole and quadrupole polarizabilities, effective numbers of electrons, and C_9 coeffecients for Ar-Ar-X interactions.

Atom	α (a_0^3)	C (a_0^5)	N	C_9 (eV·Å^9)
Ar	11.08[a]	27.11[d]	5.90[e]	-
Br^-	35.2[b]	164[d]	6.70[e]	127
Br	20.6[c]	-	6.2[e]	83
I^-	52.7[b]	254[d]	7.79[e]	179
I	36.1[c]	-	6.5[e]	129

References for Table 4.6

[a] R.R. Teachout and R.T. Pack, At. Data **3**, 195 (1971).

[b] H. Coker, J. Phys. Chem. **80**, 2078 (1976).

[c] *Handbook of Chemistry and Physics*, 74th ed. (CRC, Boca Raton, 1994), pp.10-198.

[d] M.V.K. Sastri, P.L. Narasimhulu and K.D. Sen, J. Chem. Phys. **80**, 584 (1984). Note that we use Buckingham's definition [Adv. Chem. Phys. **12**, 107 (1967)] of the quadrupole polarizability, C, which is equal to half of the quadrupole polarizability, α_q, used by Sastri *et al.* [See E.A. Gislason and M.S. Rajan, Chem. Phys. Lett. **50**. 251 (1977) and references therein for information on the various quadrupole polarizability conventions.]

[e] E.A. Mason and E.W. McDaniel, *Transport Properties of Ions in Gases* (Wiley, New York, 1988), pp. 533-4.

The triple-dipole interaction is repulsive for near equilateral geometries. In the case of Ar_2I^-, V_{ddd} at the equilibrium geometry is +8.1 cm^{-1}, and 6.3 cm^{-1} for Ar_2I. For Ar_2Br^- and Ar_2Br, the results are 9.0 cm^{-1} and 5.7 cm^{-1}, respectively. The larger values for the anionic clusters are mainly due to their greater polarizabilities. The net result, then, is a decrease in the calculated *EA* by about 2-3 cm^{-1} compared with the additive potentials. This effect is of the same order as the experimental uncertainty, but may be more significant for larger clusters. In the calculations below on clusters with n≥3, only the Ar-Ar-X triple-dipole interactions are included. The Ar-Ar-Ar interactions are neglected, because we expect their energies to be nearly equal in the anion and neutral.

It has been shown that higher-multipole three-body dispersion terms, such as the dipole-dipole-quadrupole (V_{ddq}) potential, may also contribute substantially to the three-body dispersion energy.[2] To ascertain their importance here, we used the formulae of Koutselos and Mason[50a] for the higher multipole coefficients, and the geometrical factors given by Bell[52] to estimate V_{ddq} for Ar_2I^- and Ar_2I. At the equilibrium geometries of the clusters determined with additive potentials, we obtain approximately 4 cm^{-1} for Ar_2I^- and 3 cm^{-1} for Ar_2I. The resulting 1 cm^{-1} shift in the *EA* is smaller than the experimental uncertainty. Therefore, V_{ddq} and all higher multipole three-body dispersion terms were neglected in subsequent calculations.

4.4.7.2. Three-Body Exchange

The second type of three-body interaction that occurs in both anion and neutral clusters is the three-body exchange interaction. This is caused by the exchange induced

electron charge distortion on a pair of atoms, which alters the pair's exchange interaction with a third atom. This effect is difficult to model without recourse to *ab initio* calculations, and has been the subject of some controversy.[2,3] As far as we are aware, such calculations are not available for the Ar_nBr/Ar_nBr^- or Ar_nI/Ar_nI^- systems studied here. However, we can get an idea of the magnitude of this effect from an *ab initio* calculation on Ar_3 by Chalasinski *et al.*[49] For equilateral Ar_3 at internuclear separations close to the equilibrium Ar_2 bond length, they find the sum of first and second order exchange three-body energies to be -1.5 cm^{-1}, or about 42% of the third order dispersion non-additive energy (+3.6 cm^{-1}), and of opposite sign. If we assume the exchange nonadditivity is a similar percentage of the dispersion nonadditivity in the Ar_nBr/Ar_nBr^- and Ar_nI/Ar_nI^- systems, we would anticipate a 2-4 cm^{-1} negative contribution to the binding energies, and an approximately 1 cm^{-1} difference between anion and neutral three-body exchange energies. Because this effect is expected to be small compared with our experimental uncertainties, and due to the practical difficulty of accurately modeling it, it will be neglected here.

4.4.7.3. Induction Non-Additivity

The anion pair potentials are dominated by induction. Likewise, we expect a rather large non-additive effect to arise from the interaction between multipole moments induced in the rare gas atoms by the halide charge. In addition, there is non-additivity due to the polarization of the halide atom itself. Because these effects are entirely absent in the neutral clusters (if we neglect the relatively small inductive effects due to the permanent

quadrupole moment of the neutral halogen), we expect the induction non-additivity to have a large effect on the *EA*.

A model for treating non-additive effects in systems of polarizable particles, first developed by Vesely[53], has been extended and used extensively by various workers in computer simulations of solvated ions[54] and electrons,[55] polar liquids,[56] and ionic clusters.[18,56d,57] Our adaptation of this model is as follows.

Each atom is characterized by a point charge (halide only) and point dipole and quadrupole polarizabilities (halide and rare gases), located at the nucleus. We assume that the induced dipole of an atom depends linearly on the electric field produced by the charges and multipoles of the other atoms *via* the dipole polarizability, α. We neglect the cubic dependence on the electric field due to the hyperpolarizability γ, and all higher terms. Likewise, we consider only quadrupoles induced by the field gradient due to the other atoms, characterized by the quadrupole polarizability C, neglecting the smaller contribution quadratic in the electric field via the dipole-quadrupole hyperpolarizability B,[58] and higher terms.

We also neglect the damping of the polarizabilities and charges at short range due to exchange or charge-transfer. Such effects are believed to be significant in the case of hydrogen bonding[56a] and in anions in ionic crystals.[59] However, they are probably less important in the weakly bound clusters considered here.

With these assumptions, the electric field at atom i is given by[60]

$$E_\alpha^{(i)} = \sum_{j\neq i}\left(-T_\alpha^{(ij)} q_j + T_{\alpha\beta}^{(ij)}\mu_\beta^{(j)} - \tfrac{1}{3}T_{\alpha\beta\gamma}^{(ij)}\Theta_{\beta\gamma}^{(j)}\right), \qquad (4.21)$$

and the electric field gradient is[60]

$$E_{\alpha\beta}^{(i)} = \sum_{j \neq i} \left(-T_{\alpha\beta}^{(ij)} q_j + T_{\alpha\beta\gamma}^{(ij)} \mu_\gamma^{(j)} - \tfrac{1}{3} T_{\alpha\beta\gamma\delta}^{(ij)} \Theta_{\gamma\delta}^{(j)} \right). \qquad (4.22)$$

Here, following the notation of Buckingham,[60] the subscripts α, β, γ and δ stand for any of the Cartesian components of a vector or tensor, and repeated Greek subscripts imply summation over the three components. The permanent electric charge is represented by q_i (-1 for the halide and 0 for the rare gases), and $\mu_\alpha^{(i)}$ and $\Theta_{\alpha\beta}^{(i)}$ are components of the induced dipole and quadrupole moments, respectively, at atom i. We use Buckingham's definition of the quadrupole moment as a traceless tensor.[60] The multipole interaction tensors are defined by $T_{\alpha\beta\ldots\nu}^{(ij)} = \nabla_\alpha \nabla_\beta \cdots \nabla_\nu \left(1/R_{ij} \right)$, where $\mathbf{R}_{ij}$ is the vector from atom j to atom i. The induced dipole at atom i is then given by[60]

$$\mu_\alpha^{(i)} = \alpha_i E_\alpha^{(i)}, \qquad (4.23)$$

and the induced quadrupole is[60]

$$\Theta_{\alpha\beta}^{(i)} = C_i E_{\alpha\beta}^{(i)}, \qquad (4.24)$$

where α_i and C_i are the dipole and quadrupole polarizabilities, respectively, of atom i. The values of α and C used here are given in Table 4.6.

At each time step in the simulated annealing procedure, the induced moments are calculated iteratively from Equations (4.21)-(4.24). At the first time step, the field and field gradient due to the halide permanent charge are initially calculated from (4.21) and (4.22). Then the induced moments are found from (4.23) and (4.24), and substituted back into (4.21) and (4.22). The process is repeated until the magnitudes of the induced moments do not change by more than one part in 10^{-10} with successive iterations. It is found that the moments converge about twice as fast if the individual moments are

immediately substituted into (4.21) and (4.22) for subsequent calculations during a given iteration, rather than "saved" until the next iteration. For subsequent MD time steps, the algorithm is initiated with the induced moments saved from the previous MD step. This saves some computer time.

The total induction energy is then given by

$$V_{ind.total} = V_{q\mu} + V_{q\Theta} + V_{\mu\mu} + V_{\mu\Theta} + V_{\Theta\Theta} + V_{self}\,, \tag{4.25}$$

where the first five terms on the right hand side are the charge-dipole, charge-quadrupole, dipole-dipole, dipole-quadrupole and quadrupole-quadrupole interaction energies. The final term is the energy required to create the induced dipoles and quadrupoles, given by[61,62]

$$V_{self} = \sum_i \left(\frac{\mu_\alpha^{(i)} \mu_\alpha^{(i)}}{2\alpha_i} + \frac{\Theta_{\alpha\beta}^{(i)} \Theta_{\alpha\beta}^{(i)}}{6C_i} \right), \tag{4.26}$$

where the sum runs over all atoms. By using (4.21)-(4.24) for one of each of the dipoles and quadrupoles in (4.26) and substituting the explicit expressions for the interaction energies[60] and (4.26) into (4.25), one can show that (4.25) simplifies to

$$V_{ind.total} = \tfrac{1}{2} V_{q\mu} + \tfrac{1}{2} V_{q\Theta} = \sum_i \sum_{j \neq i} q_i \left(-\tfrac{1}{2} T_\alpha^{(ij)} \mu_\alpha^{(j)} + \tfrac{1}{6} T_{\alpha\beta}^{(ij)} \Theta_{\alpha\beta}^{(j)} \right). \tag{4.27}$$

This equation gives the total induction energy of the cluster. However, part of this energy is already implicitly included in the Ar-X^- pair potential. In order to extract the non-additive portion, we calculate the induction energy for each Ar-X^- pair, neglecting the other Ar atoms in the cluster, using the same iterative method. The sum of the pair induction energies is then subtracted from (4.27) to give the non-additive induction energy:

$$V_{ind} = V_{ind,total} - V_{ind,pair}. \tag{4.28}$$

In practice, due to the computational "expense" of this iterative calculation, a simpler model was employed for the initial simulated annealing procedure. In the simpler model, the interaction energy between dipoles directly induced in the rare gas atoms by the halide charges is calculated.[63] The minimum energy geometries found with the simpler model were then optimized using the full iteratively calculated induction model described above. This simple induction model is described in detail in Appendix C.

The results of the calculation for Ar_2I^- and Ar_2Br^- show that the non-additive induction effect is indeed quite large. For Ar_2Br^-, for example, V_{ind} is 35.3 cm^{-1}. The result for Ar_2I^- is somewhat smaller, because of the larger Ar-X^- internuclear distance. The non-additive induction energy is always found to be positive, showing that it is dominated by the repulsion between adjacent induced multipoles on the Ar atoms. The dipole term of (4.27) contributes 32.7 cm^{-1} to the total in Ar_2Br^-, and the quadrupole term contributes 2.6 cm^{-1}. Thus, it does appear necessary to include the induced quadrupole effect, usually neglected in this type of simulation, for accurate calculation of the binding energies. The results for the larger clusters are discussed below.

4.4.7.4. Exchange and Dispersion Multipoles

As first described by Dick and Overhauser, [64] the exchange repulsion between two closed shell atoms produces a buildup of negative charge near the nuclei and a depletion of electron density between the nuclei. At large distances from the pair of atoms this distortion of the electron clouds is equivalent to a set of multipole moments, as discussed by Jansen. [65] If the atoms are identical, the first non-vanishing moment is a quadrupole.

There is also a quadrupole, of opposite sign, arising from the dispersion interaction between two atoms. At the usual van der Waals distances, the dispersion contribution is somewhat smaller than the exchange contribution.[66]

In the case of Ar_nBr^- and Ar_nI^-, a three-body effect then arises from the interaction of the halide charge with the Ar_2 exchange/dispersion multipoles. This is another type of many-body interaction that is present in the anionic but not in the neutral clusters, and is therefore expected to have a significant effect on the *EA*. As with induction, we expect the interaction of the permanent quadrupole of the neutral halogen atom with the exchange/dispersion moments to be negligible.

In their studies of the Ar_2-HCl, -DCl, and -HF systems, Hutson and coworkers[10] have found that the interaction of the exchange/dispersion quadrupole of the Ar_2 unit with the permanent multipoles of the HX molecule is quite important. This work was mainly concerned with the interpretation of the vibration-rotation spectra[7,8] of the clusters. However they also found the contribution to the binding energy to be significant. Chalasinski *et al.*[67] have found these conclusions about the importance of the exchange quadrupole effect on the Ar_2HX potential energy surfaces to be qualitatively consistent with their *ab initio* calculations. In recent work more closely related to our own, Burcl *et al.* have extracted information about the exchange multipole energy from *ab initio* calculations on Ar_2Cl^-.[68] These authors calculated this effect to be -12.8 cm^{-1} near the equilibrium geometry of Ar_2Cl^-. In this light we expect the exchange/dispersion multipole contribution to the non-additive binding energies of our Ar_nBr^- and Ar_nI^- clusters also to be quite significant.

Jansen derived a simple expression for the exchange quadrupole using an effective one electron model for the atomic charge distributions.[65] In this approach, the electronic charge density of an atom is approximated by a single Gaussian function,

$$\rho_i(\mathbf{r}) = -\frac{|e|\beta^3}{\pi^{3/2}} e^{-\beta^2|\mathbf{R}_i-\mathbf{r}|^2}, \tag{4.29}$$

where β is the Gaussian range parameter, $\mathbf{R}_i$ is the position of the nucleus of atom i, and $\mathbf{r}$ is the position of the effective electron. Then the atomic wave function is defined as

$$\varphi_i(\mathbf{r}) = \left|\frac{\rho_i(\mathbf{r})}{e}\right|^{1/2}. \tag{4.30}$$

The zero-order wavefunction of a pair of atoms, i and j, is taken to be the antisymmetrized product of the two atomic wavefunctions (normalized to 2):

$$\Psi_{ij}^0(\mathbf{r},\mathbf{r}') = \frac{1}{\left(1-S_{ij}^2\right)^{1/2}}\left[\varphi_i(\mathbf{r})\varphi_j(\mathbf{r}') - \varphi_i(\mathbf{r}')\varphi_j(\mathbf{r})\right]. \tag{4.31}$$

Here $\mathbf{r}$ and $\mathbf{r}'$ are the positions of the two electrons, and S_{ij} is the overlap integral, which for like atoms with Gaussian wavefunctions (4.30) is given by

$$S_{ij}^2 = \exp\left(-\beta^2 R_{ij}^2/2\right),$$

where R_{ij} is the internuclear separation between the atoms. Then, taking the expectation value of the quadrupole moment operator with wavefunction (4.31), a simple expression for the cylindrically symmetric exchange quadrupole is found:[65]

$$\Theta_{ex}\left(R_{ij}\right) = -\frac{|e|R_{ij}^2}{2}\left(\frac{S_{ij}^2}{1-S_{ij}^2}\right). \tag{4.32}$$

In Jansen's original treatment, the range parameter β was estimated from the long range dispersion interactions, and assumed to be valid for short range exchange interactions.

This method of estimating β is now believed to significantly overestimate the exchange quadrupole.[69,70] An approach that has been used to improve the accuracy of the model is to fit the one-electron functional form for the quantity of interest to the results of accurate *ab initio* calculations, to arrive at a more reasonable value of β.[10,69] Here, we shall use the value $\beta = 0.936$ Å^{-1}, derived in this way by Hutson and coworkers[10c] from an SCF calculation of the quadrupole moment of Ar_2.

The problem now arises of how to calculate the interaction energy of the exchange quadrupole with the halide charge. The simplest way is to represent the exchange charge distribution with a point quadrupole, calculated from (4.32), located at the midpoint between the two Ar atoms. The energy is then obtained from the standard expression for a charge-quadrupole interaction.[60] However, because the typical halide-Ar_2 distances in the clusters are on the same order as the Ar-Ar distance, the point quadrupole representation *overestimates* the magnitude of the interaction. The point quadrupole representation was used by Hutson *et al.* in their work on Ar_2-HCl and -DCl, and was found by them to somewhat overcorrect the pairwise additive potential.[10a,b] In more recent work on Ar_2-HF, Ernesti and Hutson[10c] proposed a distributed dipole representation: The Ar_2 exchange charge distribution is represented by opposed point dipole moments at the two Ar nuclei, parallel to the internuclear axis, with magnitudes chosen to give the same overall quadrupole moment as (4.32). Ernesti and Hutson found the distributed dipole representation superior to the point quadrupole representation for Ar_2-HF, but noted that it somewhat *underestimated* the electric field of the true charge distribution[10c]

The difficulty with both of these approaches arises from the use of a multipole representation at short range. Therefore, it seems logical to attempt a more direct calculation of the interaction of the exchange charge distribution with the halide charge. To do this, we form an effective charge density--the part of the charge density that contributes to the exchange quadrupole--by subtracting the atomic charge densities (4.29) from the charge density of the antisymmetrized wavefunction (4.31):

$$\begin{aligned}\rho_{eff}(\mathbf{r}) &= -|e|\int \left|\Psi_{ij}^{0}(\mathbf{r},\mathbf{r}')\right|^2 d\mathbf{r}' - \rho_i(\mathbf{r}) - \rho_j(\mathbf{r}) \\ &= -\frac{|e|S_{ij}^2}{1-S_{ij}^2}\left(\frac{\beta}{\pi^{1/2}}\right)^3\left[e^{-\beta^2|\mathbf{R}_i-\mathbf{r}|^2} + e^{-\beta^2|\mathbf{R}_j-\mathbf{r}|^2} - 2e^{-\beta^2|\mathbf{R}_C-\mathbf{r}|^2}\right]\end{aligned} \tag{4.33}$$

Here, $\mathbf{R}_C = \frac{1}{2}\left(\mathbf{R}_i + \mathbf{R}_j\right)$ is the midpoint between the two Ar nuclei. We see that the effective charge density is the sum of two negative Gaussian charge distributions located at the nuclei, and a positive Gaussian distribution, twice as large, at $\mathbf{R}_C$.[64] If we approximate the halide with a point charge at $\mathbf{R}_0$, the Coulomb interaction energy is then found to be[71]

$$V_{ec} = \sum_{i<j}\frac{e^2 S_{ij}^2}{1-S_{ij}^2}\left[\frac{\mathrm{erf}(\beta R_{i0})}{R_{i0}} + \frac{\mathrm{erf}(\beta R_{j0})}{R_{j0}} - 2\frac{\mathrm{erf}(\beta R_{C0})}{R_{C0}}\right], \tag{4.34}$$

where R_{i0}, R_{j0} and R_{C0} are the distances of the halide from the Ar nuclei and the midpoint between the nuclei, respectively, and i and j run over the Ar atoms. The error functions in (4.34) can be easily evaluated using standard subroutines.[72] In the limit $\beta R \to \infty$, $\mathrm{erf}(\beta R) \to 1$. So at long range, (4.34) is equivalent to the Coulomb interaction of the halide charge with negative point charges $\delta = -|e|S_{ij}^2/\left(1-S_{ij}^2\right)$ at the Ar nuclei, and

a positive point charge, $+2|\delta|$, at $\mathbf{R}_C$.[65] In order to prevent non-physical behavior of Equation (4.34) for small values of R_{C0} (near linear geometries), V_{ec} is cut off for Ar-Ar separations greater then a certain value, typically 6.5 Å.

We should discuss the approximations implicit in (4.34). First, the nuclear charges are, in effect, approximated by Gaussian distributions with the same β parameter as the atomic electron densities. Thus, effects of nuclear de-shielding are not included in (4.34). Second, the approximation of the halide by a point charge will underestimate the extent of overlap effects and hence tend to slightly overestimate the magnitude of the interaction energy. This deficiency could be corrected if more were known about the charge densities of the halide atoms. Finally, and most importantly, we are still working within the Gaussian one-electron approximation. A single Gaussian function is known to be a rather poor approximation to the true electron density of an atom.[70] This problem could be overcome by using a more accurate model of the Ar_2 charge distribution, such as the result of an *ab initio* calculation with Gaussian type basis functions. The method of Gaussian multipoles developed by Wheatley[71] could then be used to calculate the Coulomb energy. Despite the limitations of the present model, we nonetheless expect (4.34) to give a more accurate value of V_{ec} than either the point quadrupole or distributed dipole representations.

The three models of V_{ec} are compared for Ar_2X^- in Figure 4.12. In the Figure, the Ar-Ar distance is held constant at the equilibrium value of the Ar_2 molecule, and the Ar-Ar axis is kept perpendicular to $\mathbf{R}_{C0}$, as the halide-Ar_2 distance is varied. It can be seen that at large separations, the three models approach each other, as expected. However, at separations near the equilibrium structures of Ar_2I^- and Ar_2Br^-, the differences among the

three models are quite significant. For example, at R_{C0} = 3.61 Å, corresponding to Ar_2I^-, Equation (4.34) gives V_{ec} = -15.0 cm^{-1}, compared with -18.0 cm^{-1} for the point quadrupole model, and -12.6 cm^{-1} for the distributed dipole representation. The differences among the three models at the equilibrium R_{C0} of Ar_2Br^- (3.28 Å) are even more pronounced. We conclude that at the interatomic distances considered here, it is important to use an accurate representation of the exchange charge distribution to calculate V_{ec}. In the remainder of this work, we shall use (4.34) for V_{ec}.

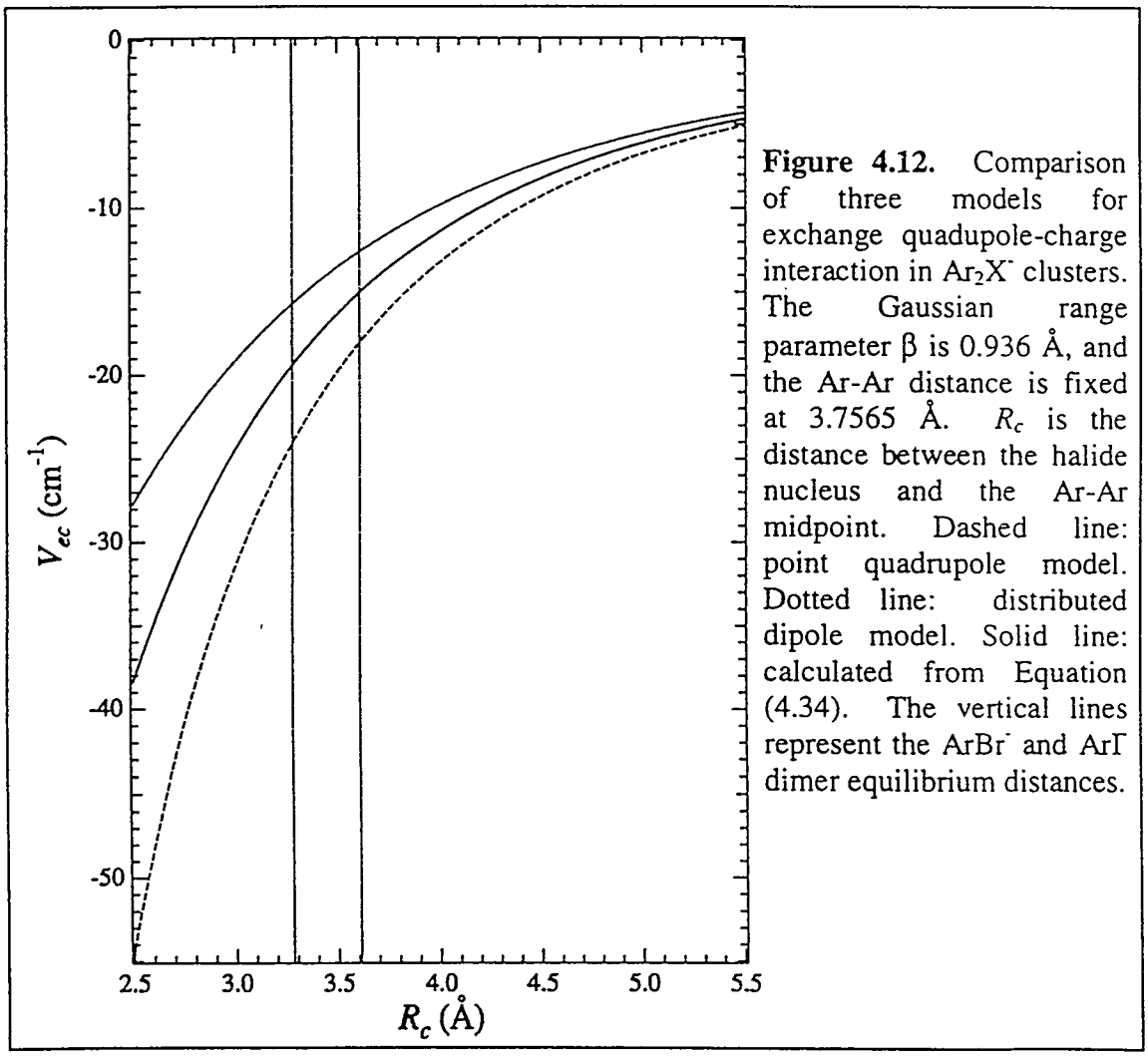

Figure 4.12. Comparison of three models for exchange quadupole-charge interaction in Ar_2X^- clusters. The Gaussian range parameter β is 0.936 Å, and the Ar-Ar distance is fixed at 3.7565 Å. R_c is the distance between the halide nucleus and the Ar-Ar midpoint. Dashed line: point quadrupole model. Dotted line: distributed dipole model. Solid line: calculated from Equation (4.34). The vertical lines represent the $ArBr^-$ and ArI^- dimer equilibrium distances.

We also need to consider the multipole moments induced in the rare gas atoms by dispersion. Hunt[73] has developed a model for the dispersion induced dipole and quadrupole moments in terms of atomic polarizabilities and dispersion coefficients. The average dipole moment induced on atom i by the dispersion interaction with other like atoms is given by:

$$\mu^{i,disp} = C_\mu \sum_{j \neq i} \frac{\hat{\mathbf{R}}_{ij}}{R_{ij}^7}, \text{ with } C_\mu = \frac{3}{2}\frac{C_6 B}{\alpha} \tag{4.35}$$

where $\hat{\mathbf{R}}_{ij}$ is the unit vector pointing from atom j to atom i, C_6 is the Ar -Ar dispersion coefficient, α is the dipole polarizability, and B is the dipole-quadrupole hyperpolarizability. The components of the dispersion induced quadrupole moment on atom i are given by:

$$\Theta_{\alpha\beta}^{i,disp} = -C_\Theta \sum_{j \neq i} \frac{T_{\alpha\beta}^{(ij)}}{2R_{ij}^3}, \text{ with } C_\Theta = \frac{1}{4}\frac{C_6 B}{\alpha} \tag{4.36}$$

where $T_{\alpha\beta}^{(ij)} = \nabla_\alpha \nabla_\beta \left(1/R_{ij}\right)$. For example, in the special case of two atoms lying on the Z-axis, the quadrupoles have cylindrical symmetry, with $\Theta_{zz}^{i,disp} = -C_\Theta / R_{ij}^6$, and $\Theta_{xx}^{i,disp} = \Theta_{yy}^{i,disp} = -\frac{1}{2}\Theta_{zz}^{i,disp}$. Following Ernesti and Hutson[10c] the values of C_μ and C_Θ were found using the C_6 constant from the Aziz HFDID1 potential,[29] and the ratio B/α from the calculation of Maroulis and Bishop.[74] We obtain $C_\mu = 1252\ ea_0^8$ and $C_\Theta = 208.6\ ea_0^8$. The total dispersion induced dipoles and quadrupoles are calculated from (4.35) and (4.36) for each Ar atom. Then the charge-dipole interaction energy, V_{ddis}, and the charge-quadrupole energy, V_{qdis}, are computed from the standard electrostatic formulae.[60] We denote the total charge-dispersion multipole energy by $V_{mdis} = V_{ddis} + V_{qdis}$.

We should note that this calculation is carried out independently of the non-additive induction energy calculation described in the previous section (4.4.7.3). Therefore, interactions between the electrostatically induced multipoles and the exchange/dispersion induced multipoles have been neglected. This is reasonable because the exchange/dispersion multipoles are about an order of magnitude smaller than the charge induced multipoles, and, therefore, the interactions of the exchange/dispersion multipoles with the charge are much larger than their interactions with the charge induced multipoles.

The charge-dipole energy, V_{ddis}, is generally positive and about 30% as large as V_{ec}. This proportion is qualitatively consistent with the calculations of Lacey and Byers Brown[66] and with the results of Ernesti and Hutson.[10c] V_{qdis} is negative, and only about 5% as large as V_{ddis}. For example, in Ar_2Br^-, the dispersion dipole energy is +6.0 cm^{-1}, and the dispersion quadrupole energy is -0.3 cm^{-1}. This may be compared with the exchange charge energy of -20.3 cm^{-1}. Thus we see that the dispersion dipole makes a non-negligible contribution to the non-additive energy, whereas the dispersion quadrupole could be neglected without any significant loss of accuracy.

Complete results for the larger clusters are discussed in the next section.

4.4.8. Electron Affinities Calculated with Many-Body Potentials

In order to assess the importance of the various many-body effects mentioned above, we re-optimized the minimum energy geometries found from the simulated annealing procedure with the additive potentials, successively adding the many-body terms, in order of their relative magnitudes.

The first non-additive term considered was the multipole induction energy; the anion potential was then found from

$$\varepsilon_a = \min\left(V_{ArX} + V_{ArAr} + V_{ind}\right) \tag{4.37a}$$

where the right hand terms are the pairwise additive argon-halide and argon-argon potentials, and the many-body multipole induction potential. The neutral potential was identical to that used in the calculation in Section 4.4.6:

$$\varepsilon_X = \min\left(V_{Ar_nX} + V_{ArAr}\right) \tag{4.38a}$$

where V_{Ar_nX} is the "matrix additive" X state potential described in Section 4.4.5. We refer to the electron affinities calculated from (4.37a), (4.38a) and (4.5) as EA_{ind}.

We next considered the effect of addition of the exchange-charge and multipole dispersion energies. The anion binding energies are then

$$\varepsilon_a = \min\left(V_{ArX} + V_{ArAr} + V_{ind} + V_{ec} + V_{mdis}\right), \tag{4.37b}$$

and the neutral binding energies still given by (4.38a). The electron affinities calculated from (4.37b), (4.38a) and (4.5) are referred to as $EA_{ind+ec+mdis}$.

Finally, the Axilrod-Teller term was included in both the anion and neutral potentials to give

$$\varepsilon_a = \min\left(V_{ArX} + V_{ArAr} + V_{ind} + V_{ec} + V_{mdis} + V_{at}^{anion}\right), \tag{4.37c}$$

$$\varepsilon_X = \min\left(V_{Ar_nX} + V_{ArAr} + V_{at}^{neutral}\right). \tag{4.38c}$$

The EAs calculated from (4.37c), (4.38c) and (4.5) are referred to as $EA_{ind+ec+mdis+at}$. The binding energies calculated from (4.37c) and (4.38c) and their components are shown graphically in Figures 4.13 and 4.14. The anion and neutral binding energies calculated from (4.37a), (4.37b) and (4.38a), and from (4.37c) and (4.38c) are given, along with the

corresponding zero-point energies, in Tables 4.7 and 4.8. The theoretical electron affinities EA_{ind}, $EA_{ind+ec+mdis}$ and $EA_{ind+ec+mdis+at}$ are given in Tables 4.9 and 4.10. The deviations of the theoretical *EA*s from the experimental values are shown in Figures 4.15 and 4.16.

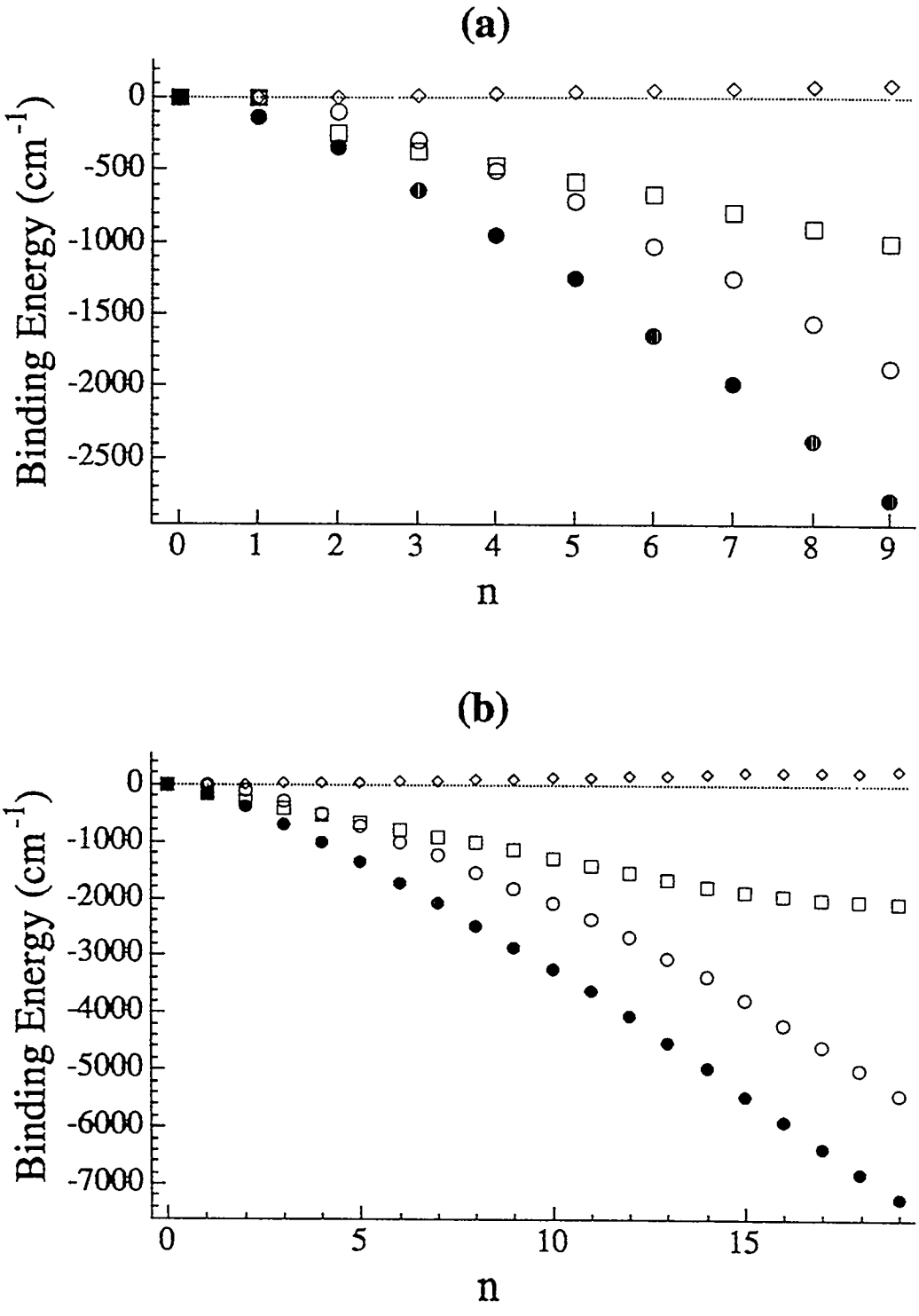

Figure 4.13. Calculated contributions to the *X* state binding energies (*BE*) of (a) Ar_nBr and (b) Ar_nI neutral clusters. Solid circles: total *BE*. Open squares: Ar-Br contribution. Open circles: Ar-Ar contribution. Open diamonds: Axilrod-Teller triple-dipole contribution.

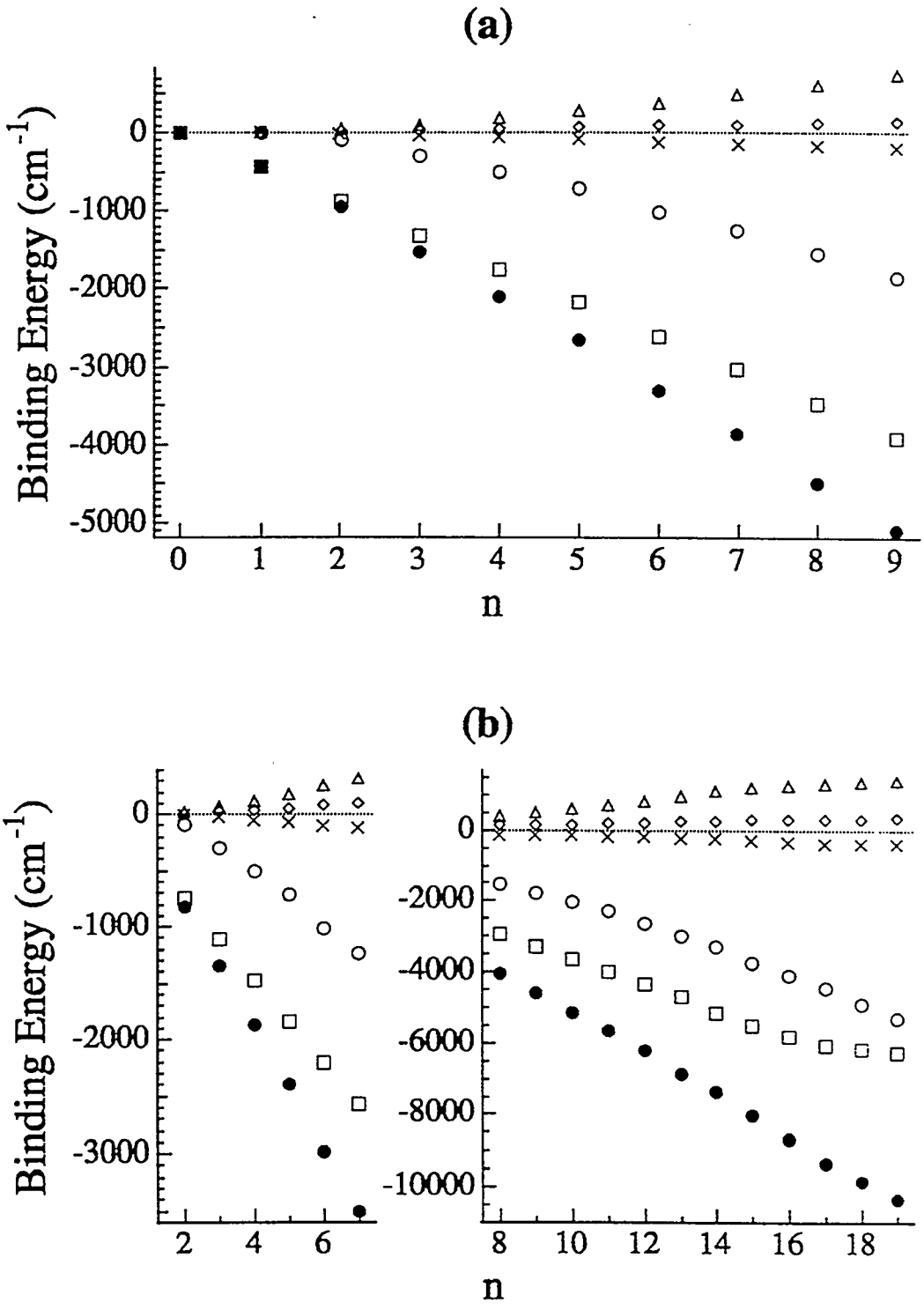

Figure 4.14. Calculated contributions to the binding energies of (a) Ar_nBr^- and (b) Ar_nI^- anions. Solid circles: total *BE*. Open squares: Ar-Br^- contribution. Open circles: Ar-Ar contribution. Open triangles: non-additive induction. Crosses: exchange charge and multipole dispersion. Open diamonds: triple-dipole dispersion.

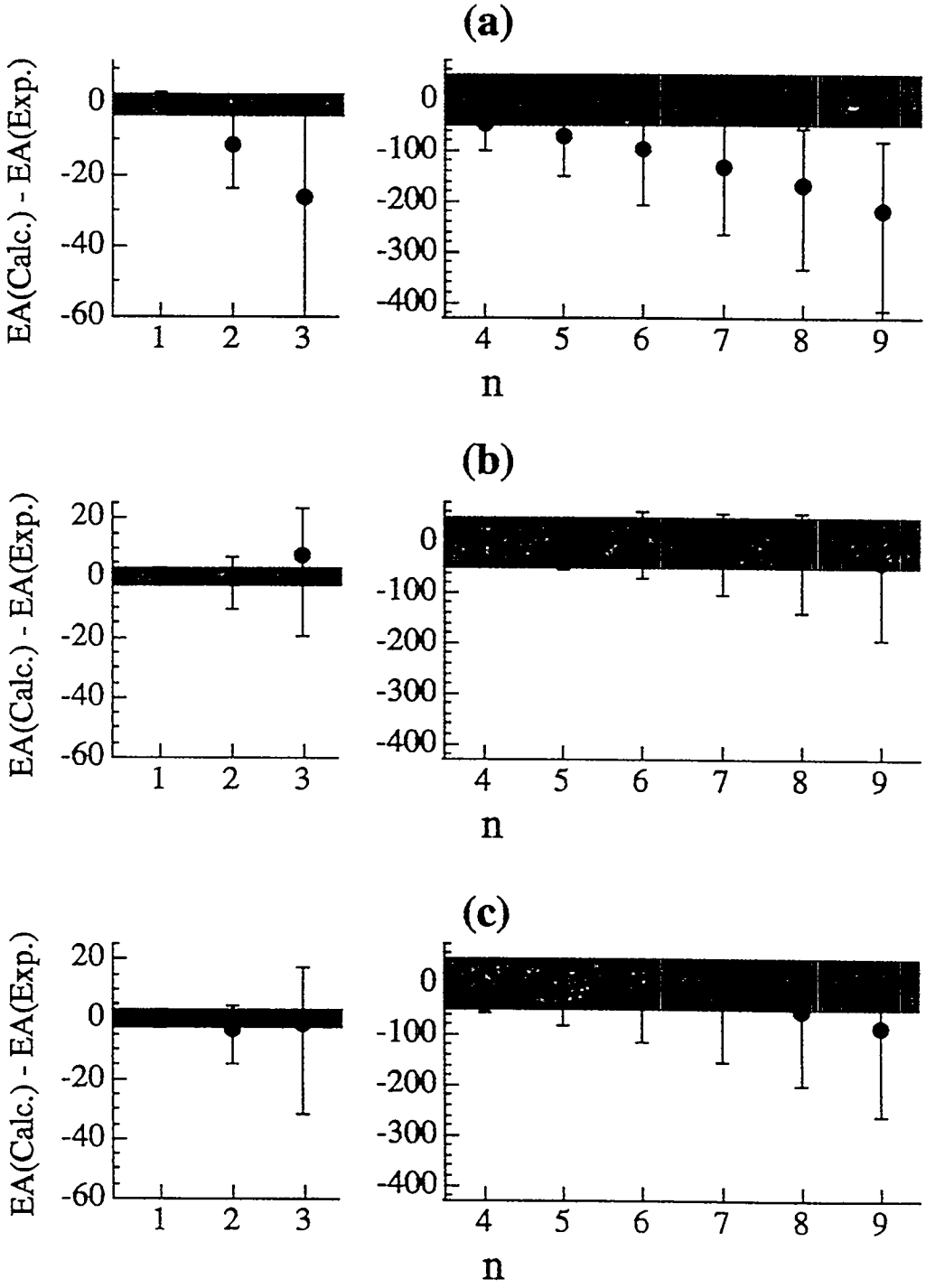

Figure 4.15. Differences between calculated and experimental *EA*s for Ar_nBr, plotted as a function of n, including various three body terms: (a) EA_{ind} - EA_{expt}, (b) $EA_{ind+ec+mdis}$ - EA_{expt}, (c) $EA_{ind+ec+mdit+at}$ - EA_{expt}. The shaded region represents the experimental uncertainty. The error bars represent the uncertainty in the calculated *EA*s.

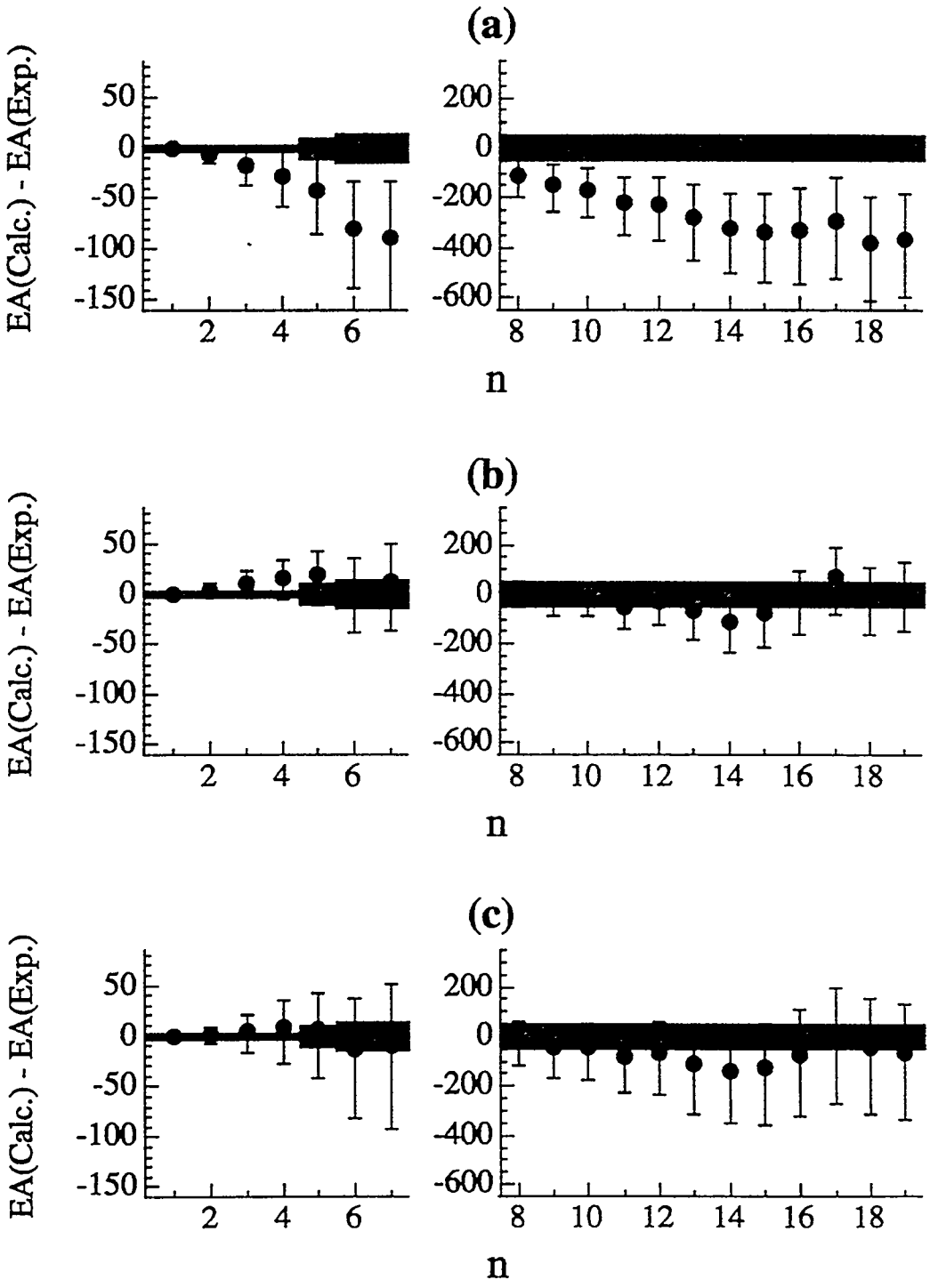

Figure 4.16. Differences between calculated and experimental *EA*s for Ar_nI, plotted as a function of n, including various three body terms: (a) EA_{ind} - EA_{expt}, (b) $EA_{ind+ec+mdis}$ - EA_{expt}, (c) $EA_{ind+ec+mdit+at}$ - EA_{expt}. The shaded region represents the experimental uncertainty. The error bars represent the uncertainty in the calculated *EA*s.

Table 4.7. Calculated Ar_nBr^- anion and Ar_nBr neutral X state binding energies and zero point energies, including non-additive terms. IND: non-additive induction. EC: exchange charge. MDIS: dispersion multipole. AT: Axilrod-Teller triple-dipole dispersion. Energies are in cm^{-1}.

n	IND		IND+EC+MDIS		IND+EC+MDIS +AT (anion)		AT (neutral)	
	ε_a	ω_0^a	ε_a	ω_0^a	ε_a	ω_0^a	ε_X	ω_0^X
1	438.8	20.7	438.8	20.7	438.8	20.7	133.1	15.6
2	942.4	53.7	955.9	56.1	946.8	55.0	343.7	41.7
3	1515.3	98.5	1555.7	105.1	1528.9	102.3	646.1	83.3
4	2081.3	142.6	2148.4	153.2	2103.9	148.8	950.0	124.7
5	2640.9	185.0	2734.3	199.0	2672.3	193.5	1255.5	164.7
6	3273.3	236.7	3402.4	255.2	3315.3	247.6	1643.6	207.8
7	3819.9	278.3	3974.4	300.4	3871.5	291.8	1978.7	262.0
8	4434.4	326.4	4620.2	351.6	4494.4	341.1	2385.1	307.4
9	5029.0	367.2	5238.7	395.2	5091.0	381.9	2783.7	349.6

Table 4.8. Calculated Ar_nI^- anion and Ar_nI neutral X state binding energies and zero point energies, including non-additive terms. IND: non-additive induction. EC: exchange charge. MDIS: dispersion multipole. AT: Axilrod-Teller triple-dipole dispersion. Energies are in cm^{-1}.

n	IND		IND+EC+MDIS		IND+EC+MDIS +AT (anion)		AT (neutral)	
	ε_a	ω_0^a	ε_a	ω_0^a	ε_a	ω_0^a	ε_X	ω_0^X
1	369.4	17.3	369.4	17.3	369.4	17.3	151.6	14.6
2	814.6	48.4	825.2	50.4	817.0	49.2	379.0	40.6
3	1338.4	90.2	1369.9	96.0	1345.7	92.6	699.7	81.4
4	1859.9	132.3	1912.3	141.5	1871.5	136.4	1020.8	122.1
5	2378.0	171.8	2451.2	184.0	2393.7	177.7	1342.6	161.6
6	2967.4	221.9	3066.3	239.4	2985.9	229.2	1740.5	203.9
7	3476.4	261.5	3595.8	280.9	3499.6	270.5	2080.2	251.2
8	4040.0	306.9	4179.8	330.7	4062.8	316.2	2482.0	292.6
9	4567.5	344.5	4721.2	366.2	4585.0	352.0	2859.1	325.1
10	5102.2	386.5	5285.1	415.0	5131.7	398.9	3246.9	378.1
11	5623.9	423.4	5821.0	456.8	5647.2	436.2	3627.1	415.7
12	6197.5	475.9	6427.8	510.4	6233.0	488.9	4047.9	470.3
13	6818.0	516.8	7068.8	554.8	6847.5	528.7	4533.9	513.4
14	7363.8	547.9	7625.0	593.4	7371.8	557.4	4957.0	554.8
15	8021.4	610.6	8322.8	649.3	8038.4	617.9	5465.2	619.4
16	8615.8	683.1	8973.2	725.1	8673.8	694.6	5908.9	667.5
17	9232.3	763.9	9647.9	808.9	9336.7	781.2	6361.9	747.4
18	9737.0	818.1	10159.9	864.8	9840.0	836.3	6810.4	802.8

19	10247.0	872.0	10676.5	921.2	10347.4	891.3	7254.8	855.9

Table 4.9. Ar_nBr electron affinities calculated with various non-additive terms. EA_{ind}: non-additive induction term only. $EA_{ind+ec+mdis}$: non-additive induction, exchange charge, and multipole dispersion terms. $EA_{ind+ec+mdis+at}$: induction, exchange charge, multipole dispersion, and triple-dipole dispersion terms. Energies are in cm^{-1}.

n	EA_{ind}	$EA_{ind+ec+mdis}$	$EA_{ind+ec+mdis+at}$
0	27129.2	27129.2	27129.2
1	27429.8	27429.8	27429.8
2	27710.8	27721.9	27718.9
3	27968.5	28002.2	27993.0
4	28218	28275	28259
5	28460	28540	28517
6	28681	28792	28761
7	28897	29029	28992
8	29088	29249	29205
9	29271	29453	29404

Table 4.10. Ar_nI electron affinities calculated with various non-additive terms. EA_{ind}: non-additive induction term only. $EA_{ind+ec+mdis}$: non-additive induction, exchange charge, and multipole dispersion terms. $EA_{ind+ec+mdis+at}$: induction, exchange charge, multipole dispersion, and triple-dipole dispersion terms. Energies are in cm^{-1}.

n	EA_{ind}	$EA_{ind+ec+mdis}$	$EA_{ind+ec+mdis+at}$
0	24673.3	24673.3	24673.3
1	24888.3	24888.3	24888.3
2	25095.4	25103.9	25102.6
3	25286.6	25312.3	25308.1
4	25474	25518	25510
5	25660	25721	25708
6	25828	25909	25893
7	25995	26095	26073
8	26138	26254	26231
9	26268	26400	26372
10	26410	26565	26537
11	26537	26701	26673
12	26680	26875	26840
13	26798	27011	26972
14	26903	27119	27086
15	27039	27302	27248
16	27156	27472	27411
17	27323	27693	27614
18	27336	27712	27669
19	27431	27811	27731

It is important to note that the many-body terms V_{ind}, V_{ec}, V_{mdis}, and V_{at}^{anion} all depend on the *absolute* values of the Ar-X^- distances. Therefore, we must consider the uncertainties in these terms due to the absolute uncertainty in R_m in the pair potentials, which, as mentioned in Section 4.4.1, is ±0.2 Å for both $ArBr^-$ and ArI^-. In order to estimate the uncertainties in the many-body terms, we calculated the changes in these terms in the Ar_2Br^- and Ar_2I^- systems with the Ar-Ar distance fixed at the Ar_2 equilibrium value, as the Ar-X^- distances were varied over ±0.2 Å about the ArX^- equilibrium values. To estimate these uncertainties in the larger clusters, the Ar_2X^- uncertainties were multiplied by the number of nearest neighbor Ar-Ar pairs in contact with the halide. The uncertainties introduced into the calculated EAs were found to be significantly larger than those due to the uncertainty in $\varepsilon_a - \varepsilon_X$ in the pair potentials. These uncertainties are displayed as error bars in Figures 4.15 and 4.16.

The addition of the many-body induction term significantly decreases the *EA* compared with the additive calculation. [Compare Figures 4.15(a) and 4.16(a) with Figures 4.10 and 4.11.] EA_{ind} is closer to the experimental values than EA_{add}, but is somewhat *overcorrected.* This is clearest in the Ar_nI clusters, in which EA_{ind} lies well below the experimental uncertainty region (shaded areas in Figures 4.15 and 4.16) for n=6, 7 and 9-19. Thus, the Ar_nI results clearly indicate the need for additional non-additive terms. In the Ar_nBr clusters, the experimental and theoretical *EA* uncertainty regions overlap except for n=8 and 9, but the EA_{ind} values are all systematically lower than the experimental *EA*s, again suggesting that induction effects alone decrease the electron affinity by too much.

Inclusion of V_{ec} and V_{mdis} in the calculation brings the theoretical *EA*s closer to the experimental results. [See Figures 4.15(b) and 4.16(b).] In the case of Ar_nBr, $EA_{ind+ec+mdis}$ lies within the experimental error bars in all cases except for Ar_3Br, which is overcorrected by about 9 cm^{-1}. But even in this case, the model potential and experimental uncertainty regions overlap. For the Ar_nI clusters, $EA_{ind+ec+mdis}$ is overcorrected by 3.8-20.1 cm^{-1} for n=2-5, which is outside the experimental error bars (the shaded region in Figures 4.15 and 4.16). For $6 \leq n \leq 19$, $EA_{ind+ec+mdis}$ lies within experimental uncertainties except for n=13-15 and n=17. However, as in the Ar_nBr clusters, the theoretical and experimental uncertainty ranges overlap in all cases for Ar_nI.

Inclusion of the Axilrod-Teller term brings the theoretical *EA*s closer to experiment for the smaller clusters, but overcorrects somewhat for some of the larger clusters. [See Figures 4.15(c) and 4.16(c).] Now for the Ar_nBr clusters $EA_{ind+ec+mdis+at}$ lies within the experimental uncertainties for all cases except n=9. For $Ar_{2-4}I$, addition of the Axilrod-Teller term brings the theoretical *EA* closer to the experimental result, but is still a few wavenumbers above the experimental error bars. For n=5-10, 17 and 18, the theoretical result is within experimental uncertainties, but lies below the uncertainty region for n=11-16 and 19. But, again, the theoretical and experimental error bars overlap in all cases. Thus, inclusion of the triple-dipole term appears to help somewhat for the smaller clusters, but, because of the uncertainties in V_{ind} and $V_{ec}+V_{mdis}$, it is not possible to draw definite conclusions about the importance of the Axilrod-Teller term from the present results. To do so would require more precise knowledge of r_m in the pair potentials.

4.5. Conclusions

We have obtained experimental electron affinities and electronic structure information from the ZEKE and PDTP spectra of $Ar_{2-9}Br^-$ and $Ar_{2-19}I^-$. We have compared these with electronic state splittings and *EA*s calculated from both pairwise additive and non-additive model potentials. The following conclusions can be drawn from this work.

(1) The first-order degenerate perturbation theory treatment of the open-shell neutral potentials described in Section 4.4.5 is accurate enough to account for the *X-I* electronic state splittings observed in $Ar_{2-3}Br$ and $Ar_{2-7}I$, within experimental uncertainties. (See Figure 4.8). However, this model somewhat overestimates the *X-II* state splittings for $Ar_{2-3}Br$ and $Ar_{2-3}I$, possibly indicating that the spin-orbit splitting decreases as Ar atoms are added around the halogen. (See Figure 4.9).

(2) A pairwise additive model of the anion potentials is completely inadequate to account for the experimentally measured *EA*s. (See Figures 4.10 and 4.11). Non-additive effects in the anion are clearly very important.

(3) The many-body induction effect is the most important non-additive effect in the anion potential. Inclusion of V_{ind} accounts for most of the discrepancy between the additive and experimental *EA*s, but somewhat overcorrects, especially in the case of Ar_nI. This result is consistent with the work of Berkowitz *et at.*[18] who found a non-additive inductive effect to be very important to model the experimental *BE*s of $Br^-(H_2O)_n$ clusters. We also note that, although not explicitly discussed by Bowen and coworkers in their paper on Ar_nO^- clusters,[16] it seems likely that the non-additive induction effect may in large part account for the non-linearity of the binding energies as a function of n observed by them for n<12. (See Figure 4 of Reference 16.)

(4) The exchange/dispersion multipole term also plays an important role. In both the $Ar_{2-9}Br^-$ and $Ar_{2-19}I^-$ clusters inclusion of the V_{ec} and V_{mdis} terms in the *EA* calculation brings the calculated EAs within the experimental error bars, when the uncertainty in the size of the induction effect due to the uncertainty of the pair potential bond length is taken into account.

(5) Inclusion of the triple-dipole dispersion effect in the anion and neutral potentials appears to slightly improve the agreement with experiment for the smaller clusters ($Ar_{2-5}Br^-$ and $Ar_{2-5}I^-$), but makes the fit slightly worse for the larger clusters. However, due to the uncertainties in r_m in the Ar-X^- pair potentials, nothing conclusive can be said about the role of the triple-dipole dispersion effect in these clusters on the basis of our results. Furthermore, we cannot draw any conclusions about the role of three-body exchange effects or higher-order multipole dispersion terms from the present work.

Overall, this type of detailed energetic study of many-body effects is complementary to studies of non-additive effects *via* high resolution spectroscopy. In this type of experiment we are able to directly measure the difference between anion and neutral binding energies, allowing a direct comparison of experimental observables with model potentials including non-additive effects. However, due to limited resolution and uncertainties in the pair potentials this experiment is not sensitive to the most subtle non-additive effects, such as the triple-dipole dispersion energy. This is in contrast to the high resolution spectroscopic studies of the Ar_2-HX systems,[3,7-10] which provide precise values of molecular constants. Comparison of such results with non-additive model potentials is more difficult, but can in principle provide more precise information on non-additive effects.

Further theoretical work needs to be done to interpret the vibrational structure observed in the smaller clusters ($Ar_{2-3}Br^-$ and $Ar_{2-3}I^-$) studied here. These spectra present an opportunity to test the various methods of dynamical calculations that have been developed for weakly bound clusters,[75] and such studies would be welcome.

In the future, we hope to observe the ZEKE spectrum of Ar_2Cl^-. This would allow direct comparison with the recently published *ab initio* study of this system by Burcl *et al.*[68] Also, the Ar_2Cl neutral cluster would present a more tractable problem for *ab initio* theorists than the larger halogen containing clusters studied in the present work.

4.6. Acknowledgments

This research is supported by the Air Force Office of Scientific Research under Contract No. F49620-94-1-0115, and by a Department of Defense Augmentation Award for Science and Engineering Research Training under Contract No. F49620-94-1-0412. We also wish to thank R. B. Gerber, P. Jungwirth, A. Krylov, C. C. Martens and Z. Li for helpful discussions and for providing molecular dynamics programs.

4.7. References for Chapter 4

[1] H. Margenau, and N. R. Kestner, *Theory of Intermolecular Forces* (Pergamon Press, Oxford, 1969), Ch. 5.

[2] W.J. Meath, and R.A. Aziz, Mol. Phys. **52**, 225 (1984).

[3] M.J. Elrod, and R.J. Saykally, Chem. Rev. **94**, 1975 (1994), and references therein.

[4] Y. Zhao, I. Yourshaw, G. Reiser, C.C. Arnold, and D.M. Neumark, J. Chem. Phys. **101**, 6538 (1994).

[5] Y. Zhao, C.C. Arnold, and D.M. Neumark, J. Chem. Soc. Faraday Trans. **89**, 1449 (1993).

[6] Y. Xu, W. Jäger, and M. C. L. Gerry, J. Chem. Phys. **100**, 4171 (1994).

[7] M.A. Suhm, and D.J. Nesbitt, Chem. Soc. Rev. **24**, 45 (1995); A. McIlroy, R. Lascola, C.M. Lovejoy, and D.J. Nesbitt, J. Phys. Chem. **95**, 2636 (1991).

[8] M.J. Elrod, J.G. Loeser, and R.J. Saykally, J. Chem. Phys. **98**, 5352 (1993).

[9] M. M. Szczesniak, G. Chalasinski, and P. Piecuch, J. Chem. Phys. **99**, 6732 (1993); S. M. Cybulski, M. M. Szczesniak, and G. Chalasinski, J. Chem. Phys. **101**, 10708 (1994).

[10] (a) A.R. Cooper, and J.M. Hutson, J. Chem. Phys. **98**, 5337 (1992); (b) M.J. Elrod, R.J. Saykally, A.R. Cooper, and J.M. Hutson, Mol. Phys. **81**, 579 (1994); (c) A. Ernesti, and J.M. Hutson, Phys. Rev. A **51**, 239 (1995).

[11] S. Martrenchard-Barra, C. Jouvet, C. Lardeux-Dedonder, and D. Solgadi, J. Chem. Phys. **98**, 5281 (1993).

[12] J.P. Visticot, P. de Pujo, J.M. Mestdagh, A. Lallement, J. Berlande, O. Sublemontier, P. Meynadier, and J. Cuvellier, J. Chem. Phys. **100**, 158 (1994).

[13] (a) C.R. Bieler, D.D. Evard, and K.C. Janda, J. Phys. Chem. **94**, 7452 (1990); (b) D.G. Jahn, S.G. Clement, and K.C. Janda, J. Chem. Phys. **101**, 283 (1994).

[14] T. Bürgi, T. Droz, and S. Leutwyler, Chem. Phys. Lett. **225**, 351 (1994).

[15] G. Markovich, S. Pollack, R. Giniger, and O. Cheshnovsky, J. Chem. Phys. **101**, 9344 (1994).

[16] S.T. Arnold, J.H. Hendricks, and K.H. Bowen, J. Chem. Phys. **102**, 39 (1995).

[17] D.W. Arnold, S.E. Bradforth, E.H. Kim, and D.M. Neumark, J. Chem. Phys. **102**, 3510 (1995); **97**, 9468 (1992).

[18] L.S. Sremaniak, L. Perera, and M.L. Berkowitz, Chem. Phys. Lett. **218**, 377 (1994).

[19] K. Müller-Dethlefs, M. Sander, and E.W. Schlag, Z. Naturforsch. Teil A **39**, 1089 (1984); Chem. Phys. Lett. **12**, 291 (1984); K. Müller-Dethlefs, and E.W. Schlag, Ann. Rev. Phys. Chem. **42**, 109 (1991).

[20](a) T.N. Kitsopoulos, I.M. Waller, J.G. Loeser, and D.M. Neumark, Chem. Phys. Lett. **159**, 300 (1989); (b) C.J. Chick, Y. Zhao, T.N. Kitsopoulos, and D.M. Neumark, J. Chem. Phys. **97**, 6121 (1992); (c) T.N. Kitsopoulos, Ph.D. Thesis, University of California, Berkeley, 1991; (d) C.C. Arnold, Ph.D. Thesis, University of California, Berkeley, 1994.

[21] J.M.B. Bakker, J. Phys. E **6**, 785 (1973).

[22] T. Baer, W.B. Peatman, and E.W. Schlag, Chem. Phys. Lett. **4**, 243 (1969); R. Spohr, P.M. Guyon, W.A. Chupka, and J. Berkowitz, Rev. Sci. Instrum. **42**, 1872 (1971).

[23] C. E. Moore, *Atomic Energy Levels*, v. I, Circ. Natl. Bur. Std. 467 (1949).

[24] H. Haberland, Z. Phys A. **307**, 35 (1982).

[a] C. Blondel, P. Cacciani, C. Delsart and R. Trainham, Phys. Rev. A **40**, 3698 (1989).

[b] C.E. Moore, *Atomic Energy Levels*, v. I, Circ. Natl. Bur. Std. 467 (1949).

[25] C. Blondel, P. Cacciani, C. Delsart, and R. Trainham, Phys. Rev. A **40**, 3698 (1989).

[26] H. Hotop, and W.C. Lineberger, J. Phys. Chem. Ref. Data **14**, 731 (1985).

[27] P. Casavecchia, G. He, R.K. Sparks, and Y.T. Lee, J. Chem. Phys. **75**, 710 (1981).

[28] (a) G. Liuti, and F. Pirani, Chem. Phys. Lett. **122**, 245 (1985); (b) D. Cappelletti, G. Liuti, and F. Pirani, Chem. Phys. Lett. **183**, 297 (1991); (c) R. Cambi, D. Cappelletti, G. Liuti, and F. Pirani, J. Chem. Phys. **95**, 1852 (1991).

[29] R.A. Aziz, and M.J. Slaman, Mol. Phys. **58**, 679 (1986). A more accurate Ar-Ar potential has recently been published--the HFDID1 potential of Aziz, which was fit to more recent spectroscopic data [R.A. Aziz, J. Chem. Phys. **99**, 4518 (1993)]. However it was felt that the simpler HFD-B2 potential is sufficiently accurate for the purposes of this work.

[30] Z. Li, A. Borrmann, and C.C. Martens, J. Chem. Phys. **97**, 7234 (1992); A. Borrmann, Z. Li, and C.C. Martens J. Chem. Phys. **98**, 8514 (1993).

[31] M. P. Allen, and D. J. Tildesley, *Computer Simulation of Liquids* (Oxford Univ. Press, Oxford, 1987), p. 231.

[32] P.R. Bevington, *Data Reduction and Error Analysis for the Physical Sciences* (McGraw-Hill, New York, 1969), Ch. 11.

[33] E.B. Wilson, J.C. Decius, and P.C. Cross, *Molecular Vibrations* (Dover, New York, 1955).

[34] D.O. Harris, G.G. Engerholm, and W.D. Gwinn, J. Chem. Phys. **43**, 1515 (1965).

[35] J.C. Light, I.P. Hamilton, and J.V. Lill, J. Chem. Phys. **82**, 1400 (1985).

[36] R.B. Metz, Ph.D. Thesis, University of California, Berkeley, 1991.

[37] W.G. Lawrence, and V.A. Apkarian, J. Chem. Phys. **101**, 1820 (1994). Note the misprint in Eqn. A1 of this paper: The matrix element $\langle \frac{1}{2},-\frac{1}{2}|H'|\frac{3}{2},\frac{1}{2}\rangle$ should read $\frac{1}{3}V_{01}-\frac{2}{3}V_{-10}$, in the notation of this reference.

[38] V. Aquilanti, G. Liuti, F. Pirani, and F. Vecchiocattivi, J. Chem. Soc. Faraday Trans. 2 **85**, 955 (1989).

[39] W.E. Baylis, J. Phys. B **10**, L477 (1977).

[40] D. L. Huestis, and N. E. Schlotter, J. Chem. Phys. **69**, 3100 (1978).

[41] F.O. Ellison, J. Am. Chem. Soc. **85**, 3540 (1963); J.C. Tully, J. Chem. Phys. **58**, 1396 (1973).

[42] (a) K.M. Sando, G.J. Erickson, and R.C. Binning Jr., J. Phys. B **12**, 2697 (1979); (b) G.J. Erickson, and K.M. Sando, Phys. Rev. A **22**, 1500 (1980).

[43] L.C. Balling, and J.J. Wright, J. Chem. Phys. **79**, 2941 (1983).

[44] J.A. Boatz, and M.E. Fajardo, J. Chem. Phys. **101**, 3472 (1994).

[45] (a) A.I. Krylov, R.B. Gerber, and V.A. Apkarian, Chem. Phys. **189**, 261 (1994); (b) A.I. Krylov, and R.B. Gerber, Chem. Phys. Lett. **231**, 395 (1994).

[46] B.M. Axilrod, and E. Teller, J. Chem. Phys. **11**, 299 (1943); B.M. Axilrod, J. Chem. Phys. **19**, 719 (1951).

[47] Y. Muto, Proc. Phys-Math. Soc. Japan **17**, 629 (1943).

[48] (a) K.T. Tang, J.M. Norbeck, and P.R. Certain, J. Chem. Phys. **64**, 3063 (1976); (b) D.J. Margoliash, T.R. Proctor, G.D. Zeiss, and W.J. Meath, Mol. Phys. **35**, 747 (1978); (c) A. Kumar, and W.J. Meath, Mol. Phys. **54**, 823 (1985).

[49] G. Chalasinski, M.M. Szczesniak, and S.M. Cybulski, J. Chem. Phys. **92**, 2481 (1990).

[50] (a) A.D. Koutselos, and E.A. Mason, J. Chem. Phys. **85**, 2159 (1986); (b) H.L. Kramer, and D.R. Herschbach, J. Chem. Phys. **53**, 2792 (1970); (c) K.T. Tang, Phys. Rev. **177**, 108 (1969).

[51] J.C. Slater, and J.G. Kirkwood, Phys. Rev. **37**, 682 (1931); K.S. Pitzer, Adv. Chem. Phys. **2**, 59 (1959); J.N. Wilson, J. Chem. Phys. **43**, 2564 (1965).

[52] R. J. Bell, J. Phys. B **3**, 751 (1970).

[53] F.J. Vesely, J. Comput. Phys. **24**, 361 (1977).

[54] E.L. Pollock, and B.J. Alder, Phys. Rev. Lett. **41**, 903 (1978).

[55] P. Stampfli, J. Chem. Phys. **101**, 6024 (1994).

[56] For example, water: (a) F.H. Stillinger, and C.W. David, J. Chem. Phys. **69**, 1473 (1978); (b) P. Barnes, J.L. Finney, J.D. Nicholas, and J.E. Quinn, Nature **282**, 459 (1979); (c) P. Ahlström, A. Wallqvist, S. Engström, and B. Jönsson, Mol. Phys. **68**, 563 (1989); (d) J. Caldwell, L.X. Dang, and P.A. Kollman, J. Am. Chem. Soc. **112**, 9144 (1990); acetone: (e) P. Jedlovszky, and G. Pálinkás, Mol Phys. **84**, 217 (1995).

[57] For example, (a) Xe_n^{++}: J.G. Gay, and B.J. Berne, Phys. Rev. Lett. **49**, 194 (1982); (b) $Na^+(H_2O)_n$: P. Perez, W.K. Lee, and E.W. Prohofsky, J. Chem. Phys., **79**, 388 (1983); (c) $X(H_2O)_n$ (X = halide and alkali ions): S. Sung, and P.C. Jordan, J. Chem. Phys. **85**, 4045

(1986); (d) $X_nBr_2^-$ (X = Ar, CO_2): L. Perera, and F.G. Amar, J. Chem. Phys. **90**, 7354 (1989); (e) $(H_2O)_nX$ (X = Na^+, Cl^-): L. Perera, and M.L. Berkowitz, J. Chem. Phys. **95**, 1954 (1991); **99**, 4236 (1993).

[58] The contribution of the term in *B* to the induced quadrupole moment is estimated to be approximately 20% of the *C* contribution, and of opposite sign, where we have used the values of *B* for Ar from Ref. 74.

[59] U.C. Dikshit, and M. Kumar, Physica Status Solidi B **165**, 599 (1991).

[60] A.D. Buckingham, Adv. Chem. Phys. **12**, 107 (1967).

[61] C.J.F. Böttcher, *Theory of Electric Polarization*, 2nd Ed., Vol. I (Elsevier, Amsterdam, 1973).

[62] J. Applequist, Chem. Phys. **85**, 279 (1984); **190**, 153 (1995).

[63] R.L. Asher, D.A. Micha, and P.J. Brucat, J. Chem. Phys. **96**, 7683 (1992).

[64] B.G. Dick, Jr., and A.W. Overhauser, Phys. Rev. **112**, 90 (1958).

[65] L. Jansen, Adv. Quantum Chem. **2**, 119 (1965).

[66] A.J. Lacey, and W. Byers Brown, Mol. Phys. **27**, 1013 (1974).

[67] (a) M.M. Szczesniak, G. Chalasinski, and P. Piecuch, J. Chem. Phys. **99**, 6732 (1993); (b) S.M. Cybulski, M.M. Szczesniak, and G. Chalasinski, J. Chem. Phys. **101**, 10708 (1994).

[68] R. Burcl, S.M. Cybulski, M.M. Szczesniak, and G. Chalasinski, J. Chem. Phys. **103**, 299 (1995).

[69] B. Guillot, R.D. Mountain, and G. Birnbaum, Mol. Phys. **64**, 747 (1988).

[70] W. Kolos, and A. Les, Int. J. Quantum Chem. **6**, 1101 (1972).

[71] R.J. Wheatley, Mol. Phys. **79**, 597 (1993); J. Comput. Chem. **15**, 1187 (1994).

[72] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes*, 2nd Ed. (Cambridge Univ. Press, Cambridge, 1992), p. 213.

[73] K.L.C. Hunt, Chem. Phys. Lett. **70**, 336 (1980).

[74] G. Maroulis, and D.M. Bishop, J. Phys. B. **18**, 4675 (1985).

[75] See, for example, (a) the vibrational SCF method: T.R. Horn, R.B. Gerber, and M.A. Ratner, J. Chem. Phys. **91**, 1813 (1989); T.R. Horn, R.B. Gerber, J.J. Valentini, and M.A. Ratner, J. Chem. Phys. **94**, 6728 (1991), (b) DVR method: R.M. Whitnell, and J.C. Light, J. Chem. Phys. **90**, 1774 (1989); D.M. Leitner, J.D. Doll, and R.M. Whitnell, J. Chem. Phys. **94**, 6644 (1991), (c) basis set expansion methods: A.R. Cooper, S. Jain, and J.M. Hutson, J. Chem. Phys. **98**, 2160 (1993).

Appendix A. Program for calculating vibrational states and Franck-Condon factors for highly anharmonic one-dimensional potentials based on the eigenfunctions of the Morse potential using the DVR method

In this Appendix we describe the computer program used to calculate the vibrational eigenvalues, eigenfunctions and Franck-Condon factors for simulating the rare-gas halide diatomic spectra described in Chapters 3 and 4. The program uses a matrix method (the discrete variable representation, or DVR, method) to solve the Hamiltonian, with the eigenfunctions of a Morse oscillator as a basis set. The use of a Morse eigenfunction basis set has advantages over the more commonly used oscillator basis set when one is dealing with highly anharmonic potentials, such as those of the van der Waals complexes of interest here.

A1. Brief description of the DVR method

In the DVR method, or in any matrix method, one first sets up the Schrödinger equation in a matrix representation

$$\mathbf{H}\cdot\mathbf{c} = E\mathbf{c}\,, \tag{A1}$$

where $\mathbf{H} = \mathbf{T} + \mathbf{V}$ is the Hamiltonian, $\mathbf{T}$ is the kinetic energy matrix and $\mathbf{V}$ is the potential energy matrix. The problem then is to solve for the eigenvalues E_n and eigenvectors $\mathbf{c}_n$.

To this one chooses a set of basis functions $|\phi_n\rangle$ which are solutions of

$$H_0|\phi_0\rangle = E_n^0|\phi_n\rangle\,, \tag{A2}$$

where

$$H_0 = T + V_0\,. \tag{A3}$$

V_0 is a reference potential, for which the eigenfunctions, $|\phi_n\rangle$, and the eigenvalues, E_n^0, are known.

Introducing the "difference potential," $\Delta V = V - V_0$, which is the difference between the potential of interest and the reference potential, the Hamiltonian becomes $H = H_0 + \Delta V$, and the matrix elements of $\mathbf{H}$ are

$$\begin{aligned}\langle\phi_n|H|\phi_m\rangle &= \langle\phi_n|H_0|\phi_m\rangle + \langle\phi_n|\Delta V|\phi_m\rangle \\ &= E_n^0 + \langle\phi_n|\Delta V|\phi_m\rangle\end{aligned} \quad . \tag{A5}$$

The Schrödinger equation then becomes

$$\left(\mathbf{E}^0 + \Delta\mathbf{V}\right)\cdot\mathbf{c} = E\mathbf{c}\,. \tag{A6}$$

In order to calculate the matrix elements of the difference potential, we then use the "transformation method" of Harris, Engerholm and Gwinn,[1] commonly known as the HEG method. The first step in this method is to find the matrix elements, $\langle\phi_n|X|\phi_m\rangle$, of the position operator, X, in our basis; *i.e.*, we find the transformation matrix $\mathbf{T}$ such that

$$\mathbf{T}^{-1}\cdot\mathbf{X}\cdot\mathbf{T} = \mathrm{diag}(R_i) \tag{A7}$$

The eigenvalues, R_i, of $\mathbf{X}$ are the DVR points. It is then a simple matter to calculate the $\Delta\mathbf{V}$ in the DVR basis, because the potential (and difference potential) are diagonal in the position representation, so that

$$\Delta\mathbf{V}_{\mathrm{DVR}} = \begin{bmatrix} \Delta V(R_0) & & & 0 \\ & \Delta V(R_1) & & \\ & & \ddots & \\ 0 & & & \Delta V(R_N) \end{bmatrix} . \tag{A8}$$

In order to solve Eq. (A6) , we must then transform $\mathbf{E}^0$ into the DVR basis using the transformation matrices from (A7)

$$\mathbf{E}^0_{\mathrm{DVR}} = \mathbf{T}^{-1} \cdot \mathbf{E}^0 \cdot \mathbf{T}. \tag{A9}$$

It can be shown[2] that the errors introduced in this approach are minimized if the matrix **X** is tridiagonal, as it is in the harmonic oscillator basis. For further details and listing of computer code for implementing the DVR method using a harmonic oscillator basis, see the dissertation of R. B. Metz.[3]

Once the eigenvectors $\mathbf{c}_n$ and the eigenvalues E_n are found by diagonalizing Eq. (A6) it is a simple matter to simulate the vibrational stick spectrum. The anion and neutral eigenvectors are calculated in the same basis, so the Franck-Condon factors (FCFs) are found from

$$\mathrm{FCF}\left(v_n^{an} \leftarrow v_m^{neut}\right) = \mathbf{c}_n^{-1} \cdot \mathbf{c}_m. \tag{A10}$$

The peak spacings are found from the differences in the anion and neutral eigenvalues, and then the FCFs are multiplied by a Boltzmann factor for an assumed anion vibrational temperature.

Functions of R are quite easy to calculate in the DVR basis. For example we find the rotational constant for a given vibrational level from

$$B_v = \frac{\hbar^2}{2\mu} \cdot \frac{1}{\langle v | R^2 | v \rangle} = \frac{\hbar^2}{2\mu} \cdot \frac{1}{\mathbf{c}_v^{-1} \cdot \mathbf{R} \cdot \mathbf{R} \cdot \mathbf{c}_v}, \tag{A11}$$

where **R** is the diagonal matrix of DVR points diag(R_i).

A2. How to implement DVR using Morse basis functions

The implementation of the DVR method with a harmonic oscillator basis is straightforward, because the matrix elements of the position operator have a simple analytical form.[3] The eigenfunctions of the Morse oscillator are more complicated, and

no analytical form exists for $\langle \phi_n | X | \phi_m \rangle$. However, because the eigenfunctions of the Morse oscillator are a closer approximation to the eigenfunctions of the eigenfunctions of a diatomic van der Waals molecule, it is worthwhile investigating the problem of using the Morse eigenfunctions in the DVR scheme because faster convergence may be expected. This problem was considered, and solved, by Greenawalt and Dickinson in 1969.[4] Here we describe their approach to the problem, and our implementation.

The Morse potential[5] is given by

$$V(R) = D_e\left(1 - \exp\left(-a(R - R_e)\right)\right)^2 \tag{A12}$$

where D_e is the well depth, R_e is the bond length, and a is a parameter that determines shape of the potential. Note that in this Appendix, we define the parameter a in the Morse potential such that it has units of inverse length, whereas the β parameter of the Morse portion of the MMSV potential [See Eq. (2.1)] is unitless. We also define the quantity

$$t = \frac{(2\mu D_e)^{1/2}}{\hbar a} \tag{A13}$$

where μ is the reduced mass. The number of eigenvalues of the Morse potential is the number of integers from zero to $t - \frac{1}{2}$. The eigenfunctions of the Morse potential are then given by

$$\phi_n(y) = N_n^{-1/2} y^{t-n-\frac{1}{2}} \exp\left(-\tfrac{y}{2}\right) L_{2t-n-1}^{2t-2n-1}(y), \tag{A14}$$

with

$$y = 2t \exp\left(-a(R - R_e)\right), \tag{A15}$$

where the $L_n^m(y)$ are the Laguerre polynomials, and N_n is a normalization factor given by

$$N_n = \frac{\Gamma(n+1)\Gamma(2t-n)\Gamma(2t-2n-1)}{a\Gamma(2t-n)}. \qquad \text{(A16)}$$

Because no analytical form exists for $\langle \phi_n | X | \phi_m \rangle$ in the Morse basis, we use a generalization of the HEG transformation method,[6] as follows. First we choose a function, $u(R)$, such that the elements $\langle \phi_n | u(R) | \phi_m \rangle$ constitute a tridiagonal matrix. Then we diagonalize the matrix $\mathbf{u}$ to find the transformation matrix $\mathbf{T}$ such that $\mathbf{T}^{-1} \cdot \mathbf{u} \cdot \mathbf{T} = \mathrm{diag}(\lambda_i)$. The DVR points are then found by inverting the function $\lambda_i = u(R_i)$, and the rest of the calculation proceeds as above from Eq. (A8). Greenawalt and Dickenson[4] showed that for the Morse basis, the function

$$u(R) = \frac{1}{y} = \frac{1}{2t \exp(-a(R - R_e))} \qquad \text{(A17)}$$

gives a tridiagonal matrix, with elements

$$\langle \phi_n | u(R) | \phi_m \rangle = \delta_{n,m} \frac{t}{2(t-n)(t-n-1)} + \delta_{n,m-1} \left(\frac{N_m}{N_n} \right)^{1/2}, \quad n \le m. \qquad \text{(A18)}$$

Note that it is not necessary to calculate the Laguerre polynomials in Eq. (A14) in order to find the matrix elements from Eq. (A18). Also note that the ratio of the two normalization constants in (A18) can be simplified so that one need not evaluate the gamma functions in Eq. (A16). Therefore computation of the matrix elements of X in the Morse basis is not significantly more time-consuming than in the harmonic oscillator basis.

A3. Choosing the parameters for the Morse basis

In this section we describe general guidelines for choosing the optimal Morse basis set parameters that we have found effective in our calculations with the RgX anion and neutral MMSV potentials. Note that these are only guidelines, and some trial and error may be necessary to find the best basis set parameters for other potential forms.

First, we set R_e of the Morse potential equal to the bond length of the smaller of the neutral or anion MMSV potentials. Next we choose the desired number of basis functions, N. Twenty-five to 75 basis functions were found to be sufficient for most purposes. We then choose the parameter t [which is roughly equal to the total number of bound vibrational states of the Morse potential; see Eq. (A13)] so that the shape of the Morse eigenfunctions include functions with shapes similar to those expected for the range of vibrational levels of the model potential we wish to calculate. To do this, we set

$$\frac{t}{N} = \frac{\text{Estimated total number of MMSV eigenvalues}}{\text{Number of MMSV eigenvalues to calculate}}. \qquad \text{(A19)}$$

The total number of MMSV eigenvalues is estimated by using $(\beta_1 + \beta_2)/(2R_m)$ in place of a in Eq. (A13)--where β_1, β_2, and R_m are the MMSV parameters as defined in Eq. (2.1)-- along with the well depth of the MMSV potential, and μ calculated from R_m of the MMSV potential. If one chooses t by trial and error instead, as is sometimes necessary, one must remember that the quantity $t - \frac{1}{2}$ must be larger than the number of basis functions, N. Next one chooses the Morse potential force constant

$$k = \left.\frac{d^2V(R)}{dR^2}\right|_{R_e} = 2aD_e. \qquad \text{(A20)}$$

Finally, using these values for t and k, one calculates a and D_e from Eqs. (A13) and (A20).

As written, the program automatically computes the Morse basis set parameters based on the desired number of eigenvalues to calculate and the number of basis functions used according to the above considerations if one uses the MMSV potential form. If one uses a different potential form, one must provide the Morse parameters R_m, a, and D_e in the input file to the program.

A4. Comparison of Morse and harmonic oscillator basis set convergence for a highly anharmonic potential

In this section we test the performance of the Morse DVR program by comparing its convergence behavior with a harmonic oscillator based DVR program for the MMSV potential for the $I\frac{3}{2}$ state of KrI. This potential has a well depth of 16.7 meV and supports 18 bound vibrational states. See Ref. 7 for further details of the KrI potential. The program was tested by calculating the eigenvalues using both the Morse and harmonic basis sets, increasing the basis set size, N, by 5 with each calculation. The calculation was performed wit The results for v=1, 5, 10 and 15 are shown in Figure A1 on the following page. The ordinate of each graph is the relative difference between the eigenvalue calculated with a given number of basis functions, and that calculated with 5 fewer basis functions.

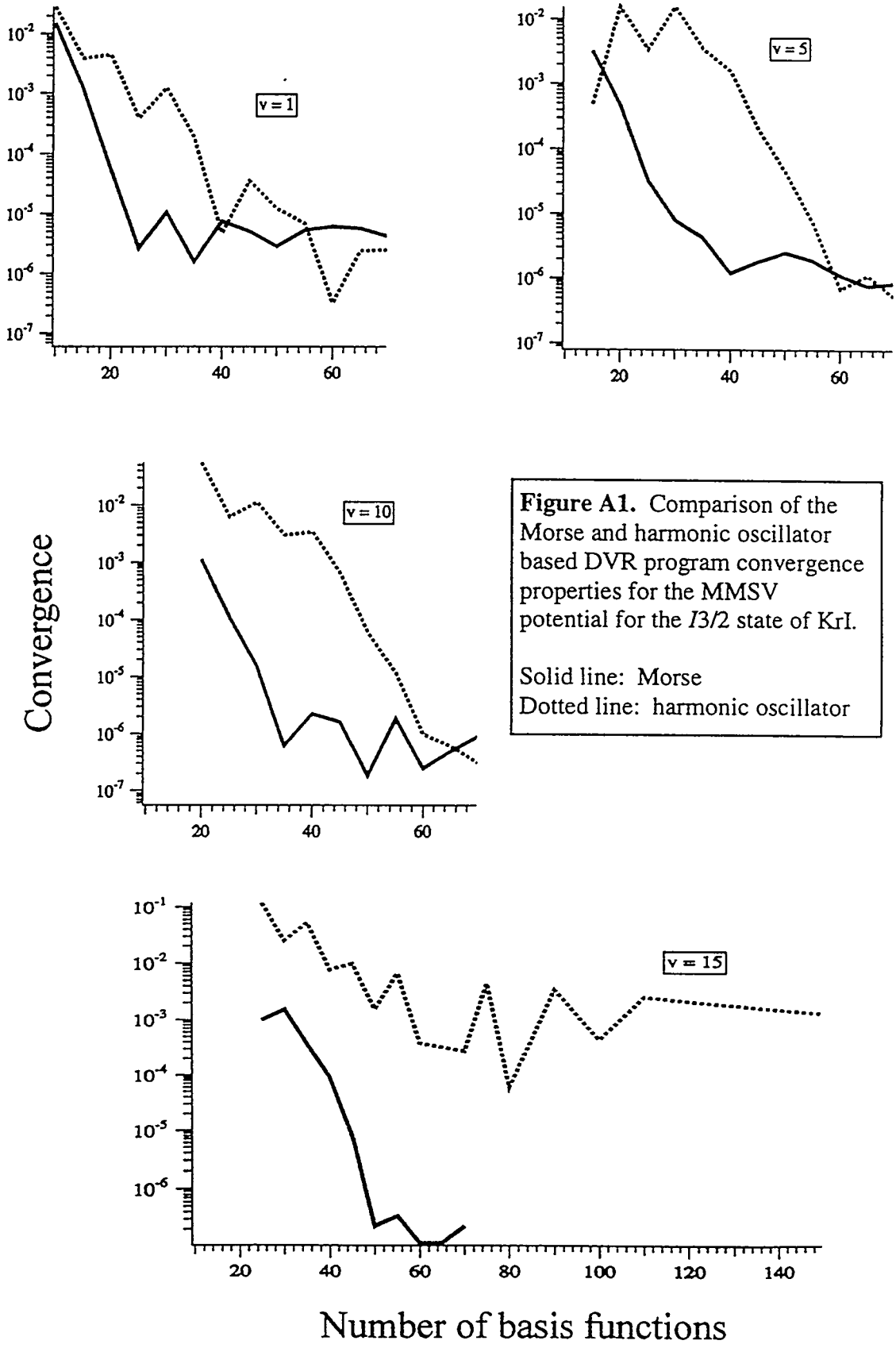

Figure A1. Comparison of the Morse and harmonic oscillator based DVR program convergence properties for the MMSV potential for the *I*3/2 state of KrI.

Solid line: Morse
Dotted line: harmonic oscillator

This number gives an indication of the convergence of the program. Also, the exponent of the ordinate is an approximation to the number of significant figures of the calculated eigenvalues. We see that for v = 1, 5 and 10, the Morse program converges with 25-35 basis functions, whereas, the harmonic oscillator program requires 40-60 basis functions to converge. In the case of v = 15, the Morse program converges with 50 basis functions, but the harmonic oscillator program does not converge with up to 150 basis functions, demonstrating the inadequacy of the harmonic oscillator model for treating high vibrational levels of anharmonic oscillators.

In general, then, it is found that the Morse program requires nearly a factor of 2 fewer basis functions than the harmonic oscillator program for satisfactory performance with anharmonic potentials. This is made more significant when one considers that the speed of the "rate limiting step" in the calculation, solution for the eigenvalues and eigenvectors of Eq. (A6), is proportional to the third power of the number of basis functions.[8] Therefore the use of the Morse basis offers an approximately eight fold increase in computing speed over the harmonic oscillator basis.

A5. Documentation of the Morse DVR program "idvr": example of the $X\,^{1}_{2}$ state of XeBr

In this section we demonstrate the how to use the interactive Morse DVR program "idvr" for fitting diatomic ZEKE spectra.

A5.1. The input files

A sample input file for fitting the XeBr $X\frac{1}{2}$ state spectrum is shown below. Note that the line numbers in boldface are for reference only and are not included in the actual input file. The file is named "xebr_x_dvrin."

```
1:  xebr_x_dvrpar
2:  xebr_x_exp
3:  xebr_x_wf
4:  xebr_x_stx
5:  xebr_x_bv
6:  xebr_x_gau
7:  xebr_x_zke
8:  3.       !* FWHM in cm-1 for Gaussian convolution
9:  8.5      !* FWHM in cm-1 for ZEKE convolution
10: 45       !* number of basis functions
11: 10       !* number of neutral eigenvalues to calculate
12: 6        !* number of anion eigenvalues to calculate
13: 27890.   !* origin (cm-1)
14: 70.      !* vibrational temp (K)
15: 0.00     !* baseline for ZEKE conv. (as fraction of highest peak)
```

The first 7 lines are the names of files used by the program. Lines **1** and **2** are input files which are not modified by the program. Line **1** is the name of a file which contains the potential and basis set parameters. (See below for a sample parameter file.) Line **2** is the name of the file containing the experimental spectrum that is being fit. This file must be in wavenumber format. It is used in order to calculate the chi-square goodness of fit function for comparison with the model spectrum. Lines **3-7** list the names of the output files. Line **3** names the file to save the DVR points and eigenvectors for the anion and neutral. The eigenvectors should be examined to ensure proper convergence: for each vibrational level, the eigenvector should be close to 0 at the smallest and largest DVR points. If this is not the case, it is necessary to either adjust the basis set parameters or use a larger number of basis functions. Line **4** is the name of the file to save the vibrational stick spectrum in wavenumber format. Line **5** is the name of the file to save

the rotational constants. The rotational constants are listed in this file in the same order as the vibrational sticks, and correspond to the rotational constants for the neutral vibrational level corresponding to each vibrational stick. Line **6** is the output file for a spectrum consisting of the vibrational sticks convoluted a Gaussian function with the FWHM (full width at half maximum) given in line **8**. Line **7** is the output file for the stick spectrum convoluted with the asymmetric ZEKE line shape, with the FWHM given in line **9**. See Eq. (4.11) of Chapter 4 for the form of the ZEKE line shape. Both of the convoluted spectra are output in wavenumber format. Line **10** is the number of basis functions to use in the calculation. Lines **11** and **12**, respectively, are the number of neutral and anion eigenvalues one wishes to calculate. Line **13** is the position of the vibrational origin, in wavenumbers. Line **14** is the anion vibrational temperature. Line **15** is a baseline, expressed as a fraction of the maximum peak, which may be added to the spectrum convoluted with the ZEKE line shape.

The parameter file "xebr_x_dvrpar" is listed below. Again, the boldface line numbers to not appear in the actual file.

```
1:  131.30, 79.909
2:  3.90, 1.3, 30.
3:  2, 2
4:  0.12692, 3.81, 3.50, 5.30, 1.03, 1.60, 228.3, 2135.2
5:  0.03153, 3.82, 4.35, 6.15, 1.01, 2.00, 4060., 39962.
```

Line **1** lists the atomic masses in amu. Line **2** lists the Morse basis parameters R_e, a, and t. If one uses the MMSV potential form, these parameters are not used by the program, but are calculated directly from the MMSV potential parameters. See Section A3, above, for a discussion of the choice of the basis parameters. Line **3** contains integers which specify which potential form to use for the neutral and anion. See the source code listing

of the subroutine "poten" in Section A6.6, below, for a listing of the available potential forms, and their associated parameters. Lines **4** and **5** list the potential parameters of the for the neutral and anion potentials, respectively. In this example the potentials are of the MMSV form. The order and units of the MMSV parameters in each line are:

ε (eV)
R_m (Å)
β_1 (unitless)
β_2 (unitless)
x_1
x_2
C_6/ε (Å^6) *or* B_4/ε (Å^4)
C_8/ε (Å^8) *or* B_6/ε (Å^6)

Note that the values of the dispersion coefficients input to the program are divided by the well depth, so the numbers shown in the sample input file differ from the parameters given in Table 3.4 of Chapter 4. *Note also that because of this quirk in the program one must change the dispersion inputs whenever one changes the well depth.* Refer to the source code listing in Section A6.6 for the order and units of the parameters for other potential forms.

A5.2. Running the program

An example of running the Morse DVR program "idvr" using the input files shown in the previous section is given below. Because this is an interactive program the input and output files names are input in response to prompts from the program rather than using pipes in the customary Unix style. The user input is shown in boldface.

```
> idvr

Interactive Morse DVR program - Version 1.01, Released March 1998
Copyright 1998, Anion ZEKE Group, Neumark Group,
Department of Chemistry, UC Berkeley
```

```
Enter input filename :
xebr_x_dvrin
Enter output filename :
xebr_x_dvrout

Gaussian convolution   781 points saved to : xebr_x_gau

ZEKE convolution   275 points saved to : xebr_x_zke


     Anion,  Neutral parameters
  1 :    0.12692000000000    3.15300000000000D-02
  2 :     3.8100000000000    3.8200000000000
  3 :     3.5000000000000    4.3500000000000
  4 :     5.3000000000000    6.1500000000000
  5 :     1.0300000000000    1.0100000000000
  6 :     1.6000000000000    2.0000000000000
  7 :     228.30000000000    4060.0000000000
  8 :     2135.2000000000    39962.000000000
Vibrational temperature :     70.000000000000
ZEKE FWHM (cm-1)        :     8.5000000000000
Baseline for ZEKE conv. :   0.
Chi-square from ZEKE vibrational convolution : 0.860324E-01
-----------------------------------------------------------------------
Enter "a/A" to change/optimize an anion parameter
      "n/N" to change/optimize a neutral parameter
      "v/V" to change/optimize the vib. temperature
      "f/F" to change/optimize the ZEKE FWHM
      "b/B" to change/optimize the baseline
      "q" to quit idvr
```

At this point, the user may examine the convoluted spectra, and the output file "xebr_x_dvrout," which lists the eigenvalues and Franck-Condon factors. If the fit is not satisfactory, one can modify the parameters as often as desired before exiting the program. Note that the modified parameters are *not* saved in the parameter file.

A5.3. Outline of the program

In the table below we list the subroutines and functions that are used by the "idvr" interactive Morse DVR fitting program sorted according to which file they reside in.

Table A1. Subroutines and functions used by the "idvr" program.

Source File	Subroutine or Function	Description
idvr.f	idvr	main program
	umat	sets up the **u** matrix using Eq. (A18)
	rcalc	calculates the DVR points R_i
	initpars	reads in the potential parameters
	diffpot	calculates the difference potential matrix [Eq. (A8)]
	morseig	calculates the matrix $\mathbf{E}^0_{DVR}$ from Eq. (A9)
	hcalc	sets up the total Hamiltonian matrix **H**
	setbasis	chooses the Morse basis set parameters according to the considerations of Section A3
	calcfcf	calculates the Franck-Condon factors
	bvcalc	calculates rotational constants for individual vibrational levels from Eq. (A11)
rsb.f	rsb	EISPACK subroutine for computing the eigenvalues and eigenvectors of a real (double-precision) symmetric band matrix
rs.f	rs	EISPACK subroutine for computing the eigenvalues and eigenvectors of a real (double-precision) symmetric matrix
matrix.f	showarr4	subroutine to print out a matrix
poten2.f	poten	calculates the potential for a specified potential form
	ms4	Maitland-Smith n-4 anion potential function
	anmmsv	anion MMSV potential function
	morse	Morse potential function
	bucky	Buckingham exp-n potential function
	anmmsvus	unscaled anion MMSV potential function
	ms6	Maitland-Smith n-6 neutral potential function

	neummsv	neutral MMSV potential function
	msv	Morse-Spline-van der Waals potential function for neutral
	aqx	Aquilanti potential[9] for the RgX $X\,^{1}_{2}$ state
	aqi	Aquilanti potential[9] for the RgX $I\,^{3}_{2}$ state
	neummsvus	unscaled neutral MMSV potential function
	hfd_c	Aziz HFD-C potential[10]
	hfd_b	Aziz HFD-B2 potential[10]
convol.f	congauss	subroutine for convolution with a Gaussian line shape
	conzeke	subroutine for convolution with the ZEKE line shape

A6. Source code for the Morse DVR program "idvr.f"

In this section we list the complete source code of the "idvr" program and relevant subroutines and functions. The "rsb" and "rs" subroutines are the standard EISPACK functions for matrix diagonalization;[11] hence the only the headers are listed for these.

A6.1. File "makeidvr"

The makefile used for recompiling the Morse DVR program is shown below. To recompile the program one enters "`make -f makeidvr`".

```
idvr: idvr101.o rsb.o rs.o matrix.o poten2.o convol.o
      f77 -O2 -o idvr idvr101.o rsb.o rs.o matrix.o poten2.o convol.o -
C

idvr101.o: idvr101.f
      f77 idvr101.f -c O2 -C

rsb.o: rsb.f
      f77 rsb.f -c -O2 -C

rs.o: rs.f
      f77 rs.f -c -O2 -C

matrix.o: matrix.f
```

```
	f77 matrix.f -c -O2 -C

poten2.o: poten2.f
	f77 poten2.f -c -O2 -C

convol.o: convol.f
	f77 convol.f -c -O2 -C
```

A6.2. File "idvr101.f"

```
      program idvr

      implicit undefined(a-z)
      integer ndim,nd1,nd5
      parameter(ndim=150,nd1=1000,nd5=5000)
      double precision norm(ndim+1),u(ndim+1,ndim+1),eigval(ndim+1),
     &  eigvec(ndim+1,ndim+1),fv1(ndim+1),fv1(ndim+1),r(ndim+1),
     &  diagv(ndim+1),eigmdvr(ndim+1,ndim+1),ham(ndim+1,ndim+1),
     &  peigval(ndim+1),peigvec(ndim+1,ndim+1),adiagv(ndim+1),
     &  apeigval(ndim+1),bvan(ndim+1),bvneu(ndim+1),
     &  apeigvec(ndim+1,ndim+1),fcf(ndim+1,ndim+1),totfcf(ndim+1),
     &  ymxy1(nd1),ymxy2(nd1),ymxy3(nd1),
     &  ymxy4(nd1),xgauss(nd5),ygauss(nd5),xzeke(nd5),yzeke(nd5),
     &  xexp(nd5),yexp(nd5),xexp2(nd5),yexp2(nd5),ysim(nd5)
      integer ianxy(nd1),ineutxy(nd1),npot(2),iarr(3),
     &  nmax,nfcf,nconv,i,j,ierr,nshow,nanshow,it,iab1,noconv,
     &  nexp,cint,ibad,cbad,notfirst
      character*30 fn,feig,fvsticks,fbv,fgauss,fzeke,fexp,fin,fout
      character*1 cit
      double precision rmu,anbasis(3),neubasis(3),
     &  anpot(10),neupot(10),zero,chisq,amass1,amass2,bline,
     &  gfwhm,zfwhm,evtocm,e0,vibtemp,re,beta,t,
     &  fv2,depthneu,poten,temp,temp2,depthan,depthm,anioncm,factor,
     &  ab1

      parameter(evtocm=8065.541)

      notfirst = 0

      if (notfirst.eq.0) then
      write(*,805)
11    write(*,*) 'Enter input filename :'
      read(*,*) fin
      write(*,*) 'Enter output filename :'
      read(*,*) fout
      endif
      if (fout.eq.fin) then
        write(*,*)'You do not really want to overwrite the input file,'
        write(*,*) 'Do you???'
        goto 11
      endif

1     open(8,file=fout)
      open(7,file=fin)
```

```
      write(8,805)
805   format(/,'Interactive Morse DVR program - ',
     &       'Version 1.01, Released March 1998',/,
     &       'Copyright 1998, Anion ZEKE Group, Neumark Group,',/,
     &       'Department of Chemistry, UC Berkeley',/)
      call idate(iarr)
      write(8,300) iarr(2),iarr(1),iarr(3)
      call itime(iarr)
      write(8,301) iarr(1),iarr(2),iarr(3)
300   format('Date: ',i2,'/',i2,1x,i4,$)
301   format(t30,'Time: ',i2,':',i2,':',i2,/)

      read(7,*) fn
      write(8,*) 'Parameter file : ',fn

      if (notfirst.eq.0) then
        call initpars(fn,rmu,amass1,amass2,anbasis,npot,anpot,neupot)
        do i=1,3
          neubasis(i)=anbasis(i)
        enddo
      endif

      write(8,105) amass1,amass2
105   format('Atomic masses : ',2f18.10)
      write(8,106) npot(1),npot(2)
106   format('Potential type      Anion : ',i2,'     Neutral : ',
     &  i2,/)
      write(8,*) '          Anion,  Neutral parameters'
      do i=1,10
        write(8,*) i,' : ',anpot(i),neupot(i)
      enddo

      read(7,*) fexp
      write(8,*) 'Experimental spectrum file (input) : ',fexp

      open(10,file=fexp)       !* Read in experimental data
      do i=1,99999
        read(10,*,end=303) xexp(i),yexp(i)
      enddo
303   nexp=i-1
      write (8,*) nexp,' points read from ',fexp
      write (8,309) xexp(1),xexp(nexp)
309   format('Low, High experimental cm-1 : ',2f10.2)
      if (xexp(nexp).lt.xexp(1)) then
        write(8,*) 'Note that experimental file is backwards'
        do j=1,nexp
          xexp2(j) = xexp(nexp-j+1)
          yexp2(j) = yexp(nexp-j+1)
        enddo
        do j=1,nexp
          xexp(j) = xexp2(j)
          yexp(j) = yexp2(j)
        enddo
      endif

      if (notfirst.eq.0) then
```

```
        read(7,*) feig
        read(7,*) fvsticks
        read(7,*) fbv
        read(7,*) fgauss
        read(7,*) fzeke
        read(7,*) gfwhm
        read(7,*) zfwhm
        read(7,*) nmax
        nmax=nmax-1
        read(7,*) nshow
        nshow=nshow-1
        read (7,*) nanshow
        nanshow=nanshow-1
        read(7,*) e0
        read(7,*) vibtemp
        read(7,*) bline
      endif


      write(8,*) 'File to save DVR points and eigenvectors : ',feig
      write(8,*) 'File to save vib. sticks                 : ',fvsticks
      write(8,*) 'File to save rotational constants        : ',fbv
      write(8,*) 'File to save Gaussian convolution        : ',fgauss
      write(8,*) 'File to save ZEKE lineshape convolution  : ',fzeke
      write(8,23) gfwhm
23    format(/,'FWHM for Gaussian convolution : ',f5.2,' cm-1')
      write(8,24) zfwhm
24    format('FWHM for ZEKE convolution : ',f5.2,' cm-1')
      write(8,26) nmax+1
26    format('Number of basis functions : ',i3)
      write(8,*) 'Show ',nshow+1,' neutral eigenvalues'
      write(8,*) 'Show ',nanshow+1,' anion eigenvalues'
      write(8,50) e0
50    format('Origin : ',f9.2,' cm-1')
      write(8,55) vibtemp
55    format('Vibrational Temperature : ',f6.2,' K')
      write(8,57) bline
57    format('Baseline for ZEKE convolution : ',f7.4,
     &  ' frac. of max peak')


      call setbasis(nmax,nshow,nanshow,rmu,npot,
     &              anpot,neupot,neubasis)
      re=neubasis(1)
      beta=neubasis(2)
      t=neubasis(3)
      depthm=(2.09008e-3)*((beta*t)**2)/rmu

      write(8,*)
      write(8,*) 'Basis paramaters:'
      write(8,61) re
61    format('    Re = ',f7.4,' A')
      write(8,66) beta
66    format('  beta = ',f7.4,' A-1')
      write(8,71) t
71    format('     t = ',f7.2)
      write(8,76) depthm
```

```
76    format(' Depth = ',f10.6,' eV')
      write(8,*)


      call umat(nmax,t,ndim,norm,u)

      call rsb(ndim+1,nmax+1,2,u,eigval,1,eigvec,fv1,fv2,ierr)
      if (ierr.ne.0) then
        write(*,*)'*** ierr=',ierr,' from rsb'
      endif

      call rcalc(nmax,ndim,re,t,beta,eigval,r)

      call diffpot(ndim,nmax,rmu,neubasis,2,npot(2),neupot,
     &             r,diagv,depthneu)

      open(2,file=feig)
      write(2,*)
      write(2,*)'DVR points (Angsroms, cm-1):'
      write(2,*)
      write(2,345)
345   format(2x,'r(i)',5x,'anion',4x,'neutral')
      do 400 i=0,nmax
        write(2,350) r(i+1),(poten(1,npot(1),anpot,r(i+1))+
     &    anpot(1))*evtocm,(poten(2,npot(2),neupot,r(i+1))+
     &    depthneu)*evtocm
400   continue
350   format(f8.4,1x,f8.2,1x,f8.2)

      call morseig(ndim,nmax,rmu,neubasis,eigvec,eigmdvr)
      call hcalc(ndim,nmax,eigmdvr,diagv,ham)
      call rs(ndim+1,nmax+1,ham,peigval,1,peigvec,fv1,fv2,ierr)
      call bvcalc(ndim,nmax,nshow,rmu,r,peigvec,bvneu)

      write(8,*) ' Neutral vibrational eigenvalues and rotational'
      write(8,*) ' constants (in cm-1) : '
      write(8,*) ' '
      write(8,245)
245   format(15x, 'Total', 6x, 'Spacing', 6x,'B(v)')
      write(8,*) ' '
      temp2=0.
      do i=0,nshow
        zero=(peigval(1))*8065.541
        temp=(peigval(i+1))*8065.541
        write(8,250) i,temp-zero,temp-temp2,bvneu(i+1)
        temp2=temp
      enddo

250   format(1x,'v = ',i2,4x,f9.2,4x,f8.2,4x,f9.5)

      write(2,*)
      write(2,*) 'Neutral eigenfunctions:'
      call showarr4(2,ndim+1,peigvec,nmax+1,nshow+1)
*
*  Anion calculation--using same basis
*
      call diffpot(ndim,nmax,rmu,neubasis,1,npot(1),anpot,
```

```
     &              r,adiagv,depthan)
      call hcalc(ndim,nmax,eigmdvr,adiagv,ham)
      call rs(ndim+1,nmax+1,ham,apeigval,1,apeigvec,fv1,fv2,
     &        ierr)
      call bvcalc(ndim,nmax,nanshow,rmu,r,apeigvec,bvan)

      write(8,*)
      write(8,*) ' Anion vibrational eigenvalues and rotational'
      write(8,*) ' constants (in cm-1) : '
      write(8,*) ' '
      write(8,245)
      write(8,*) ' '
      depthm=((beta*t/21.87343)**2)/rmu
      temp2=0.
      do i=0,nanshow
        zero=(apeigval(1))*8065.541
        temp=(apeigval(i+1))*8065.541
        write(8,250) i,temp-zero,temp-temp2,bvan(i+1)
        temp2=temp
      enddo
      write(2,*)
      write(2,*) 'Anion eigenfunctions:'
      call showarr4(2,ndim+1,apeigvec,nmax+1,nanshow+1)
      close(2)

      call calcfcf(ndim,nmax,nshow,nanshow,apeigvec,peigvec,fcf,
     &              totfcf)

      write(8,*)
      write(8,*)'Franck-Condon Factors :'
      write(8,145)
145   format(19x,'Anion v = ')
      write(8,150) (i, i = 0,nanshow)
150   format (17x,20(i3,4x),/)
      write(8,*)
      do 900 j=0,nshow
        write(8,155) j,(fcf(i+1,j+1), i = 0,nanshow)
900   continue
      write(8,*)
155   format ('Neutral v = ',i3,2x,20(f6.4,1x),/)
      write(8,160) (totfcf(i), i=1,nanshow+1)
160   format ('    Total = ',5x,20(f6.4,1x),/)
      write(8,*)
*
*  Make vibrational sticks - NOT normalized
*
       it=0
       do i = 1, nanshow+1
         do j = 1, nshow+1
          it=it+1
          ymxy1(it) = e0 + 8065.541*((peigval(j)-peigval(1)) -
     &          (apeigval(i)-apeigval(1)))
          ymxy2(it) = fcf(i,j)
          ymxy3(it) = bvan(i)
          ymxy4(it) = bvneu(j)
          ianxy(it) = i-1
          ineutxy(it) = j-1
```

```
        enddo
      enddo

      nfcf = (nshow+1)*(nanshow+1)

      it=0
      do i = 1, nanshow+1
        anioncm = (apeigval(i)-apeigval(1))*evtocm
        factor = exp(-anioncm/vibtemp/.695)
        do j = 1, nshow+1
          it=it+1
          ymxy2(it) = ymxy2(it)*factor
        enddo
      enddo
*
*  Sort vib sticks
*
      do i = 1, nfcf
        do j = i, nfcf
          if(ymxy1(j).lt.ymxy1(i)) then

            ab1 = ymxy1(j)
            ymxy1(j) = ymxy1(i)
            ymxy1(i) = ab1

            ab1 = ymxy2(j)
            ymxy2(j) = ymxy2(i)
            ymxy2(i) = ab1

            ab1 = ymxy3(j)
            ymxy3(j) = ymxy3(i)
            ymxy3(i) = ab1

            ab1 = ymxy4(j)
            ymxy4(j) = ymxy4(i)
            ymxy4(i) = ab1

            iab1 = ianxy(j)
            ianxy(j) = ianxy(i)
            ianxy(i) = iab1

            iab1 = ineutxy(j)
            ineutxy(j) = ineutxy(i)
            ineutxy(i) = iab1

          endif
        enddo
      enddo
*
*  Save vib. sticks and rotational constants
*
      write(8,*)
      write(8,*) 'Vibrational stick spectrum :'
      write(8,3030)
3030  format(/,' Pos./cm-1',5x,'% Inten.',2x,'v neut. - v an.',/)
      open(4,file=fvsticks)
      open(10,file=fbv)
```

```
       do i = 1, nfcf
         if (ymxy2(i).ge..003) then
           write(4,930) ymxy1(i), ymxy2(i)
           write(10,935) ymxy3(i),ymxy4(i)

           write(8,9030) ymxy1(i),ymxy2(i)*1.0d2,ineutxy(i),
     &                   ianxy(i)
         endif
420     enddo
        close(4)
        close(10)
930     format(f10.2,f10.6)
935     format(2f12.8)
9030    format(f9.1,3x,f8.2,8x,i2,' -',i2)
*
*Convolute vib sticks with Gaussian and save only those points that lie
*  within the range of the experimental spectrum.
*
       open(10,file=fgauss)
       call congauss(gfwhm,nfcf,1000,ymxy1,ymxy2,nconv,5000,
     &  xgauss,ygauss)
       noconv = 0
       do i = 1,nconv
         if ((xgauss(i).ge.xexp(1)).and.(xgauss(i).le.xexp(nexp))) then
           write(10,9050) xgauss(i),ygauss(i)
         else
          noconv = noconv+1
         endif
       enddo
       close(10)
9050   format (f12.4,f10.6)

       write(8,892) nconv-noconv,fgauss
       write(*,892) nconv-noconv,fgauss
892    format(/,'Gaussian convolution ',i5,' points saved to : ',a30)
*
*  Convolute vib sticks with ZEKE lineshape and save convoluted points
*  that lie within the exp spectrum. Also calculate chi-square for ZEKE
*  convolution vs. experimental data (all taken care of in subroutine
*  conzeke()
*
       open(10,file=fzeke)
       call conzeke(zfwhm,nfcf,1000,ymxy1,ymxy2,nconv,5000,
     &  xzeke,yzeke,nexp,xexp,yexp,ysim,bline,chisq)
       noconv = 0
       do i = 1,nconv
         if ((xzeke(i).ge.xexp(1)).and.(xzeke(i).le.xexp(nexp))) then
           write(10,9050) xzeke(i),yzeke(i)
         else
          noconv = noconv+1
         endif
       enddo
       close(10)
       write(8,891) nconv-noconv,fzeke
       write(*,891) nconv-noconv,fzeke
891    format(/,'ZEKE convolution ',i5,' points saved to : ',a30)
       write(8,898) chisq
```

```
      write(*,*)
898   format('Chi-square from ZEKE vibrational convolution : ',g12.6)
      close(7)
      close(8)
*
*  Begin the next iteration in the fitting process
*
      notfirst = 1
      write(*,*)
      write(*,*) '     Anion,  Neutral parameters'
      do i=1,10
        if ((anpot(i).ne.0.).and.(neupot(i).ne.0.)) then
          write(*,*) i,' : ',anpot(i),neupot(i)
        endif
      enddo
      write(*,*) 'Vibrational temperature : ',vibtemp
      write(*,*) 'ZEKE FWHM (cm-1)        : ',zfwhm
      write(*,*) 'Baseline for ZEKE conv. : ',bline
      write(*,898) chisq
      write(*,1693)
1693  format(79('-'))
1695  cbad = 0
      write(*,*) 'Enter "a/A" to change/optimize an anion parameter'
      write(*,*) '      "n/N" to change/optimize a neutral parameter'
      write(*,*) '      "v/V" to change/optimize the vib. temperature'
      write(*,*) '      "f/F" to change/optimize the ZEKE FWHM'
      write(*,*) '      "b/B" to change/optimize the baseline'
      write(*,*) '      "q" to quit idvr'

      read(*,*) cit
      if (cit.eq.'a') then
1795    write(*,*) 'Change which anion parameter (integer 1-10)?'
        read(*,*) cint
        ibad = 0
        if ((cint.le.10).and.(cint.ge.1)) then
          write(*,*)'Enter new value for anion parameter ',cint
          read(*,*) anpot(cint)
        else
          ibad = 1
        endif
        if (ibad.eq.1) goto 1795
      elseif (cit.eq.'n') then
1796    write(*,*) 'Change which neutral parameter (integer 1-10)?'
        read(*,*) cint
        ibad = 0
        if ((cint.le.10).and.(cint.ge.1)) then
          write(*,*)'Enter new value for neutral parameter ',cint
          read(*,*) neupot(cint)
        else
          ibad = 1
        endif
        if (ibad.eq.1) goto 1796
      elseif (cit.eq.'v') then
        write(*,*) 'Enter new value for vibrational temperature'
        read(*,*) vibtemp
      elseif (cit.eq.'f') then
        write(*,*) 'Enter new value for ZEKE FWHM'
```

```
        read(*,*) zfwhm
      elseif (cit.eq.'b') then
        write(*,9738)
9738    format('Enter new baseline as a fraction of total intensity',
     &         /,'(i.e. a real number from 0 to 1)')
        read(*,*) bline
      elseif ((cit.eq.'q').or.(cit.eq.'Q')) then
        goto 9999
      elseif ((cit.eq.'A').or.(cit.eq.'N').or.(cit.eq.'V').or.
     &        (cit.eq.'F').or.(cit.eq.'B')) then
        write(*,*) '"Optimize" feature not yet implemented'
        cbad=1
      else
        write(*,*) cit,' is not an option'
        cbad=1
      endif
      if (cbad.eq.1) goto 1695

      goto 1

9999  end

*=============================================================
      subroutine umat(nmax,t,ndim,norm,u)
*=============================================================
*
*  Set up diagonal and diag-1 of u(z) matrix elements, in
*  form suitable for input to rsb diagonalization routine
*
*  Input:  nmax, largest n value to calc.
*        t,  for Morse
*        ndim      dimension of norm vector
*        norm  vector containing ln(norm). constants
*
*  Output: u    contains diag-1 in first column,
*           diag in 2nd, first row index is 1.
*
      implicit double precision(a-h,o-z)
      double precision t,norm(ndim+1),u(ndim+1,ndim+1)

*  Diag-1 elements

      u(1,1)=0.
      do 100 i=1,nmax
         u(i+1,1)=sqrt(i*(2*t-i)/((2*t-2*i+1)*(2*t-2*i)**2
     &                            *(2*t-2*i-1)))
100   continue

*  Diagonal elements

      do 200 i=0,nmax
        u(i+1,2)=t/(2*(t-i)*(t-i-1))
200   continue

      return
      end
```

```
*=============================================================
      subroutine rcalc(nmax,ndim,re,t,beta,udiag,r)
*=============================================================
*
*  Calculate points R(i) from diagonalized u(R) matrix
*  elements.
*
*  Input:  udiag,  vector with eigenvalues of u(R)
*        nmax,   maximum n value (=# of evals -1)
*          ndim      dimension of udiag vector
*        re,         bond length of Morse
*        t,    for Morse
*        beta,   for Morse
*
*  Output: r,        contains R(i)'s (in Angstroms if beta
*             is in angst-1)
*
      implicit double precision(a-h,o-z)
      double precision udiag(ndim),r(ndim),re,t
      integer nmax,ndim

      do 100 i=0,nmax
        r(i+1)=re+log(2*t*udiag(i+1))/beta
100   continue
      return
      end

c ***********************************************************************
      subroutine initpars(fn,rmu,amass1,amass2,anbasis,npot,
     &  anpot,neupot)
c ***********************************************************************
c
c     Read masses, potential parameters, etc from
c     the file fname.
*
*   Input:  fn          file name with parms
*
*   Output: rmu               Reduced mass (amu)
*           anbasis(re,beta,t)      Morse basis parms (used if basis
*                             parameters can't be calculated
*                             from the potential)
*           amass1                  Atomic masses (amu) of each atom
*           amass2                  "     "     "     "
*           npot(1),npot(2)         Type of anion,neutral potential
*           anpot(...)        Anion potential parms
*           neupot(...)       Neutral potential parms`
*
      implicit double precision(a-h,o-z)
      character*30 fn
      double precision rmu,anbasis(3),anpot(10),neupot(10)
      integer npot(2)

      open(1,file = fn)
      read(1,*) amass1,amass2
      rmu=amass1*amass2/(amass1+amass2)
      read(1,*) anbasis(1),anbasis(2),anbasis(2)
      read(1,*) npot(1), npot(2)
```

```
      if ((npot(1).eq.1).or.(npot(1).eq.4)) then
        read(1,*) anpot(1),anpot(2),anpot(3),anpot(4)
      else if ((npot(1).eq.2).or.(npot(1).eq.6)) then
        read(1,*) anpot(1),anpot(2),anpot(3),anpot(4),anpot(5),
     &            anpot(6),anpot(7),anpot(8)
      elseif (npot(1).eq.3) then
        read(1,*) anpot(1),anpot(2),anpot(3)
      else
        pause 'Error in anion npot in paramater file'
      endif
c
      if ((npot(2).eq.1).or.(npot(2).eq.4)) then
        read(1,*) neupot(1),neupot(2),neupot(3),neupot(4)
      else if ((npot(2).eq.2).or.(npot(2).eq.6)) then
        read(1,*) neupot(1),neupot(2),neupot(3),neupot(4),
     &            neupot(5),neupot(6),neupot(7),neupot(8)
      elseif (npot(2).eq.3) then
        read(1,*) neupot(1),neupot(2),neupot(3)
      elseif (npot(2).eq.5) then
        read(1,*) neupot(1),neupot(2),neupot(3),neupot(4),
     &            neupot(5),neupot(6)
      elseif ((npot(2).eq.7).or.(npot(2).eq.8)) then
        read(1,*) neupot(1),neupot(2),neupot(3),neupot(4),
     &            neupot(5),neupot(6),neupot(7),neupot(8),
     &            neupot(9)
      elseif ((npot(2).eq.9).or.(npot(2).eq.10)) then
        read(1,*) (neupot(i), i=1,10)
      else
        pause 'Error in neutral npot in paramater file'
      endif

      close(1)

      return
      end

*=====================================================================
      subroutine diffpot(ndim,nmax,rmass,basis,numb,npot,parms,
     &                   r,diagv,depthp)
*=====================================================================
*
*  Calculate difference potential matrix  (delt-V = V - Vref)
*  in dvr by quadrature at points R(i), and transform to Morse fbr.
*  Zero energy is bottom of potential wells.
*
*  Inputs:  ndim
*           nmax
*               rmass               Reduced mass (amu)
*           basis(re,beta,t)  Morse parms
*               numb                1 for anion, 2 for neutral
*               npot                type of potential
*           parms()                 potential paramaters
*           r()               quadrature points
*
*  Output:  diagv()           diff. potl. elements in dvr basis
*           deltav()    difference potl matrix in Morse basis
```

```
*           depthp            well depth of poten (needed for Aq poten)
*
      implicit double precision(a-h,o-z)
      parameter(evtocm=8065.541)

      integer ndim,nmax,numb,npot
      double precision basis(3),parms(10),
     &    r(ndim+1),diagv(ndim+1)
      double precision poten,morse

      re=basis(1)
      beta=basis(2)
      t=basis(3)
      depthm=(2.09008e-3)*((beta*t)**2)/rmass
      if ((npot.eq.9).or.(npot.eq.10)) then !* Need to find min. forAq.
        depthp = 1.d99
        step = 0.001
        r1 = 3.0
        r2 = 5.0
        nstep = (r2-r1)/step
        rt = r1
        do i=1,nstep
          vtest = poten(numb,npot,parms,rt)
          if (vtest.lt.depthp) then
            depthp = vtest
            rvmin = rt
          endif
          rt = rt+step
        enddo
        depthp = -depthp
        write(*,5000) depthp
        write(*,5002) depthp*evtocm
        write(*,5001) rvmin
5000    format('Neutral potential depth = ',f8.5,' eV')
5002    format('                        = ',f8.2,' cm-1')
5001    format('At Rmin = ',f7.3,' Angstroms')
        write(*,*)
      else
        depthp=parms(1)
      endif
*
*  Fill diagonal delta-v with dvr matrix elements
*
      do 100 i=1,nmax+1
        diagv(i)=poten(numb,npot,parms,r(i))+depthp
     &          -morse(depthm,re,beta,r(i))-depthm
100   continue

      return
      end

*=======================================================================
      subroutine morseig(ndim,nmax,rmass,basis,tmat,eigmdvr)
*=======================================================================
*
*  Calculates eigenvalues of the morse potential
*
```

```
*  Inputs:  ndim
*           nmax
*           rmass
*           basis(re,beta,t)
*           tmat()            Transformation matrix
*
*  Output:  eigmdver()  Matrix with Morse eigenvalues
*                            in eV referred to asymptote,
*                       transformed to dvr basis.
*
      implicit double precision(a-h,o-z)
      parameter(c1=0.0457176,c2=2.09010e-3)

      integer ndim,nmax
      double precision basis(3),eigmdvr(ndim+1,ndim+1),
     &                 tmat(ndim+1,ndim+1)
      double precision eigm(300)

      re=basis(1)
      beta=basis(2)
      t=basis(3)
      depthm=(2.09008e-3)*((beta*t)**2)/rmass

      do 100 n=0,nmax
        eigm(n+1)=(-5.22519e-4)*beta*beta*((t*2-1-2*n)**2)
     &                  /rmass+depthm
100   continue

*
*  Transform E0(dvr) = T(transpose)*E0(fbr)*T
*
      do 400 n=1,nmax+1
        do 300 m=1,n
          temp=0.
          do 200 j=1,nmax+1
            temp=temp+tmat(j,n)*tmat(j,m)*eigm(j)
200       continue
          eigmdvr(n,m)=temp
300     continue
400   continue

      return
      end


*=====================================================================
      subroutine hcalc(ndim,nmax,eigmdvr,diagv,ham)
*=====================================================================
*
*  Calculate total hamiltonian in the dvr basis by adding Morse
*  eigenvalue (K) matrix to the potential difference matrix
*
*  Input:   ndim
*           nmax
*           eigmdvr()   Morse eigenvalues transformed to dvr
*           diagv()           potl difference matrix in dvr
*
```

```
*  Output:  ham()       Total ham matrix(lower half)
*
      implicit double precision(a-h,o-z)
      integer ndim,nmax
      double precision eigmdvr(ndim+1,ndim+1),diagv(ndim+1),
     &                 ham(ndim+1,ndim+1)

      do 100 i=1,nmax+1
        do 200 j=1,i-1
          ham(i,j)=eigmdvr(i,j)
200     continue
100   continue
      do 300 i=1,nmax+1
        ham(i,i)=eigmdvr(i,i)+diagv(i)
300   continue
      return
      end

*=====================================================================
      subroutine setbasis(nmax,nnshow,nanshow,rmass,
     &              npot,anparms,neuparms,basis)
*=====================================================================
*
*  Calculate Morse basis paramaters for a given number of basis
*  functions and potential paramaters.  t is set to the same proportion
* of the number of basis functions as the proportion of the approximate
*  total number of vibrational levels of the neutral potential to the
*  number of neutral levels to be calculated,
*  and then beta is chosen to match the
*  neutral potential second derivatives (force constants=2*De*beta^2)
*  at Re, which also determines De. Re's are set equal to min Re of
*  anion and neut.
*  If it doesn't know  force constants, basis read in is returned
*
*  Input:   nmax        +1 = number of basis functions
*           nnshow             number of neutral eigvalues to show
*           nanshow               "      anion    "
*           rmass
*           npot()             potential types (1=an,2=neu)
*           anparms()   anion potential parameters
*           neuparms()  neutral potential parameters
*           basis()            read from file
*
*  Output:  basis(re,beta,t)
*
      implicit double precision(a-h,o-z)
      double precision anparms(10),neuparms(10),basis(3)
      double precision k
      integer npot(2),nmax,nnshow,nanshow

      parameter( evtoj=1.60218e-19, amutokg=1.66054e-27,
     &   hbar=1.054573e-34, angsttom=1.e-10 )

      rmu=rmass*amutokg
      depthn=neuparms(1)*evtoj
      deptha=anparms(1)*evtoj
```

```
      if (npot(1).eq.2) then
        beta1a=anparms(3)/(anparms(2)*angsttom)
        ank=2.*deptha*(beta1a)**2
      else
        write(8,*) 'I dont know the anion second derivative'
        write(8,*) 'Using neutral basis from parameter file:'
        return
      endif

      if (npot(2).eq.2) then
        betan=(neuparms(3)+neuparms(4))/(2*neuparms(2)*angsttom)
        aneuk=2.*depthn*(betan)**2
        tneu=sqrt(2*rmu*depthn)/(hbar*betan)
      elseif ((npot(2).eq.7).or.(npot(2).eq.8)) then
        betan=8./(neuparms(2)*angsttom)
        aneuk=2.*depthn*(betan)**2
        tneu=sqrt(2*rmu*depthn)/(hbar*betan)
      else
        write(8,*) 'I dont know the neutral second derivative'
        write(8,*) 'Using neutral basis from parameter file:'
        return
      endif

      write(8,*) 't-neutral=',tneu
      if (nnshow.gt.tneu) then
        t=dble(nmax)+1.001
      else
        t=((tneu+1.5)*(nmax+1)/(nnshow+1))
      endif
      k=(aneuk)

      beta=(k*rmu)**0.25/sqrt(hbar*t)

      basis(1)=min(neuparms(2),anparms(2))
      basis(2)=beta*angsttom
      basis(3)=t

      return
      end

*=====================================================================
      subroutine calcfcf(ndim,nmax,nnshow,nanshow,anwf,neuwf,fcf,
     &                   totfcf)
*=====================================================================
*
*  Calculate Franck-Condon factors
*
*     Input:      ndim
*           nmax         max n in basis
*           nnshow             # neutral wf's shown
*           nanshow            # anion wf's shown
*           anwf()             anion eigenfunctions in columns
*           neuwf()            neutral eigenfunctions (in same basis)
*
*     Output:     fcf(anion n, neutral n)
*           totfcf(anion n)   Total fcf from anion levels
*
```

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

      integer ndim,nmax,nnshow,nanshow
      double precision anwf(ndim+1,ndim+1),neuwf(ndim+1,ndim+1),
     &                 fcf(ndim+1,ndim+1),totfcf(ndim+1)

      do 100 i=1,nanshow+1
        ftot=0.
        do 200 j=1,nnshow+1
          ftemp=0.
          do 300 n=1,nmax+1
            ftemp=ftemp+anwf(n,i)*neuwf(n,j)
300       continue
          fcf(i,j)=ftemp*ftemp
          ftot=ftot+fcf(i,j)
200     continue
        totfcf(i)=ftot
100   continue

      return
      end

*=====================================================================
      subroutine bvcalc(ndim,nmax,nshow,rmass,r,wf,bv)
*=====================================================================
*
*  Calculate rotational constants
*        B(v)=(hbar/4*pi*rmass*c)*<v|(1/R^2)|v>
*
*     Input:      ndim
*           nmax  # of basis fns-1
*           nshow # of r consts to calculate-1
*           rmass
*           r()   DVR points, in angstroms
*           wf()  matrix with wavefunctions in dvr
*
*     Output:     bv()  vector with B(v)s in wavenumbers
*
*

      parameter( evtoj=1.60218e-19, amutokg=1.66054e-27,
     &  hbar=1.054573e-34, angsttom=1.e-10, c=2.99792458e8,
     &  evtocm=8065.541,pi=3.14159265359)

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

      integer ndim,nmax,nshow
      double precision wf(ndim+1,ndim+1),bv(ndim+1),
     &     r(ndim+1),rmass

      integer i,j
      double precision sum,rmu

      rmu=rmass*amutokg

      do 100 i=1,nshow+1
        sum=0.
        do 200 j=1,nmax+1
```

```
          sum=sum+(wf(j,i)/r(j))**2
200       continue
          sum=sum/angsttom**2
          bv(i)=((hbar/(4*pi*rmu*c))*sum)/100.
100     continue

        return
        end
```

A6.3. File "rsb.f"[11]

```
c   ************************************************************

        subroutine rsb(nm,n,mb,a,w,matz,z,fv1,fv2,ierr)
c
        integer n,mb,nm,ierr,matz
        double precision a(nm,mb),w(n),z(nm,n),fv1(n),fv2(n)
        logical tf
c
c       this subroutine calls the recommended sequence of
c       subroutines from the eigensystem subroutine package (eispack)
c       to find the eigenvalues and eigenvectors (if desired)
c       of a real symmetric band matrix.
c
c       on input
c
c          nm  must be set to the row dimension of the two-dimensional
c          array parameters as declared in the calling program
c          dimension statement.
c
c          n  is the order of the matrix  a.
c
c          mb  is the half band width of the matrix, defined as the
c          number of adjacent diagonals, including the principal
c          diagonal, required to specify the non-zero portion of the
c          lower triangle of the matrix.
c
c          a  contains the lower triangle of the real symmetric
c          band matrix.  its lowest subdiagonal is stored in the
c          last  n+1-mb  positions of the first column, its next
c          subdiagonal in the last  n+2-mb  positions of the
c          second column, further subdiagonals similarly, and
c          finally its principal diagonal in the  n  positions
c          of the last column.  contents of storages not part
c          of the matrix are arbitrary.
c
c          matz  is an integer variable set equal to zero if
c          only eigenvalues are desired.  otherwise it is set to
c          any non-zero integer for both eigenvalues and eigenvectors.
c
c       on output
c
c          w  contains the eigenvalues in ascending order.
c
c          z  contains the eigenvectors if matz is not zero.
```

```
c
c        ierr  is an integer output variable set equal to an error
c           completion code described in the documentation for tqlrat
c           and tql2.  the normal completion code is zero.
c
c        fv1  and  fv2  are temporary storage arrays.
c
c     questions and comments should be directed to burton s. garbow,
c     mathematics and computer science div, argonne national laboratory
c
c     this version dated august 1983.
c
```

A6.4. File "rs.f"[11]

```
      subroutine rs(nm,n,a,w,matz,z,fv1,fv2,ierr)
c
      integer n,nm,ierr,matz
      double precision a(nm,n),w(n),z(nm,n),fv1(n),fv2(n)
c
c     this subroutine calls the recommended sequence of
c     subroutines from the eigensystem subroutine package (eispack)
c     to find the eigenvalues and eigenvectors (if desired)
c     of a real symmetric matrix.
c
c     on input
c
c        nm  must be set to the row dimension of the two-dimensional
c        array parameters as declared in the calling program
c        dimension statement.
c
c        n  is the order of the matrix  a.
c
c        a  contains the real symmetric matrix.
c
c        matz  is an integer variable set equal to zero if
c        only eigenvalues are desired.  otherwise it is set to
c        any non-zero integer for both eigenvalues and eigenvectors.
c
c     on output
c
c        w  contains the eigenvalues in ascending order.
c
c        z  contains the eigenvectors if matz is not zero.
c
c        ierr  is an integer output variable set equal to an error
c           completion code described in the documentation for tqlrat
c           and tql2.  the normal completion code is zero.
c
c        fv1  and  fv2  are temporary storage arrays.
c
c     questions and comments should be directed to burton s. garbow,
c     mathematics and computer science div, argonne national laboratory
c
c     this version dated august 1983.
```

```
c
```

A6.5. File "matrix.f"

```
c -----------------------------------------------------------------
      subroutine showarr4(nfile,ndim,a,nra,nca)
c
c     displays the matrix A
c
      implicit double precision (a-h,o-z)
      dimension a(ndim,ndim)

      do 10 ir = 1, nra
         write(nfile,100) (a(ir,ic), ic = 1,nca)
100      format(20(f6.4,1x),/)
10    continue
      return
      end
```

A6.6. File "poten2.f"

```
c ********************************************************
       double precision function poten(numb,npot,p,rt)
c ********************************************************
*
*  Input:    numb  set to 1 for anion, 2 for neutral
*            npot  scalar specifies type of potential
*            p()   potential parameter list
*            rt    point in angstroms
c
c  Returns:        potential in eV
c
c     npot = 1 for Maitland-Smith type potential;
c     npot = 2 for scaled Lee (Morse-Morse-Switch-van der Waals)
c                      type  potential;
c     npot = 3 for Morse potential.
c     npot = 4 for Buckingham potential (exp-n for neutral,
c                     or anion.)
c     npot = 5 for Morse-Spline-van der Waals (for neutral only.)
c     npot = 6 for unscaled Morse-Morse-Switch-vdW
c     npot = 7 for HFD-C potential (neutral only)
*     npot = 8 for HFD-B potential (neutral only)
*     npot = 9 for Aquilanti X1/2 (neut only) MSV for V0,bucky for V2
*     npot = 10 for Aquilanti I3/2, MSV for V0, bucky for V2
*
c
c     (1.) when npot(numb) = 1, potentials are the Maitland-Smith form,
c          n(r*)-6 type for the neutral and n(r*)-4 type for the anion,
c          respectively. There will be 4 parameters:
*                 (1)   Well depth (eV)
*                 (2)   Re (Angs)
*                 (3)   x
*                 (4)   gamma
c
```

```
c     (2.) when npot(numb) = 2, potentials are the Lee form, i.e., the
c          Morse-Morse-Switch-van der Waals form, there will be eight
c          parameters for each potential:
*           (1)   .Well depth (eV)
*           (2)    Re (Angst.)
*           (3)    Beta1 (unitless)
*           (4)    Beta2 (unitless)
*           (5)    x1
*           (6)    x2
*           (7)    anion C4/well depth, or neutral C6/well depth
*           (8)    anion C6/well depth, or neutral C8/well depth
c                  (units Angs^(-n), n=4,6 or 8)
*
c     (3.) when npot(numb) = 3, potentials take on the Morse potential
c          form, there will be three parameters for each potential.
*           (1)    depth (eV)
*           (2)    Re (Angst)
*           (3)    Beta (Angst-1)
*
*     (4.) npot(numb) = 4:  Buckingham (exp-n) potential:
*           (1)    Well depth
*           (2)    Re
*           (3)    beta
*           (4)    xm
*
*     (5.) npot(2)=5:  Morse-Spline-van der Waals (Aquilanti)--Neutral
*           6 parameters
*           (1)    Depth (eV)
*           (2)    Rmin (Angst)
*           (3)    Beta (Angst-1) -- Unscaled, different from Aqui.
*           (4)    x1 (unitless)
*           (5)    x2 (unitless)
*           (6)    C0 (eV*Angst^6)
*
*     (6.) npot(numb) = 6:  Unscaled Morse-Morse-Switch-vdW
*           --Same parameters as scaled MMSV, but betas have
*             angst-1 units, and Cn have eV*Angs^(-n) units
*             (Obtained from scaled beta and Cn by dividing
*             by Re or multiplying by well depth, respectively)
*
*     (7.) npot(2)=7: HFD-C potential
*           (1)    Depth (eV)
*           (2)    Re (Angst)
*           (3)    Alpha
*           (4)    gamma
*           (5)    A
*           (6)    C6 (unitless)
*           (7)    C8     "
*           (8)    C10    "
*           (9)    D
*
*     (8.) npot(2)=8: HFD-B potential
*           (1)    Depth (eV)
*           (2)    Re (Angst)
*           (3)    Alpha
*           (4)    beta
*           (5)    A
```

```
*           (6)   C6 (unitless)
*           (7)   C8     "
*           (8)   C10    "
*           (9)   D
*
*     (9) npot(2)=9: Aquilanti X1/2 potential, V0=MSV, V2=Buckingham
*           (1)   Depth of V0 (eV) (Different from poten depth!)
*           (2)   Rmin of V0 (Angst)
*           (3)   Beta of V0 (Angst-1) -- Unscaled.
*           (4)   x1 of V0 (unitless)
*           (5)   x2 of V0 (unitless)
*           (6)   C0 of V0 (eV*Angst^6)
*           (7)   A2 of V2 (eV)
*           (8)   alpha2 of V2 (Angst^-1)
*           (9)   C2 of V2 (ev*Angst^6)
*           (10)  halogen spin-orbit constant (eV)
c
c
*     (10) npot(2)=10: Aquilanti I3/2 potential, same parameters as 9
*
      implicit double precision(a-h,o-z)
      integer numb,npot
      double precision p(10),rt
      double precision ms4,anmmsv,morse,bucky,anmmsvus,
     &                 ms6,neummsv,msv,neummsvus,hfd_c,hfd_b,
     &                 aqx,aqi

      if (numb.eq.1) then
        if (npot.eq.2) then
          poten=anmmsv(p(1),p(2),p(3),p(4),p(5),p(6),p(7),p(8),rt)
        elseif (npot.eq.1) then
          poten=ms4(p(1),p(2),p(3),p(4),rt)
        elseif (npot.eq.3) then
          poten=morse(p(1),p(2),p(3),rt)
        elseif (npot.eq.4) then
          poten=bucky(p(1),p(2),p(3),p(4),rt)
        elseif (npot.eq.6) then
          poten=anmmsvus(p(1),p(2),p(3),p(4),p(5),p(6),p(7),p(8),rt)
        else
          pause 'Error in anion potential call'
        endif
      elseif (numb.eq.2) then
        if (npot.eq.2) then
          poten=neummsv(p(1),p(2),p(3),p(4),p(5),p(6),p(7),p(8),rt)
        elseif (npot.eq.8) then
          poten=hfd_b(p(1),p(2),p(3),p(4),p(5),p(6),p(7),p(8),p(9),rt)
        elseif (npot.eq.7) then
          poten=hfd_c(p(1),p(2),p(3),p(4),p(5),p(6),p(7),p(8),p(9),rt)
        elseif (npot.eq.1) then
          poten=ms6(p(1),p(2),p(3),p(4),rt)
        elseif (npot.eq.3) then
          poten=morse(p(1),p(2),p(3),rt)
        elseif (npot.eq.4) then
          poten=bucky(p(1),p(2),p(3),p(4),rt)
        elseif (npot.eq.5) then
          poten=msv(p(1),p(2),p(3),p(4),p(5),p(6),rt)
        elseif (npot.eq.6) then
```

```
          poten=neummsvus(p(1),p(2),p(3),p(4),p(5),p(6),p(7),p(8),rt)
        elseif (npot.eq.9) then
          poten=aqx(p(1),p(2),p(3),p(4),p(5),p(6),p(7),p(8),p(9),
     &              p(10),rt)
        elseif (npot.eq.10) then
          poten=aqi(p(1),p(2),p(3),p(4),p(5),p(6),p(7),p(8),p(9),
     &              p(10),rt)
        else
          pause 'Error in neutral potential call'
        endif
       endif
       return
       end

c-----------------------------------------------------------------
       double precision function ms4(depth1,rmin1,xm1,xlamda1,rt)
c-----------------------------------------------------------------
*
c     Maitland-Smith n-4
c

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

        xt = rt/rmin1
        rtn = xm1+xlamda1*(xt-1.)
        ms4 = depth1/(rtn-4.)*(4.*(1./xt)**rtn -
     &         rtn*(1./xt)**4)
        return
        end

*-----------------------------------------------------------------
       double precision function anmmsv(depth1,rmin1,pmbeta11,
     &  pmbeta21,xrstar11,xrstar21,c4vdw1,c6vdw1,rt)
c-----------------------------------------------------------------
c     Scaled MMSV for anion
c
        implicit double precision(a-h,o-z)
        parameter(pi=3.14159265359)

        xt = rt/rmin1
         if (xt.le.1.) then
          Tsum = depth1*(exp(2.*pmbeta11*(1.-xt))-2.*exp(
     &            pmbeta11*(1.-xt)))
         else if ((xt.gt.1).and.(xt.le.xrstar11)) then
          Tsum = depth1*(exp(2.*pmbeta21*(1.-xt))-2.*exp(
     &            pmbeta21*(1.-xt)))
         else if ((xt.gt.xrstar11).and.(xt.lt.xrstar21)) then
          swxt=0.5*(cos(pi*(xt-xrstar11)/(xrstar21-xrstar11))+1.)
          pmorse2=exp(2.*pmbeta21*(1.-xt))-2.*exp(
     &            pmbeta21*(1.-xt))
          pvdw=-1.*(c4vdw1*rt**(-4)+c6vdw1*rt**(-6))
          Tsum = depth1*(swxt*pmorse2+(1.-swxt)*pvdw)
         else
          Tsum = -1.*depth1*(c4vdw1*rt**(-4)+c6vdw1*rt**(-6))
         end if
        anmmsv=Tsum
        return
```

```
      end

*-----------------------------------------------------------------
      double precision function morse(depth,rmin,beta,rt)
c-----------------------------------------------------------------
c    Morse potential
c
       implicit double precision(a-h,o-z)

       xt = rt/rmin
       morse = depth*(exp(2.*beta*rmin*(1.-xt))-2.*exp(
     &            beta*rmin*(1.-xt)))
       return
       end

*-----------------------------------------------------------------
      double precision function bucky(depth,rmin,beta,xm,rt)
c-----------------------------------------------------------------
*
c    Exp-n Potential
c
      implicit double precision(a-h,o-z)
      xt=rt/rmin

      bucky = (depth/(beta*rmin-xm))*
     &          (xm*exp(-beta*(rt-rmin))
     &           -beta*rmin*((rmin/rt)**xm))
      return
      end

*-----------------------------------------------------------------
      double precision function anmmsvus(depth1,rmin1,pmbeta11,
     &  pmbeta21,xrstar11,xrstar21,c4vdw1,c6vdw1,rt)
c-----------------------------------------------------------------
c    Unscaled MMSV
c
*
      implicit double precision(a-h,o-z)
      parameter(pi=3.14159265359)

      xt=rt/rmin1
        pmbeta11 = pmbeta11 * rmin1
        pmbeta21 = pmbeta21 * rmin1
        c4vdw1 = c4vdw1 / depth1
        c6vdw1 = c6vdw1 / depth1
        if (xt.le.1.) then
         Tsum = depth1*(exp(2.*pmbeta11*(1.-xt))-2.*exp(
     &            pmbeta11*(1.-xt)))
        else if ((xt.gt.1).and.(xt.le.xrstar11)) then
         Tsum = depth1*(exp(2.*pmbeta21*(1.-xt))-2.*exp(
     &            pmbeta21*(1.-xt)))
        else if ((xt.gt.xrstar11).and.(xt.lt.xrstar21)) then
         swxt=0.5*(cos(pi*(xt-xrstar11)/(xrstar21-xrstar11))+1.)
         pmorse2=exp(2.*pmbeta21*(1.-xt))-2.*exp(
     &            pmbeta21*(1.-xt))
         pvdw=-1.*(c4vdw1*rt**(-4)+c6vdw1*rt**(-6))
         Tsum = depth1*(swxt*pmorse2+(1.-swxt)*pvdw)
```

```
        else
         Tsum = -1.*depth1*(c4vdw1*rt**(-4)+c6vdw1*rt**(-6))
        end if
       anmmsvus=Tsum
       return
       end

c-----------------------------------------------------------------
      double precision function ms6(depth2,rmin2,xm2,xlamda2,rt)
c-----------------------------------------------------------------
*
c     Maitland-Smith n-6
c
      implicit double precision(a-h,o-z)
      xm2=rt/rmin2
      rtn = xm2+xlamda2*(xt-1.)
      ms6 = depth2/(rtn-6.)*(6.*(1./xt)**rtn -
     &         rtn*(1./xt)**6)
      return
      end

*-----------------------------------------------------------------
      double precision function neummsv(depth2,rmin2,pmbeta12,
     &  pmbeta22,xrstar12,xrstar22,c6vdw2,c8vdw2,rt)
c-----------------------------------------------------------------
*
c     Scaled MMSV
c
      implicit double precision(a-h,o-z)
      parameter(pi=3.14159265359)

      xt=rt/rmin2
        if (xt.le.1.) then
         Tsum = depth2*(exp(2.*pmbeta12*(1.-xt))-2.*exp(
     &            pmbeta12*(1.-xt)))
        elseif ((xt.gt.1).and.(xt.le.xrstar12)) then
         Tsum = depth2*(exp(2.*pmbeta22*(1.-xt))-2.*exp(
     &            pmbeta22*(1.-xt)))
        elseif ((xt.gt.xrstar12).and.(xt.lt.xrstar22)) then
         swxt=0.5*(cos(pi*(xt-xrstar12)/(xrstar22-xrstar12))+1.)
         pmorse2=exp(2.*pmbeta22*(1.-xt))-2.*exp(
     &            pmbeta22*(1.-xt))
         pvdw=-1.*(c6vdw2*rt**(-6)+c8vdw2*rt**(-8))
         Tsum = depth2*(swxt*pmorse2+(1.-swxt)*pvdw)
        else
         Tsum = -1.*depth2*(c6vdw2*rt**(-6)+c8vdw2*rt**(-8))
        endif
      neummsv=Tsum
      return
      end

*-----------------------------------------------------------------
      double precision function msv(depth2,rmin2,beta2,x1msv,
     &  x2msv,c0msv,rt)
c-----------------------------------------------------------------
*
c     MSV (Aquilanti)
```

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

         r1msv = x1msv*rmin2
         r2msv = x2msv*rmin2
         if (rt.le.r1msv) then
           Tsum = depth2*(exp(2.*beta2*(rmin2-rt))
     &             -2.*exp(beta2*(rmin2-rt)))
         else if (rt.lt.r2msv) then
           b1s = exp(2.*beta2*(rmin2-r1msv))
     &              -2.*exp(beta2*(rmin2-r1msv))
           b2s = -((c0msv/(depth2*r2msv**6))+b1s)/(r2msv-r1msv)
           b3s = (2.*beta2*(exp(beta2*(rmin2-r1msv))
     &                    -exp(2.*beta2*(rmin2-r1msv)))-b2s)
     &             /(r1msv-r2msv)
           b4s = ((6.*c0msv/(depth2*r2msv**7))-b2s
     &             -b3s*(r2msv-r1msv))/(r2msv-r1msv)**2
           Tsum = b1s+(rt-r1msv)*(b2s+(rt-r2msv)
     &                              *(b3s+(rt-r1msv)*b4s))
           Tsum = Tsum*depth2
         else
          Tsum = -c0msv/rt**6
         end if
         msv=Tsum
         return
         end

*-----------------------------------------------------------------
      double precision function aqx(depth,rmin,beta,x1,x2,
     &  c0,a2,alpha2,c2,soconst,rt)
*-----------------------------------------------------------------
*
*  X1/2 State Aquilanti, JPC v.97, p.2063
*
      implicit double precision(a-h,o-z)
      double precision msv

      v0=msv(depth,rmin,beta,x1,x2,c0,rt)
      v2 = -a2*exp(-alpha2*rt)  + c2/(rt**6)
      t1 = sqrt(9.*v2**2/25. + soconst**2 - 2.*v2*soconst/5.)
      aqx = v0 + v2/10. + soconst/2. - t1/2.
      return
      end

*-----------------------------------------------------------------
      double precision function aqi(depth,rmin,beta,x1,x2,
     &  c0,a2,alpha2,c2,soconst,rt)
*-----------------------------------------------------------------
*
*  I3/2 State Aquilanti
*
      implicit double precision(a-h,o-z)
      double precision msv

      v0=msv(depth,rmin,beta,x1,x2,c0,rt)
      v2 = -a2*exp(-alpha2*rt)  + c2/(rt**6)
      aqi = v0 - v2/5.
```

```
      return
      end

*-----------------------------------------------------------------
      double precision function neummsvus(depth2,rmin2,pmbeta12,
     &  pmbeta22,xrstar12,xrstar22,c6vdw2,c8vdw2,rt)
c-----------------------------------------------------------------
*
c     Unscaled MMSV
c
       implicit double precision(a-h,o-z)
       parameter(pi=3.14159265359)

       xt=rt/rmin2
        pmbeta12 = pmbeta12 * rmin2
        pmbeta22 = pmbeta22 * rmin2
        c6vdw2 = c6vdw2 / depth2
        c8vdw2 = c8vdw2 / depth2
        if (xt.le.1.) then
         Tsum = depth2*(exp(2.*pmbeta12*(1.-xt))-2.*exp(
     &            pmbeta12*(1.-xt)))
        else if ((xt.gt.1).and.(xt.le.xrstar12)) then
         Tsum = depth2*(exp(2.*pmbeta22*(1.-xt))-2.*exp(
     &            pmbeta22*(1.-xt)))
        else if ((xt.gt.xrstar12).and.(xt.lt.xrstar22)) then
         swxt=0.5*(cos(pi*(xt-xrstar12)/(xrstar22-xrstar12))+1.)
         pmorse2=exp(2.*pmbeta22*(1.-xt))-2.*exp(
     &            pmbeta22*(1.-xt))
         pvdw=-1.*(c6vdw2*rt**(-6)+c8vdw2*rt**(-8))
         Tsum = depth2*(swxt*pmorse2+(1.-swxt)*pvdw)
        else
         Tsum = -1.*depth2*(c6vdw2*rt**(-6)+c8vdw2*rt**(-8))
        end if
       neummsvus=Tsum
       return
       end

*-----------------------------------------------------------------
      double precision function hfd_c(depth2,rmin2,alpha2,gamma2,
     &  acoef2,c6vdw2,c8vdw2,c10vdw2,damp2,rt)
*-----------------------------------------------------------------
*
*  Hartree-Fock Dispersion-C
*
      implicit double precision(a-h,o-z)

      xt=rt/rmin2
*
*  Dispersion damping function
*
      if (xt.lt.damp2) then
        fdamp=exp(-(damp2/xt-1)**2)
      else
        fdamp=1.
      endif
*
*  Repulsive part
```

```
*
      repul=acoef2*(xt**gamma2)*exp(-alpha2*xt)
*
*  Dispersion terms
*
      disp=c6vdw2*xt**(-6)+c8vdw2*xt**(-8)+c10vdw2*xt**(-10)

      Tsum=repul-fdamp*disp
      hfd_c=depth2*Tsum
      return
      end
*
*-----------------------------------------------------------------
      double precision function hfd_b(depth2,rmin2,alpha2,beta2,
     &  acoef2,c6vdw2,c8vdw2,c10vdw2,damp2,rt)
*-----------------------------------------------------------------
*
*  Hartree-Fock Dispersion-B Potential
*  Ref: Aziz & Slaman Mol. Phys. v.58, p.679
*
      implicit double precision(a-h,o-z)

      xt=rt/rmin2
*
*  Dispersion damping function
*
      if (xt.lt.damp2) then
        fdamp=exp(-(damp2/xt-1)**2)
      else
        fdamp=1.
      endif
*
*  Repulsive part
*
      repul=acoef2*exp(-alpha2*xt+beta2*xt*xt)
*
*  Dispersion terms
*
      disp=c6vdw2*xt**(-6)+c8vdw2*xt**(-8)+c10vdw2*xt**(-10)

      Tsum=repul-fdamp*disp
      hfd_b=depth2*Tsum
      return
      end
```

A6.7. File "convol.f"

```
*=================================================================
      subroutine congauss(fwhm,nsticks,nvec,xstk,ystk,nconv,nvecgau,
     &  xgauss,ygauss)
*=================================================================
*
*  Convolutes stick spectrum with Gaussian
*  Sticks are assumed to already be sorted from low to high cm-1.
*
```

```
*     Input:      fwhm  FWHM of Gaussian in cm-1
*           nsticks     Number of sticks
*           nvec  Dimension of xstk & ystk vectors
*           xstk()      vector containing wavenumbers of sticks
*           ystk()  vector containing intersities of sticks
*           nvecgau     Dimension of xgauss & ygauss
*
*     Output:     nconv Number of points in convoluted spectrum
*           xgauss()    Wavenumbers of convoluted spectrum
*           ygauss()    Intensities of convoluted spectrum (normalized)
*
*
      implicit undefined(a-z)
*
      integer nsticks,nvec,nconv,nvecgau
      double precision fwhm
      double precision xstk(nvec),ystk(nvec),xgauss(nvecgau),
     &  ygauss(nvecgau)
*
      integer i,j
      double precision wmin,wmax,step,wn,cmax,con,hwhm
*
      wmin = xstk(1)
      wmax = xstk(nsticks)
*
      hwhm = fwhm/2.d0
      step = fwhm/10.d0
      wmin = wmin-fwhm*5.d0
      wmax = wmax+fwhm*5.d0
      nconv = nint((wmax-wmin)/step)+2
      cmax = 0.

      do i=1,nconv                    !* Loop over grid points
        wn = wmin+dble(i-1)*step
        xgauss(i) = wn
        ygauss(i) = 0.
        do j=1,nsticks                !* Loop over sticks
          if (((wn-xstk(j))/hwhm)**2.lt.100.) then
            con = ystk(j)*
     &            exp(-0.5d0*((wn-xstk(j))/(fwhm/2.354))**2.)
          else
            con = 0.d0
          endif
          ygauss(i)=ygauss(i)+con
        enddo
        if (ygauss(i).gt.cmax) then
          cmax = ygauss(i)
        endif
      enddo

      do i=1,nconv                    !* Normalize
        ygauss(i)=ygauss(i)/cmax
      enddo

      return
      end
```

```
*=====================================================================
      subroutine conzeke(fwhm,nsticks,nvec,xstk,ystk,nconv,nveczek,
     &  xzeke,yzeke,nexp,xexp,yexp,ysim,bline,chisq)
*=====================================================================
*
*  Convolutes stick spectrum with ZEKE lineshape from fit to bromine
*  atomic data, form
*     y=(a*x+b*x^3)/(1+c*x^2+d*x^4)
*  Sticks are assumed to already be sorted from low to high cm-1.
*  Also calculates the chi-square between the convolouted and
*  experimental spectra.
*
*     Input:       fwhm  FWHM cm-1
*           nsticks     Number of sticks
*           nvec  Dimension of xstk & ystk vectors
*           xstk()      vector containing wavenumbers of sticks
*           ystk()  vector containing intersities of sticks
*           nveczek     Dimension of xzeke & yzeke
*           nexp  number of experimental points
*           xexp  wavenumbers of experimental points
*           yexp  intensities of experimental points
*           bline Baseline for convolution (as fraction of
*                 highest peak normalized to 1)
*
*     Output:      nconv Number of points in convoluted spectrum
*           xzeke()     Wavenumbers of convoluted spectrum
*           yzeke()     Intensities of convoluted spectrum (normalized)
*           ysim()      Intensities of ZEKE convoluted simulation
*                 at points of experimental spectrum
*           chisq       chi-square, weighted by square root (ie so
*                       the peaks are weighted more than the valleys)
*
*
      implicit undefined(a-z)
*
      integer nsticks,nvec,nconv,nveczek,nexp
      double precision fwhm,bline
      double precision xstk(nvec),ystk(nvec),xzeke(nveczek),
     &  yzeke(nveczek),xexp(nexp),yexp(nexp),ysim(nexp)
*
      integer i,j
      double precision wmin,wmax,step,wn,cmax,con,hwhm,a,b,c,d,cc,x1,
     &  wn2,x2,chisq
      parameter(a=1.3342091765d0,
     &          b=0.0059500947d0,
     &          c=0.40742214d0,
     &          d=0.022329373d0)
*
      x1 = 0.43d0*fwhm
      cc = 3.2d0/fwhm
*
      wmin = xstk(1)
      wmax = xstk(nsticks)
*
      hwhm = fwhm/2.d0
      step = fwhm/10.d0
      wmin = wmin-fwhm*5.d0
```

```
      wmax = wmax+fwhm*5.d0
      nconv = nint((wmax-wmin)/step)+2
      cmax = 0.

      do i=1,nconv                    !* Loop over exp points
        wn = wmin+dble(i-1)*step
        xzeke(i) = wn
        wn2 = wn + x1
        con = 0.0d0
        yzeke(i) = 0.0d0
        do j=1,nsticks                !* Loop over sticks
          if (xstk(j).lt.wn2) then
            x2 = (wn2 - xstk(j))
            con = ystk(j)*(a*(x2*cc)+b*(x2*cc)**3)/
     &                  (1.0d0+c*(cc*x2)**2+d*(cc*x2)**4)
            yzeke(i) = yzeke(i) + con
          endif
        enddo
        if (yzeke(i).gt.cmax) then
          cmax = yzeke(i)
        endif
      enddo

      do i=1,nconv                    !* Normalize (w/baseline)
        yzeke(i)=(yzeke(i)/cmax)*(1.0d0-bline)+bline
      enddo
*
*  Do the convolution again at the points of the exp specturm
*

      cmax = 0.
*
      chisq = 0.
      do i=1,nexp              !* Loop over exp points
        wn = xexp(i)
        wn2 = wn + x1
        ysim(i) = 0.0d0
        do j=1,nsticks                !* Loop over sticks
          if (xstk(j).lt.wn2) then
            x2 = (wn2 - xstk(j))
            con = ystk(j)*(a*(x2*cc)+b*(x2*cc)**3)/
     &                    (1.0d0+c*(cc*x2)**2+d*(cc*x2)**4)
          ysim(i) = ysim(i) + con
          endif
        enddo
        if (ysim(i).gt.cmax) then
          cmax = ysim(i)
        endif
      enddo

      do i=1,nexp              !* Normalize
        ysim(i)=(ysim(i)/cmax)*(1.0d0-bline)+bline
      enddo

      chisq = 0.0d0
      do i=1,nexp
        chisq = chisq + dabs(yexp(i))*(yexp(i)-ysim(i))**2
```

```
enddo

return
end
```

A7. References for Appendix A

[1] D. O. Harris, G. G. Engerholm, and W. D. Gwinn, J. Chem. Phys. **43**, 1515 (1965).

[2] J. C. Light, I. P. Hamilton, and J. V. Lill, J. Chem. Phys. **82**, 1400 (1985).

[3] R. B. Metz, Ph.D. Thesis, University of California, 1991.

[4] E. M. Greenawalt and A. S. Dickinson, J. Mol. Spectrosc. **30**, 427 (1969).

[5] P. M. Morse, Phys. Rev. **34**, 57 (1929).

[6] A. S. Dickenson and P. R. Certain, J. Chem. Phys. **49**, 4209 (1968).

[7] Y Zhao, I. Yourshaw, G. Reiser, C. C. Arnold, and D. M. Neumark, J. Chem. Phys. **101**, 6538 (1994).

[8] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes*, 2nd ed. (Cambridge University Press, Cambridge, 1992).

[9] V. Aquilanti, D. Cappelletti, V. Lorent, E. Luzzatti, and F. Pirani, J. Phys. Chem. **97**, 2063 (1993).

[10] R. A. Aziz and M. J. Slaman, Mol. Phys. **58**, 679 (1986).

[11] The EISPACK source code may be obtained by email from netlib@research.att.com or by FTP from the server "research.att.com" using the login name "netlib."

Appendix B. Program for fitting rovibronic transitions in RgX^- photodetachment.

In this appendix we describe the computer program used to simulate the rotational profiles of the line shapes observed in the ZEKE spectra of the diatomic RgX^- complexes. First we review the derivation of the selection rules and rotational line strength factors for photoelectron transitions of Hund's case (c) molecules. We then describe in detail the use of the rotational fitting program.

B1. Line strength factors for photodetachment of Hund's case (c) molecules.*

As mentioned in Chapter 3, the peak shapes of the ZEKE spectra are asymmetric and change from band to band. In addition, the peak widths vary with source conditions. We explain the peak shape and width in terms of the asymmetric ZEKE experimental peak shape combined with the effect of the unresolved rotational structure associated with each vibrational transition. Both effects are discussed in this section, with the main focus on how one treats the rotational contribution to the peak widths.

The ejected electron has spin $s = 1/2$ and orbital angular momentum $l = 0$ (only s-wave photodetachment is observed in anion ZEKE experiments, as discussed in Chapter 1). Hence, the anion→neutral rotational selection rules are $\Delta J = \pm\frac{1}{2}, \pm\frac{3}{2}$. Since individual rotational lines are not resolved in our spectra, several assumptions are required to simulate these transitions. For calculating the energies of the rotational transitions in each band, equilibrium rotational constants B_e are assumed, using the R_m values from the potentials obtained in the vibrational analysis described in chapter 2.

* Most of this section was originally published in slightly revised form in *J. Chem. Phys.* **101**, 6538 (1994), with co-authors Y. Zhao, G. Reiser, C. C. Arnold, and D. M. Neumark.

We now derive expressions for the relative intensities of the rotational transitions in each band. Xie and Zare[1,2] derived an expression for the photoionization probability of a diatomic molecule conforming to Hund's case (b) in terms of a generalized rotational line strength factor and the reduced multipole moments. Here, we adapt this approach to a Hund's case (c) molecule. A diagram of the Hund's case (c) angular momentum scheme is presented for reference in Figure B1. The results of this treatment are valid for photodetachment as well as photoionization. Although we do not resolve individual rotations in the spectra presented in this work, the results presented below should be useful for future investigations in which rotational structure is resolved. Moreover, the final expressions obtained are quite simple, and result in a very satisfactory fitting of the peak shapes.

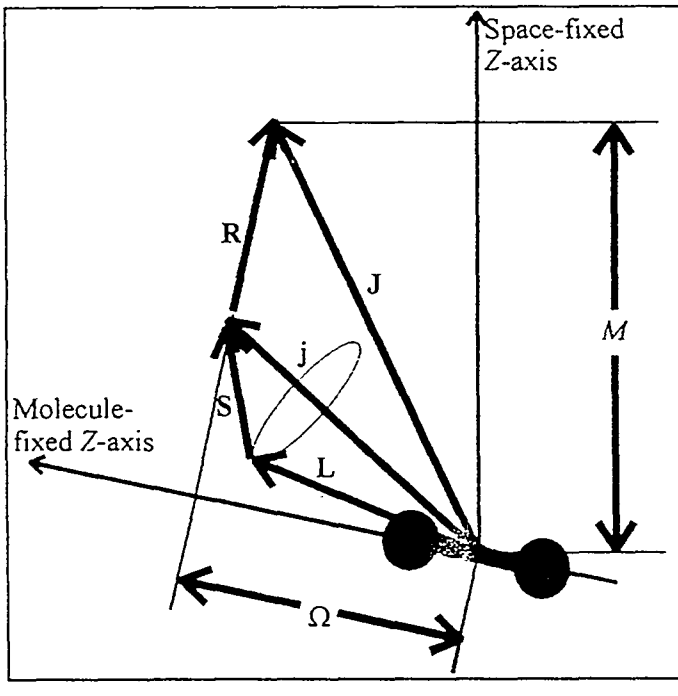

Figure B1. Hund's case (c). **L** and **S** are the orbital and spin angular momenta, which are strongly coupled to form the total electronic angular momentum **j**. **R** is the rotational angular momentum, which adds to **j** to form the total angular momentum **J**. The good quantum numbers are J, M (the projection of **J** on the space fixed axis), and Ω (the projection of **J** on the molecule-fixed axis), which defines the electronic state of the molecule.

The eigenstates of a case (c) molecule are represented by

$$|nJ\Omega Mv\rangle = |J\Omega M\rangle|n\Omega\rangle|v\rangle \qquad \text{(B1)}$$

where the angular momentum quantum numbers are defined in Table B1; v is the vibrational quantum number, and n represents the remaining quantum numbers.

Table B1. Quantum number nomenclature for rotational analysis.

Angular Momentum	Molecule-Fixed Projection	Space-Fixed Projection	Description
J	Ω	M	Total angular momentum of neutral
J^-	Ω^-	M^-	Total angular momentum of anion
j	ω	m	Total photoelectron angular momentum (orbital + spin)
1	μ	μ_0	Angular momentum of photon
k	q	p	Vector sum of photon and photoelectron angular momenta

On the right side of Equation B1 we have assumed the Born-Oppenheimer approximation and independence of rotational from vibrational degrees of freedom to separate the eigenstate into rotational, electronic, and vibrational parts. The detached electron is assumed to be well approximated asymptotically by a partial wave expansion in a spherically symmetric potential.[3] The eigenstate of a given electron partial wave in the coupled representation appropriate to case (c) is $|jm\rangle$, where j is the total (orbital and spin) angular momentum of the electron partial wave, and m is its projection in the space-fixed frame. The electronic dipole operator transforms as a spherical tensor, $T(1,\mu_0)$, under rotation. The photodetachment probability for a given electron angular momentum from rotational state J^- of the anion to state J of the neutral is then (neglecting constants and the vibrational eigenvectors)

$$P_j(J^-,J) \propto \sum_m \sum_{M,M^-} \left| \langle J\Omega M | \langle n\Omega | \langle jm | T(1,\mu_0) | n^-\Omega^- \rangle | J^-\Omega^- M^- \rangle \right|^2 . \qquad \text{(B2)}$$

It is necessary to get the electron and photon wavefunctions into the molecule-fixed frame, so that we can separate nuclear and electron coordinates. To do this we transform the dipole operator and electron eigenstates to the molecule fixed frame using the Wigner rotation matrices as defined by Zare[4] as follows:

$$\begin{aligned} T(1,\mu_0) &= \sum_\mu D^{1*}_{\mu_0\mu}(R) T(1,\mu) \qquad \text{(photon)}, \\ \langle jm| &= \sum_\omega \left[D^{j*}_{m\omega}(R) \right]^+ \langle j\omega| \qquad \text{(electron)}, \end{aligned} \qquad \text{(B3)}$$

where R stands for the Euler angles ϕ, θ, and χ, and the angular momentum quantum numbers are defined in Table B1.

The rotational eigenfunctions are

$$\begin{aligned} |J^-\Omega^- M^-\rangle &= \left(\frac{2J^-+1}{8\pi^2} \right)^{1/2} D^{J^-*}_{M^-\Omega^-}(R), \\ \langle J\Omega M| &= \left(\frac{2J+1}{8\pi^2} \right)^{1/2} \left[D^{J^-*}_{M\Omega}(R) \right]^+ . \end{aligned} \qquad \text{(B4)}$$

Substituting Equations B3 and B4 into Eq. B2 gives

$$\begin{aligned} P_j(J^-,J) \propto & \frac{(2J+1)(2J^-+1)}{(8\pi^2)^2} \\ & \times \sum_m \sum_{M,M^-} \left| \sum_{\mu,\omega} \left(\int D^J_{M\Omega}(R) D^j_{m\omega}(R) D^1_{-\mu_0-\mu}(R) D^{J^-}_{-M^--\Omega^-}(R) dR \right) \right. \qquad \text{(B5)} \\ & \left. \times \langle n\Omega | \langle j\omega | T(1,\mu) | n^-\Omega^- \rangle \right|^2 , \end{aligned}$$

where *all the dependence on angular nuclear coordinates is contained in the integral over the four rotation matrices.*

To evaluate the integral over the four rotation matrices, we expand those from the detached electron and dipole operator in a Clebsch-Gordon series[4]

$$D^{j}_{m\omega}(R)D^{1}_{-\mu_0-\mu}(R)=\sum_k(2k+1)\begin{pmatrix} j & 1 & k \\ m & -\mu_0 & p\end{pmatrix}\begin{pmatrix} j & 1 & k \\ \omega & -\mu & q\end{pmatrix}D^{k}_{-p-q}(R) \qquad \text{(B6)}$$

with

$$k=1+j,\cdots,|1-j|,$$
$$p=\mu_0-m,$$
$$q=\mu-\omega.$$

In Equation (B6) we have *coupled the electron and photon angular momenta*, where k is the vector sum of the photon electron angular momenta, and p and q are the space- and molecule-fixed projections of this quantity, respectively (see Table B1). Substituting this into Equation (B5) we obtain an integral over three rotation matrices[4]

$$\int D^{J}_{M\Omega}(R)D^{k}_{-p-q}(R)D^{J^-}_{-M^--\Omega^-}(R)dR=8\pi^2\begin{pmatrix} J^- & k & J \\ M^- & p & -M\end{pmatrix}\begin{pmatrix} J^- & k & J \\ \Omega^- & q & -\Omega\end{pmatrix}. \qquad \text{(B7)}$$

Using Eqs. (B6) and (B7) in Eq. (B5) we obtain

$$\begin{aligned} P_j(J^-,J) \propto{}& (2J+1)(2J^-+1) \\ &\times\sum_m\sum_{M,M^-}\left|\sum_k(2k+1)\begin{pmatrix} j & 1 & k \\ m & -\mu_0 & p\end{pmatrix}\begin{pmatrix} J^- & k & J \\ M^- & p & -M\end{pmatrix}\begin{pmatrix} J^- & k & J \\ \Omega & \Omega-\Omega^- & -\Omega\end{pmatrix}\right. \qquad \text{(B8)} \\ &\left.\times\sum_{\mu,\omega}\begin{pmatrix} j & 1 & k \\ \omega & -\mu & q\end{pmatrix}\langle n\Omega|\langle j\omega|T(1,\mu)|n^-\Omega^-\rangle\right|^2. \end{aligned}$$

The condition $q=\Omega-\Omega^-$ (*i.e.*, $\mu+\Omega^-=\omega+\Omega$) determines the electronic selection rules.

The M and M^- dependent terms in the square of the sum over k for a given k' and k'' give rise to sums

$$\sum_{M,M^-}\begin{pmatrix} J^- & k' & J \\ M^- & p & -M \end{pmatrix}\begin{pmatrix} J^- & k'' & J \\ M^- & p & -M \end{pmatrix}=\frac{1}{2k'+1}\delta_{k'k''} \tag{B9}$$

so that the cross terms vanish, leaving

$$P_j(J^-,J)\propto\sum_k S_k(J^-,J)\left|\mu_j(k,q)\right|^2. \tag{B10}$$

Here, following Xie and Zare,[1,2] we have defined the rotational line strength factor*

$$S_k(J^-,J)=(2J+1)(2J^-+1)\left|\begin{pmatrix} J^- & k & J \\ \Omega^- & \Omega-\Omega^- & -\Omega \end{pmatrix}\right|^2 \tag{B11}$$

which contains all the J and J^- dependence, and a generalized multipole transition moment

$$\left|\mu_j(k,q)\right|^2=(2k+1)\left|\sum_{\mu,\omega}\begin{pmatrix} j & 1 & k \\ \omega & -\mu & q \end{pmatrix}\langle n\Omega|\langle j\omega|T(1,\mu)|n^-\Omega^-\rangle\right|^2. \tag{B12}$$

The phrase "generalized multipole transition moment" refers to the fact that this term involves both the incoming photon and ejected electron angular momenta, in contrast to the familiar "dipole transition moment" in photo absorption processes which involves only the photon.

In anion ZEKE spectroscopy, $j=\frac{1}{2}$ (because l=0 for the detached electrons that are detected) and Eq. (B10) becomes

$$P_{1/2}(J^-,J)\propto C_{1/2}S_{1/2}(J^-,J)+S_{3/2}(J^-,J) \tag{B13}$$

---

* The expression for the rotational line strength factor may be compared with the familiar expression for the Hönl-London factors for the intensity of rotational transitions in the photoabsortion process $J'',K''\rightarrow J',K'$ of a rigid rotor:[4]

$$S(J'K',J''K'')=(2J'+1)(2J''+1)\left|\begin{pmatrix} J' & 1 & J'' \\ -K' & K'-K'' & K'' \end{pmatrix}\right|^2.$$

with

$$C_{1/2} = \left| \frac{\mu_{1/2}\left(\frac{1}{2}, q\right)}{\mu_{1/2}\left(\frac{1}{2}, q\right)} \right|^2 . \tag{B14}$$

For s-wave detachment, $\Delta J = J^- - J = \pm\frac{1}{2}, \pm\frac{3}{2}$. Expressions for the line strength factors for transitions from $\Omega^- = 0$ to $\Omega = \frac{1}{2}$ ($X\,\frac{1}{2}$ and $II\,\frac{1}{2}$ states) and to $\Omega = \frac{3}{2}$ ($I\,\frac{3}{2}$ state) for each of the allowed branches are given in Table B2. Note that $S_{1/2}(J^-, J)$ vanishes for the $\Delta J = \pm\frac{3}{2}$ branches of the transitions to $\Omega = \frac{1}{2}$ states and for all branches of transitions to the $\Omega = \frac{3}{2}$ state.

Table B2. Rotational line strengths for transitions to the three neutral states

ΔJ	$S_{1/2}(J^-, J)$		$S_{3/2}(J^-, J)$	
	$I\,\frac{3}{2}$	$X\,\frac{1}{2}$ or $II\,\frac{1}{2}$	$I\,\frac{3}{2}$	$X\,\frac{1}{2}$ or $II\,\frac{1}{2}$
$+\frac{1}{2}$	0	$J^- + 1$	$\frac{3(J^- + 1)(J^- + 2)}{2(2J^- + 3)}$	$\frac{J^-(J^- + 1)}{2(2J^- + 3)}$
$-\frac{1}{2}$	0	J^-	$\frac{3J^-(J^- - 1)}{2(2J^- - 1)}$	$\frac{J^-(J^- + 1)}{2(2J^- - 1)}$
$+\frac{3}{2}$	0	0	$\frac{(J^- + 2)(J^- + 3)}{2(2J^- + 3)}$	$\frac{3(J^- + 1)(J^- + 2)}{2(2J^- + 3)}$
$-\frac{3}{2}$	0	0	$\frac{(J^- - 1)(J^- - 2)}{2(2J^- - 1)}$	$\frac{3J^-(J^- - 1)}{2(2J^- - 1)}$

One can calculate a rotational stick spectrum by multiplying Eq. (B13) by a Boltzmann factor, treating the coefficient $C_{1/2}$ (for the $\Omega = \frac{1}{2}$ neutral states) as an adjustable parameter. To obtain realistic peak shapes, the rotational lines must be

convoluted with the empirically determined ZEKE instrumental line shape. Refer back to Chapter 3 [Eq. (3.11)] for the detailed form of the ZEKE line shape.

B2. Documentation of the rotational fitting program "rfit": example of the XeBr $X\,\frac{1}{2}$ state

This section describes the computer program used to fit the rotational profiles of the RgX^- ZEKE spectra. The program as input uses the vibrational stick spectrum generated by the fitting program "idvr" (described in Appendix A) to general the final rotational spectrum convoluted with the ZEKE line shape. The available variable parameters in the rotational fitting procedure are the rotational temperature, the ZEKE FWHM (full width at half maximum), the parameter $C_{1/2}$ discussed above [see Eqs. (B13) and (B14)], the baseline, an offset for the origin, and the maximum peak intensity. Note that the input vibrational sticks are not modified in any way: the program "rfit" only fits the above mentioned above. The program uses a simple gradient minimization algorithm[5] to perform the optimization.

B2.1. The input files

An example of the input file for the $X\,\frac{1}{2}$ state of XeBr is shown below. The line numbers in boldface are only for reference, and are not included in the actual input file. The file is named "xebr_x_rfin." The comments after the "!*" symbol may be included after numerical input lines, but not after text (file name) input lines.

```
1:  1        !* Mode (1 fit & uncer, 2 fit only, 3 uncertainties only)
2:  2        !* Output type (1 long, 2 short)
3:  xebr_x_exp
```

```
4:  500.     !* Peak electron counts
5:  1        !* # of vib stick input files (1 or 2)
6:  xebr_x_stx
7:  0        !* State for file 1 (0 X, 1 I, 2 II)
8:  3.82     !* Neutral rmin file 1 (Ang)
9:  3.81     !* Anion rmin (Ang)
10: 131.30, 79.909      !* mass1, mass2 (amu)
11: xebr_x_fit
12: y        !* Log of C1/2 for file 1-vary?
13: y        !*    -Uncertainty?
14: -0.      !*    -Initial value
15: 0.05     !*    -Step size
16: -2.0,2.0       !* -Min,max
17: n        !* Relative intensity of file 1 - vary?
18: n        !*    -Uncertainty?
19: 1.       !*    -Initial value
20: 0.01     !*    -Step size
21: 0.1,1.5  !*    -Min, max
22: n        !* Rot. Temp - vary?
23: n        !*    -Uncertainty?
24: 40.      !*    -Initial value
25: 2.       !*    -Step size
26: 1.0,200.       !* -Min, max
27: y        !* ZEKE FWHM - vary?
28: y        !*    -Uncertainty?
29: 3.0      !*    -Initial value (cm-1)
30: 0.1      !*    -Step size
31: 0.1,25.  !*    -Min, max
32: y        !* Origin shift - vary?
33: n        !*    -Uncertainty?
34: -0.0     !*    -Initial value (cm-1)
35: 0.02     !*    -Step size
36: -3., 3.  !*    -Min, max
37: y        !* Baseline - vary?
38: n        !*    -Uncertainty?
39: 0.00     !*    -Initial value (frac)
40: 0.01     !*    -Step size
41: 0.,0.5   !*    -Min, max
```

The first line is an integer which tells the program whether to find uncertainties (standard deviations) during the fit or not. If line **1** is set to 1 then both the fit is performed and uncertainties are found. If line **2** is set to 2, only the fit is performed, and if it is set to 3 only uncertainties are found, using the initial values given for the fitting parameters. The program performs a full multivariate analysis of the parameters in finding the uncertainties, so this can be quite time-consuming, and it is often desirable to run in mode 2 when quick results are needed. Line **2** is an integer: 1 for long form

output, *i.e.* each step in the gradient minimization is shown, 2 for short form output, where only the final result of the fit are shown, and the intermediate steps during the process of finding the standard deviations.

Line **3** contains the name of the experimental spectrum, which must be in wavenumber format. If uncertainties are to be calculated, line **4** should contain an estimate of the number of electron counts at peak in the experimental spectrum. This number is used in the calculation of the absolute value of χ^2 (Ref. 5). When uncertainties are to be calculated, the standard deviation is defined by a change in a given parameter, with re-optimization of the other parameters, such that χ^2 is increased by one.[5] If uncertainties are not calculated, the absolute value of χ^2 is not important.

Line **5** indicates the number of vibrational stick spectra to be read in (either 1 or 2). In this example only one file is read in. However, one may read in, for example. both the vibrational stick spectra for the $X\,\frac{1}{2}$ and $I\,\frac{3}{2}$ electronic states for simultaneous fitting with the same experimental spectrum. If two vibrational stick files were specified, the input file would have to contain additional lines to specify fitting parameters for the second electronic state; for details see the listing of the source code in Section B3 below. Line **6** contains the file name of the vibrational stick spectrum for the first (and in this example, only) electronic state. Line **7** is an integer which specifies the case (c) electronic state: 1 for the $X\,\frac{1}{2}$ state, 2 for the $I\,\frac{3}{2}$ state, or 3 for the $II\,\frac{1}{2}$ state. This input determines the form of the rotational line strength factors, as shown in Table B2. Lines **8** and **9** specify the neutral and anion bond lengths, respectively, in Å. Line **10** lists the atomic masses in amu. This information is used to calculate B_e.

Line **11** specifies the name of the file to which the final rotational spectrum, convoluted with the ZEKE line shape, is saved. This spectrum is in wavenumber format, with the same number and spacing of points as the experimental spectrum.

The remaining line, **12-41**, are grouped into sets of five lines for each parameter in the fit. The first line of each set is set to "y" if the parameter is to be adjusted during the fit, or to "n" if it is to be fixed at its initial value. The second line of the set is set to "y" if one wishes to find the standard deviation of a given parameter or "n" if not. Note that line **1** of the input file must be set to 1 or 3 for any uncertainties to be computed. The third line of each set specifies the initial value of the parameter. The fourth line specifies the *initial* step size for the gradient minimization. The step size is adjusted as the fit progresses. The fifth line lists the minimum and maximum allowable values of the parameters. An explanation of each parameter and is given below:

Lines 12-16: base 10 log of $C_{1/2}$ [see Eqs. (B13) and (B14)]

Lines 17-21: intensity of the highest peak of the convoluted spectrum relative to the experimental spectrum. If two electronic states are included in the fit, the intensity of one may be fixed and the other allowed to vary in order to determine the relative transition strengths.

Lines 22-26: the rotational temperature of the anion in Kelvin. A Boltzmann distribution of anion rotational states is assumed.

Lines 27-31: the ZEKE FWHM in cm^{-1}

Lines 32-36: the shift in the rotational origin, in cm^{-1}

Lines 37-41: the baseline added to the convoluted spectrum, as a fraction of the maximum peak height.

B2.2. Running the program

One runs the program in the usual Unix fashion, by piping in the input file. If one is calculating standard deviations as well as performing a fit it is usually desirable to run the program in the background, as in the following example using the input file "xebr_x_rfin" discussed above:

```
> nice +19 rfit < xebr_x_rfin > xebr_x_rfout &
>
```

The output file "xebr_x_rfout" then contains the optimized parameters and standard deviations, and the convoluted spectrum is saved to the file "xebr_x_fit" named in the input file.

B2.3. Outline of the program

The program "rfit" is contained in only one file, "rfit.f." There is no makefile; the program is recompiled with a command such as:

```
> f77 -O -o rfit rfit.f -C
```

The subroutines and functions used by the program are listed in Table B3.

Table B3. Subroutines and functions used by the rotational fitting program "rfit".

Subroutine or Function	Description
rfit	main program
optchi	general-purpose subroutine implementing the gradient optimization method
chisquar	calculates χ^2 for a given set of fitting parameters
rsticks	calculates a rotational stick spectrum given one or two vibrational stick spectra, and the fitting parameters.
be	function to calculate the rotational constant
boltz	calculates the Boltzmann factor for a given rotational temperature and anion rotational state
conzeke	convolutes the rotational stick spectrum with the ZEKE line shape
cp2	compares the experimental and convoluted spectra and calculates χ^2

B3. Source code for the rotational fitting program "rfit"

```
      program rfit


      implicit undefined(a-z)
*
      real frac,evtocm
      parameter (frac=0.1,evtocm=8065.541)
      real vstk(2,2,100)
      real temp,org,baseln
      real beneu,bean,reneu(2),rean
      real mass1,mass2,rmass
      integer ninfil,i,j,k,state(2),j,nvstk(2),rct
      integer mpts
      character*25 infile(2), outfile,especfil
      real relfcf(2),scoef(2)
      real fwhmcm,fwhmev,delta,deltawn
      real smooth(2,100000),espec(2,100000),espec2(2,100000),
     &  rstk(2,500000)
      integer nspec
      real dfwhmcm,dtemp,dscoef(2),drelfcf(2),dorg,dbaseln
      real mfwhmcm,mtemp,mscoef(2),mrelfcf(2),morg,mbaseln
      real xfwhmcm,xtemp,xscoef(2),xrelfcf(2),xorg,xbaseln
      real s1fwhm,s1temp,s1scoef(2),s1relfcf(2),s1org,s1baseln
      real s2fwhm,s2temp,s2scoef(2),s2relfcf(2),s2org,s2baseln
      logical ufwhm,utemp,uscoef(2),urelfcf(2),uorg,ubaseln
      real chisq,xinc,chiopt
      character*1 ans
      logical vscoef(2),vrelfcf(2),vtemp,vfwhm,vorg,vbaseln
      integer ncomp
      real a(30),da(30),ma(30),xa(30),aopt(30),abest(30)
      real s1a(30),s2a(30)
      logical va(30),ua(30),v1
      integer nparm,mode,tout
      logical pr
      real ncounts,chilast,slast,slope
*
      common /conv1/ fwhmcm,fwhmev,delta,deltawn
      common /conv2/ mpts
      common /func/ bean,beneu
      common /rvars/ mass1,mass2,rmass,rean,temp,org,baseln
      common /ivars/ rct,ninfil
      common/arrays/reneu(2),state(2),nvstk(2),
     &  relfcf(2),scoef(2)
      common /comp/ chisq,ncomp
      common /comp2/ ncounts
      common /opti/ nparm
      common /optr/ a(30),abest(30)
      common /sticks/ vstk(2,2,100),rstk(2,500000)
      common /spec1/ smooth(2,100000),espec(2,100000)
      common /spec2/ nspec
      common /pr1/ pr,tout
*-----------------------------------------------------------------
*  Read in constants and fitting parameters
```

```
*-----------------------------------------------------------------
      write (*,*)
      write (*,*) 'Rotational Fitting Program'
      write (*,*) ''
      write (*,*) 'Choose mode:'
      write (*,*) '  1) Fit and find uncertainties'
      write (*,*) '  2) Fit only'
      write (*,*) '  3) Find uncertainties only'
      read (*,*) mode
      write(*,*) 'Type of output:'
      write(*,*) '  1) Long -- show all steps.'
      write(*,*) '  2) Short -- show only optimized values'
      read (*,*) tout
      write (*,*) 'Enter name of file containing experimental spectrum'
      write(*,*) 'in inverted cm-1 format:'
      read (*,*) especfil
*
      open (8,file=especfil)
      do i=1,99999
        read (8,*,end=112) espec(1,i),espec(2,i)
      enddo
112   nspec=i-1
      write(*,*)
      write (*,*) nspec,' points read from ',especfil
      write(*,*) 'ranging from',espec(1,1),' to',espec(1,nspec)
      if (espec(1,nspec).lt.espec(1,1)) then
        write(*,*) 'Note that experimental file is backwards'
        do j=1,nspec
          espec2(1,j) = espec(1,nspec-j+1)
          espec2(2,j) = espec(2,nspec-j+1)
        enddo
        do j=1,nspec
          espec(1,j) = espec2(1,j)
          espec(2,j) = espec2(2,j)
        enddo
      endif

      write(*,*)
      write(*,*) 'How many electron counts at peak (determines the'
      write(*,*) 'uncertainty)?'
      read(*,*) ncounts
*
      write (*,*) 'How many vibrational stick input files? (1-2)'
      read (*,*) ninfil
      do i = 1,ninfil
        write (*,*) 'Enter name of vib. stick file',i
        read (*,*) infile(i)
        write (*,*) 'Enter neutral state for ', infile(i)
        write (*,*) '   0) X 1/2 State'
        write (*,*) '   1) I 3/2 State'
        write (*,*) '   2) II 1/2 State'
        read (*,*) state(i)
        write (*,*) 'Enter neutral rmin (angstroms) for ',infile(i)
        read (*,*) reneu(i)
      enddo

      write (*,*) 'Enter rmin (angstroms) of anion : '
```

```
read (*,*) rean
write (*,*) 'Enter masses of atoms (amu) : '
read (*,*) mass1, mass2

write (*,*) 'Enter name of file to save fit spectrum:'
read (*,*) outfile
write (*,*)
write (*,*) 'Fitting Parameters:'
dscoef(1)=0.
dscoef(2)=0.
vscoef(1)=.false.
vscoef(2)=.false.
vrelfcf(1)=.false.
vrelfcf(2)=.false.
do i=1,ninfil
  if (state(i).ne.1) then
    write(*,*)
    write(*,*) '  Log of coefficient of S1/2 for file ',infile(i)
    write(*,*) '    Vary? (Y/N) '
    read(*,*) ans
    vscoef(i)=(ans.eq.'y').or.(ans.eq.'Y')
    write(*,*) '    Find uncertainty? (Y/N) '
    read(*,*) ans
    uscoef(i)=((ans.eq.'y').or.(ans.eq.'Y'))
    write(*,*) '    Initial value : '
    read (*,*) scoef(i)
    write(*,*) '    Step size : '
    read(*,*) dscoef(i)
    write(*,*) '    Min, max : '
    read(*,*) mscoef(i),xscoef(i)
  else
    scoef(i)=0.
  endif
  write(*,*)
  write(*,*) '  Relative intensity of file ',infile(i)
  write(*,*) '    Vary? (Y/N) '
  read(*,*) ans
  vrelfcf(i)=(ans.eq.'y').or.(ans.eq.'Y')
  write(*,*) '    Find uncertainty? (Y/N) '
  read(*,*) ans
  urelfcf(i)=(ans.eq.'y').or.(ans.eq.'Y')
  write(*,*) '    Initial value : '
  read(*,*) relfcf(i)
  write(*,*) '    Step size : '
  read(*,*) drelfcf(i)
  write(*,*) '    Min, max : '
  read(*,*) mrelfcf(i),xrelfcf(i)
enddo
write(*,*)
write(*,*) '  Rotational temperature (K):'
write(*,*) '    Vary? (Y/N) '
read(*,*) ans
vtemp=(ans.eq.'y').or.(ans.eq.'Y')
write(*,*) '    Find uncertainty? (Y/N) '
read(*,*) ans
utemp=(ans.eq.'y').or.(ans.eq.'Y')
write (*,*) '    Initial value : '
```

```
      read (*,*) temp
      write(*,*) '    Step size : '
      read(*,*) dtemp
      write(*,*) '    Min, max : '
      read(*,*) mtemp,xtemp
      write(*,*)
      write(*,*) '  ZEKE instrumental FWHM (cm-1):'
      write(*,*) '    Vary? (Y/N) '
      read(*,*) ans
      vfwhm=(ans.eq.'y').or.(ans.eq.'Y')
      write(*,*) '    Find uncertainty? (Y/N) '
      read(*,*) ans
      ufwhm=(ans.eq.'y').or.(ans.eq.'Y')
      write(*,*)'    Initial value : '
      read(*,*) fwhmcm
      fwhmev = fwhmcm/evtocm
      write(*,*) '    Step size : '
      read(*,*) dfwhmcm
      write(*,*) '    Min, max : '
      read(*,*) mfwhmcm,xfwhmcm
      write(*,*)
      write(*,*) '  Origin shift from given vibrational sticks (cm-1):'
      write(*,*) '    Vary? (Y/N) '
      read(*,*) ans
      vorg=(ans.eq.'y').or.(ans.eq.'Y')
      write(*,*) '    Find uncertainty? (Y/N) '
      read(*,*) ans
      uorg=(ans.eq.'y').or.(ans.eq.'Y')
      write(*,*) '    Initial value : '
      read(*,*) org
      write(*,*) '    Step size : '
      read(*,*) dorg
      write(*,*) '    Min, max : '
      read(*,*) morg,xorg
      write(*,*)
      write(*,*) '  Baseline (as fraction of intensity):'
      write(*,*) '    Vary? (Y/N) '
      read(*,*) ans
      vbaseln=(ans.eq.'y').or.(ans.eq.'Y')
      write(*,*) '    Find uncertainty? (Y/N) '
      read(*,*) ans
      ubaseln=(ans.eq.'y').or.(ans.eq.'Y')
      write(*,*) '    Initial value : '
      read(*,*) baseln
      write(*,*) '    Step size : '
      read(*,*) dbaseln
      write(*,*) '    Min, max : '
      read(*,*) mbaseln,xbaseln
*
*-----------------------------------------------------------------------
*  Read in vibrational sticks in cm-1 format
*-----------------------------------------------------------------------
      do i = 1,ninfil
        j=6+i
        open (j, file = infile(i))
        do k = 1,100
          read (j,*,end=200) vstk(i,1,k), vstk(i,2,k)
```

```
        enddo
200     nvstk(i)=k-1
        close (j)
        write (*,*)
        write (*,*) nvstk(i),' vibrational sticks read from '
     &   ,infile(i)
      enddo
      write(*,*)
*
*-----------------------------------------------------------------
*  Assign initial values to parameters
*-----------------------------------------------------------------
      nparm=8
*
      a(1)=fwhmcm
      a(2)=temp
      a(3)=org
      a(4)=scoef(1)
      a(5)=scoef(2)
      a(6)=relfcf(1)
      a(7)=relfcf(2)
      a(8)=baseln
*
      da(1)=dfwhmcm
      da(2)=dtemp
      da(3)=dorg
      da(4)=dscoef(1)
      da(5)=dscoef(2)
      da(6)=drelfcf(1)
      da(7)=drelfcf(2)
      da(8)=dbaseln
*
      ma(1)=mfwhmcm
      ma(2)=mtemp
      ma(3)=morg
      ma(4)=mscoef(1)
      ma(5)=mscoef(2)
      ma(6)=mrelfcf(1)
      ma(7)=mrelfcf(2)
      ma(8)=mbaseln
*
      xa(1)=xfwhmcm
      xa(2)=xtemp
      xa(3)=xorg
      xa(4)=xscoef(1)
      xa(5)=xscoef(2)
      xa(6)=xrelfcf(1)
      xa(7)=xrelfcf(2)
      xa(8)=xbaseln
*
      va(1)=vfwhm
      va(2)=vtemp
      va(3)=vorg
      va(4)=vscoef(1)
      va(5)=vscoef(2)
      va(6)=vrelfcf(1)
      va(7)=vrelfcf(2)
```

```
      va(8)=vbaseln
*
      ua(1)=ufwhm
      ua(2)=utemp
      ua(3)=uorg
      ua(4)=uscoef(1)
      ua(5)=uscoef(2)
      ua(6)=urelfcf(1)
      ua(7)=urelfcf(2)
      ua(8)=ubaseln
*-------------------------------------------------
*  Optimize parameters
*-------------------------------------------------
      write(*,1400) 'FWHM','R.Temp','Scoef(1)','Inten(1)'
     & ,'Inten(2)','Org. Shft.','Baseln.','Chi-Sq.'
1400  format (8 a10)

      if (mode.eq.3) then
        call chisquar
        go to 403
      endif

      call optchi(a,da,ma,xa,va,nparm)

1700  open(15,file=outfile)
      do i = 1, nspec
        write (15, 74) smooth(1,i), smooth(2,i)
 74     format(f10.2,f10.6)
      enddo
      close(15)
      write(*,*)
      write(*,*) nspec, ' points saved in ',outfile
      write(*,*)

403   do i=1,nparm
        aopt(i)=a(i)
      enddo
      chiopt=chisq
*-------------------------------------------------------------------------
*  Find uncertainties by finding displacement in each parameter needed
*  to increase chi-square by 1.  Paramaters are displaced in steps of
*  da(i), and linear interpolation is used to find delta-chi-square=1.
*-------------------------------------------------------------------------
      if (mode.eq.2) go to 1003
      do i=1,nparm
        s1a(i)=0.
        s2a(i)=0.
      enddo

      do i=1,nparm
        if (ua(i)) then
          write(*,*) 'Finding uncertainty in a(',i,')'
          write(*,1400) 'FWHM  ','Rot.Temp','Scoef(1)',
     &      'Inten(1)','Inten(2)','Org. Shift','Baseline',
     &      'Chi-Sq.'
          v1=va(i)
          va(i)=.false.
```

```
          do j=1,nparm
            a(j)=aopt(j)
          enddo
*         ------------------
*         Pos. uncertainties
*         ------------------
          chilast=chiopt
          slast=0.0
700       xinc=a(i)+da(i)
          if (xinc.ge.xa(i)) then
            s1a(i)=xa(i)-aopt(i)
            go to 601
          endif
          a(i)=xinc
          call optchi(a,da,ma,xa,va,nparm)
          if ((chisq-chiopt).ge.1) then
            s1a(i)=a(i)-aopt(i)
            go to 600
          else
            chilast=chisq
            slast=a(i)-aopt(i)
          endif
          go to 700
*         ----------------------------
*         Linear interpolation for pos.
*         ----------------------------
600       slope=(chisq-chilast)/(s1a(i)-slast)
          s1a(i)=((1-chilast+chiopt)/slope)+slast
*         ------------------
*         Neg. uncertainties
*         ------------------
601       chilast=chiopt
          slast=0.0
          do j=1,nparm
            a(j)=aopt(j)
          enddo
800       xinc=a(i)-da(i)
          if (xinc.le.ma(i)) then
            s2a(i)=aopt(i)-ma(i)
            go to 510
          endif
          a(i)=xinc
          call optchi(a,da,ma,xa,va,nparm)
          if ((chisq-chiopt).ge.1) then
            s2a(i)=aopt(i)-a(i)
            go to 500
          else
            chilast=chisq
            slast=aopt(i)-a(i)
          endif
          go to 800
*         ----------------------------
*         Linear interpolation for neg.
*         ----------------------------
500       slope=(chisq-chilast)/(s2a(i)-slast)
          s2a(i)=((1-chilast+chiopt)/slope)+slast
510       va(i)=v1
```

```
       endif
      enddo
*
      fwhmcm=aopt(1)
      temp=aopt(2)
      org=aopt(3)
      scoef(1)=aopt(4)
      scoef(2)=aopt(5)
      relfcf(1)=aopt(6)
      relfcf(2)=aopt(7)
      baseln=aopt(8)
*
      s1fwhm=s1a(1)
      s1temp=s1a(2)
      s1org=s1a(3)
      s1scoef(1)=s1a(4)
      s1scoef(2)=s1a(5)
      s1relfcf(1)=s1a(6)
      s1relfcf(2)=s1a(7)
      s1baseln=s1a(8)
*
      s2fwhm=s2a(1)
      s2temp=s2a(2)
      s2org=s2a(3)
      s2scoef(1)=s2a(4)
      s2scoef(2)=s2a(5)
      s2relfcf(1)=s2a(6)
      s2relfcf(2)=s2a(7)
      s2baseln=s2a(8)
*
*  Print Optimized parameters & uncertainties
*
1003  write(*,*)
      write(*,*) 'Optimized Paramaters:'
      write(*,*)
      write(*,1005)' ',' ','+sigma ','-sigma '
1005  format(a22, 3 a11)
      do i=1,ninfil
        if (state(i).ne.1) then
          write(*,1011)'log(S1/2 Coef.) ',i,scoef(i),s1scoef(i),
     &     s2scoef(i)
          write(*,1011)'S1/2 Coef. ',i,10**scoef(i),
     &     10**(scoef(i)+s1scoef(i))-10**scoef(i),
     &     -(10**(scoef(i)-s2scoef(i))-10**scoef(i))
        endif
        write(*,1011)'Rel. Intensity ',i,relfcf(i),s1relfcf(i),
     &    s2relfcf(i)
       enddo
       write(*,1010) 'Rot. Temp. (K)',temp,s1temp,s2temp
       write(*,1010) 'ZEKE FWHM (cm-1)',fwhmcm,s1fwhm,s2fwhm
       write(*,1010) 'Origin Shift',org,s1org,s2org
       write(*,1010) 'Baseline',baseln,s1baseln,s2baseln
1010  format(a22, 3 f11.6)
1011  format(a21, i1, 3 f11.6)
      write(*,*)
      write(*,1030) 'Chi-Square',chiopt
1030  format(a22,f11.6)
```

```
      write(*,*)

      end
*
*=====================================================================
*  Subroutine to find optimal parameters, given starting parameter
*  values, a; parameters to vary, va; min and max parm values, ma,
*  xa.
*=====================================================================
      subroutine optchi(a,da,ma,xa,va,nparm)
*          .
      parameter (frac=0.1,niter=20)
      integer nparm,rct,ninfil
      real a(30),da(30),ma(30),xa(30)
      real ga(30)
      real abest(30)
      logical va(30),vany,pr
*
      real chisq,chibest,chinow,chiprev,gnorm,xinc
      integer i,count,ncomp,tout
*
      common /comp/ chisq,ncomp
      common /ivars/ rct,ninfil
      common /pr1/ pr,tout
      common /spec2/ nspec
*
*  Find gradient of Chi-Square
*
      pr = .true.
      call chisquar
      if (tout.eq.2) then
        pr = .false.
      endif
      chinow=chisq
      chibest=chisq
      do i=1,nparm
        abest(i)=a(i)
      enddo

      count=1
      vany=.false.
      do i=1,nparm
        if (va(i)) then
          vany=.true.
        endif
      enddo
      if (.not.vany) then
        return
      endif

1500  do i=1,nparm
        ga(i)=0
      enddo

      pr = .false.
      do i=1,nparm
        if (.not.va(i)) go to 300
```

```
        a(i)=a(i)+frac*da(i)
        call chisquar
        a(i)=a(i)-frac*da(i)
        ga(i)=(chisq-chinow)/frac
300   enddo
      if (tout.ne.2) then
        pr = .true.
      endif
*---------------------
*  Normalize gradient
*---------------------
      gnorm=0.
      do i=1,nparm
        gnorm=gnorm+ga(i)**2
      enddo
      gnorm=sqrt(gnorm)
      do i=1,nparm
        ga(i)=ga(i)/gnorm
      enddo
*-------------------------------------------------------
*  Increment parameters in direction of -gradient & find
*  chi-square
*-------------------------------------------------------
1600  do i=1,nparm
        xinc=a(i)-ga(i)*da(i)
        if (xinc.lt.ma(i)) then
          a(i)=ma(i)
        elseif (xinc.gt.xa(i)) then
          a(i)=xa(i)
        else
          a(i)=xinc
        endif

      enddo

      chiprev=chinow
      call chisquar
      chinow=chisq
      count=count+1
*-------------------------------------------------------------------
*  Stop search if chi-square hasn't improved in past niter iterations
*-------------------------------------------------------------------
      if (chisq.lt.chibest) then
        count=1
        do i=1,nparm
          abest(i)=a(i)
        enddo
        chibest=chisq
      endif
      if (count.ge.niter) then
        do i=1,nparm
          a(i)=abest(i)
      enddo
        write(*,*) 'Chi-Square has not improved in',niter,
     & ' iterations, search stopped.'
        write(*,*) 'Best values:'
        pr = .true.
```

```
        call chisquar
        return
      endif
*-------------------------------------------------------------
*  Find gradient again if chi-square has increased, otherwise
*  continue to increment parameters along same gradient.
*-------------------------------------------------------------
      if (chinow.gt.chiprev) go to 1500
      go to 1600
      end
*
*=====================================================================
*  Subroutine to find chi-square from parameter list a(nparm).
*  a(1)=fwhmcm, a(2)=rot. temp, a(3)= origin shift
*  a(4)=scoef(1), a(5)=scoef(2), a(6)=rel inten(1),
*  a(7)=rel inten(2), a(8)=baseline.
*=====================================================================
      subroutine chisquar
*
      integer nparm
      real a(30),abest(30)
*
      parameter (evtocm=8065.541)
      real vstk(2,2,100),rstk(2,500000)
      real temp,org
      real reneu(2)
      integer state(2),nvstk(2),rct,ninfil
      integer mpts
      real relfcf(2),scoef(2)
      real fwhmcm,fwhmev
      real smooth(2,100000),espec(2,100000)
      integer nspec
      real chisq
      integer ncomp
      logical pr
*
      common /conv1/ fwhmcm,fwhmev,delta,deltawn
      common /conv2/ mpts
      common /func/ bean,beneu
      common /rvars/ mass1,mass2,rmass,rean,temp,org,baseln
      common /ivars/ rct,ninfil
      common/arrays/reneu(2),state(2),nvstk(2),
     &  relfcf(2),scoef(2)
      common /comp/ chisq,ncomp
      common/opti/nparm
      common/optr/a(30),abest(30)
      common /sticks/ vstk(2,2,100),rstk(2,500000)
      common /spec1/ smooth(2,100000),espec(2,100000)
      common /spec2/ nspec
      common /pr1/ pr,tout
*
      fwhmcm=a(1)
      fwhmev=fwhmcm/evtocm
      temp=a(2)
      org=a(3)
      scoef(1)=a(4)
      scoef(2)=a(5)
```

```
      relfcf(1)=a(6)
      relfcf(2)=a(7)
      baseln=a(8)
      call rsticks(vstk,rstk,NR)
      call conzeke(rstk,rct,espec,nspec,fwhmcm,relfcf(1),baseln,
     &  smooth,NR,NS)
      call cp2(espec,smooth,nspec)
      if (pr) then
        write(*,1405) fwhmcm,temp,scoef(1),relfcf(1),
     &   relfcf(2),org,baseln,chisq
      endif
1405  format (8 g10.4)
      return
      end
*
*=====================================================================
*  Calculate rotational sticks--Note in this subroutine, scoef()
*  is assumed to actually be log(scoef()). And sc() is the real coef.
*  This is different from the rsticks.f program.
*=====================================================================
      subroutine rsticks(vstk,rstk,NR)
*
*  Common variables
*
      parameter(evtocm=8065.541)
      common /func/ bean,beneu
      common /rvars/ mass1,mass2,rmass,rean,temp,org,baseln
      common /ivars/ rct,ninfil
      common/arrays/reneu(2),state(2),nvstk(2),
     &  relfcf(2),scoef(2)
*
      real bean,beneu
      real mass1,mass2,rmass,rean,temp,baseln
      integer rct,ninfil
      real reneu(2),relfcf(2),scoef(2)
      integer state(2),nvstk(2)
*
*---------------------------------------------------------------------
*  Variables local to rsticks subroutine:
*
      real omneu,vorg,vfcf,maxstk,jan,jneu,linstr,sc(2)
      integer i,k,rc(2)
      real vstk(2,2,100),rstk(2,500000),rstk2(2,500000)
*---------------------------------------------------------------------
      do i=1,ninfil
        sc(i)=10**scoef(i)
      enddo
      rmass = mass1*mass2/(mass1+mass2)
      bean = be(rean,rmass)
      rct = 1

      do i = 1, ninfil

        beneu = be(reneu(i),rmass)

        if (state(i).eq.1) then
          omneu = 1.5
```

```
          else
            omneu = 0.5
          endif

          do k = 1, nvstk(i)

            maxstk = -1
*
*           Origin shift parameter 'org' added
*
            vorg = vstk(i,1,k)+org
            vfcf = vstk(i,2,k)
            jan = 0
*           ---------------------------
*           -3/2 Branch
*           ---------------------------
10          jneu = jan - 1.5
            if (jneu.lt.omneu) then
              go to 20
            endif
            rstk(1,rct)=vorg+beneu*jneu*(jneu+1)-bean*jan*(jan+1)
            if (state(i).eq.1) then
              linstr=0.5*(jan-1)*(jan-2)/(2*jan-1)
            else
              linstr=1.5*jan*(jan-1)/(2*jan-1)
            endif
            rstk(2,rct)=vfcf*boltz(jan,temp)*linstr
            rct = rct + 1
*           -------------------------
*           -1/2 Branch
*           -------------------------
20          jneu = jan - 0.5
            if (jneu.lt.omneu) then
              go to 30
            endif
            rstk(1,rct)=vorg+beneu*jneu*(jneu+1)-bean*jan*(jan+1)
            if (state(i).eq.1) then
              linstr=1.5*jan*(jan-1)/(2*jan-1)
            else
              linstr=sc(i)*jan
     &               +0.5*jan*(jan+1)/(2*jan-1)
            endif
            rstk(2,rct)=vfcf*boltz(jan,temp)*linstr
            rct = rct + 1
*           -------------------------
*           +1/2 Branch
*           -------------------------
30          jneu = jan + 0.5
            if (jneu.lt.omneu) then
              go to 40
            endif
            rstk(1,rct)=vorg+beneu*jneu*(jneu+1)-bean*jan*(jan+1)
            if (state(i).eq.1) then
              linstr=1.5*(jan+1)*(jan+2)/(2*jan+3)
            else
              linstr=sc(i)*(jan+1)
     &               +0.5*jan*(jan+1)/(2*jan+3)
```

```
          endif
          rstk(2,rct)=vfcf*boltz(jan,temp)*linstr
          rct = rct + 1
*         -------------------------
*         +3/2 Branch
*         -------------------------
40        jneu = jan + 1.5
          if (jneu.lt.omneu) then
            go to 50
          endif
          rstk(1,rct)=vorg+beneu*jneu*(jneu+1)-bean*jan*(jan+1)
          if (state(i).eq.1) then
            linstr=0.5*(jan+2)*(jan+3)/(2*jan+3)
          else
            linstr=1.5*(jan+1)*(jan+2)/(2*jan+3)
          endif
          rstk(2,rct)=vfcf*boltz(jan,temp)*linstr
          if ((rstk(2,rct)/vfcf).gt.maxstk) then
            maxstk = rstk(2,rct)/vfcf
          endif
          rct = rct + 1

50        jan = jan + 1
          if ((rstk(2,rct-1)/vfcf).lt.(maxstk*0.05)) then
            go to 500
          endif
          go to 10

500     enddo

        rc(i)=rct-1

      enddo

      rct = rct-1
*
*---------------------------------------------------------------
*  Normalize sticks so that largest stick = relfcf(i)
*  Normalize electronic states separately.
*---------------------------------------------------------------
      maxstk = 0.
      do i = 1,rc(1)
        if (rstk(2,i).gt.maxstk) then
          maxstk = rstk(2,i)
        endif
      enddo

      do i = 1,rc(1)
        rstk(2,i) = rstk(2,i)*relfcf(1)/maxstk
      enddo
*
      if (ninfil.gt.1) then
        maxstk = 0.
        do i = rc(1)+1,rc(2)
          if (rstk(2,i).gt.maxstk) then
            maxstk = rstk(2,i)
          endif
```

```
          enddo

          do i = rc(1)+1,rc(2)
            rstk(2,i) = rstk(2,i)*relfcf(2)/maxstk
          enddo
        endif
*
c       write(*,*) rct,' rotational sticks generated'
*
*
*   Sort rsticks into bins
*
        cmmax=0.e0
        cmmin=99999.e0
        do i=1,rct
          if (rstk(1,i).gt.cmmax) cmmax = rstk(1,i)
          if (rstk(1,i).lt.cmmin) cmmin = rstk(1,i)
c          write(*,*) rstk(1,i),cmmax,cmmin
        enddo
        delt = 1.0e0       !* 1 cm-1 bin size
        nbins = int((cmmax-cmmin)/delt)+1
c       write(*,*) cmmax,cmmin,nbins,' cmax, cmin, nbins'
        do i=1,nbins
          rstk2(1,i) = real(i)*delt+cmmin
          rstk2(2,i) = 0.e0
        enddo
        do i=1,rct
          ibin = int((rstk(1,i)-cmmin)/delt)+1
          rstk2(2,ibin) = rstk2(2,ibin)+rstk(2,i)
        enddo
        rct = nbins
        do i=1,rct
          rstk(1,i)=rstk2(1,i)
          rstk(2,i)=rstk2(2,i)
        enddo
*
c       write(*,*) rct,' bins'

        return
        end

*=====================================================================
*   Function to calculate rotational constant, from Re in angstroms,
*   reduced mass in amu.
*=====================================================================
        real function be(re,rmass)

        real mi,re,rmass
*
*   Calc. moment of inertia in kg m^2
*
        mi =  1.66054e-1 * rmass * (re**2)
        be = 2.79928 / mi
        end
*
*=====================================================================
*   Function to calulate anion Boltzmann distribution.
```

```
*  Note that line strength already includes degeneracy.
*=====================================================================
      real function boltz(jan,temp)
*
*  Convert temp in K to cm-1
*
      common /func/ bean,beneu
      real jan,temp,tempk,er
      real bean,beneu

      tempk = 0.69504*temp

      er = bean*jan*(jan+1)
      boltz=2.71828**(-er/tempk)
      return
      end


*=====================================================================
      subroutine conzeke(rstk,nsticks,espec,nexp,fwhm,relfcf,bline,
     &  ospec)
*=====================================================================
*
*  Convolutes stick spectrum with ZEKE lineshape from fit to bromine
*  atomic data, form
*     y=(a*x+b*x^3)/(1+c*x^2+d*x^4)
*  Sticks are assumed to already be sorted from low to high cm-1.
*
*  Input:   rstk()      rotational sticks
*           nsticks           number of rotational sticks
*           espec()     experimental spectrum
*           nexp        number of points in experimental spectrum
*           fwhm        FWHM (cm-1) of ZEKE lineshape
*           relfcf      intensity of largest peak (usu.=1)
*           bline baseline as fraction of relfcf
*
*  Output:  ospec()     output (convoluted) spectrum
*                       (same number of pts as espec)
*
*
      implicit undefined(a-z)
*
      integer nsticks,nexp
      real fwhm,bline

      integer i,j
      real wn,cmax,con,a,b,c,d,cc,x1,
     &  wn2,x2,chisq,relfcf,rstk(2,500000),espec(2,100000),
     &  ospec(2,nexp)

      parameter(a=1.3342091765e0,
     &          b=0.0059500947e0,
     &          c=0.40742214e0,
     &          d=0.022329373e0)
*
      x1 = 0.43e0*fwhm
      cc = 3.2e0/fwhm
```

```
      cmax = 0.
*
      chisq = 0.
      do i=1,nexp              !* Loop over exp points
        wn = espec(1,i)
        wn2 = wn + x1
        ospec(2,i) = 0.0e0
        ospec(1,i) = wn
        do j=1,nsticks              !* Loop over sticks
          if (rstk(1,j).lt.wn2) then
            x2 = (wn2 - rstk(1,j))
            con = rstk(2,j)*(a*(x2*cc)+b*(x2*cc)**3)/
     &                    (1.0d0+c*(cc*x2)**2+d*(cc*x2)**4)
          ospec(2,i) = ospec(2,i) + con
          endif
        enddo
        if (ospec(2,i).gt.cmax) then
          cmax = ospec(2,i)
        endif
      enddo

      do i=1,nexp              !* Normalize
        ospec(2,i)=relfcf*((ospec(2,i)/cmax)*(1.0d0-bline)+bline)
      enddo

      return

      end


*=====================================================================
* Compare experimental and simulated spectra, assuming they have the
* Same number of points and same x-components--no interpolation
* a() contains exp. spectrum.  ncounts is number of electron counts.
*=====================================================================
      subroutine cp2(a,b,n)
      integer n
      real a(2,n),b(2,n)
      real chisq
      integer i,ncomp
      real ncounts

      common /comp/ chisq,ncomp
      common /comp2/ ncounts

      ncomp=n
      chisq=0
      asigma=(sqrt(ncounts))/ncounts

      do i=1,ncomp
        chisq=chisq+((a(2,i)-b(2,i))/asigma)**2
      enddo
      end
```

B4. References for Appendix B

[1] J. Xie and R. N. Zare, J. Chem. Phys. **93**, 3033 (1990).

[2] J. Xie and R. N. Zare, J. Chem. Phys. **97**, 2891 (1992).

[3] A. D. Buckingham, B. J. Orr, and J. M. Sichel, Philos. Trans. R. Soc. London, Ser. A **268**, 147 (1970).

[4] R. N. Zare, *Angular Momentum* (Wiley, New York, 1988).

[5] P. R. Bevington, *Data Reduction and Error Analysis for the Physical Sciences* (McGraw-Hill, New York, 1969).

Appendix C. Simulated Annealing

C1. Introduction

One problem we face in trying to understand the energetics of a cluster containing many atoms or molecules is determining the global minimum equilibrium geometry. Especially in the case of the weakly bound clusters considered in this work, the number of possible isomers increases dramatically as one adds more atoms. For example a cluster of 13 identical Lennard-Jones atoms has 988 stable, distinct geometric isomers.[1] It is also known that for large clusters the number of local minima increases exponentially with the number of atoms.[2] Often there are many local minima located close in energy to the global minimum energy geometry, and this makes the problem of finding the global minimum non-trivial.

C2. Brief survey of methods of finding global minima of clusters

If one had an arbitrarily fast computer, the problem of finding the global minimum would simple. One could simply calculate the cluster potential at all points on as small a grid as required to locate the minimum. In this "brute force" approach, it would be necessary to calculate the potential points on a $3N$-6 dimensional grid (N being the number of atoms in a polyatomic cluster). If a 100 point grid were used, this would amount to 10^6, 10^{12} and 10^{18} calculations of the potential for 3, 4 and 5 atom clusters, respectively. Hence, we can see that with currently available computers, such a "brute force" calculation is impractical for clusters with more than 3 or 4 atoms. In the case of some small clusters, this approach could be speeded up by making assumptions about the

symmetry of the equilibrium geometry. However this is not a generally used technique for large clusters and will not be considered further.

Many algorithms are available for finding the minimum of a mathematical function when initial values of the independent variables are assumed.[3],[4] These include gradient minimization,[4] the simplex method,[3] and conjugate gradient methods.[3] If the internuclear coordinates of the cluster are taken to be the independent variables and the potential energy the function, then any of these methods may be used to efficiently find a minimum on the potential energy surface. The problem with all of these techniques when used by themselves, however, is that they cannot distinguish between local and global minima. For a given initial configuration of atoms, the minimization algorithm may become trapped in a local minimum.

One approach that may be used to try to locate the global minimum energy structure is to apply the minimization algorithm repeatedly, with many different--possibly randomly chosen--starting geometries, and with luck and persistence the global minimum may be stumbled upon in this way. This is a viable approach, but it suffers from the drawback that it is hard to determine if one has found the true global minimum. Also it is not a very systematic approach, relying as it does on random or arbitrarily chosen points on the potential energy surface.

Another quite elegant method for finding the global optimum of a system was first developed and applied to problems in computer design by Kirkpartick *et al.*[5] This is the method of simulated annealing, which was based on the Metropolis Monte-Carlo algorithm[6] originally developed as a general-purpose tool for characterizing the

properties of large collections of interacting molecules at a given temperature, *i.e.*, a way of simulating a canonical ensemble of particles at equilibrium.

The original Metropolis Monte Carlo method[6] may be outlined as follows:

(1) An arbitrary initial configuration of the system is chosen.

(2) A random displacement is made in the system.

(3) The change in the energy, ΔE, of the system is calculated.

(4) If $\Delta E<0$, the move is accepted, the system is reset with the new configuration, and the simulation continues again with step 2.

(5) If $\Delta E>0$, the move is allowed with a probability of $\exp(-\Delta E/kT)$, where k is the Boltzmann constant and T is the temperature. In practice, this is implemented by choosing a random number, ξ, between 0 and 1, and allowing the move if $\xi<\exp(-\Delta E/kT)$.

Metropolis *et al.* showed that this if one averages any property of interest over each step in the simulation, one may calculate accurate values of thermodynamic properties, the accuracy being limited only by the duration of the simulation (and of course by the accuracy of the model interaction potential between the particles).

To illustrate how the Metropolis algorithm is applied to global optimization, we begin by following the discussion in Press *et al.*[3] of the simulated annealing method applied to the "travelling salesperson" problem. In this problem, a salesperson must travel to a large number of cities, visiting no city more than once, and the cost of traveling between the cities is proportional to the distance between the cities. The problem then consists of finding the order of cities that minimizes the cost. This is a

problem in discrete combinatorial mathematics. According to Kirkpatrick *et al.*[5] there is no known algorithm for finding an *exact* solution to this problem for which computation time does not scale exponentially with the number of cities visited. This is consistent with our above discussion of the scaling of the 'brute force' approach for finding the global minimum energy of a cluster. However, according to Press *et al.*, "simulated annealing has effectively 'solved' the famous traveling salesman problem..."[3] The difficulty in this problem lies in the fact that there are many possible near-solutions--solutions close in cost to the optimal solution, as is the case in the cluster problem. The simulated annealing algorithm for a discrete combinatorial problem such as this is:

(1) Choose an initial ordering (permutation) of the set of cities.

(2) Calculate the "cost function" or "objective function," which we will also call E, which in this case is the sum of the distances between the cities taken in order.

(3) Rearrange the ordering of the cities.

(4) As in the Metropolis algorithm, evaluate E for the new permutation and keep the move if E is reduced, or, if E is increased, keep the move with a probability proportional to $\exp(-E/kT)$. In this specific example, of course, k no longer has the same value as the Boltzmann constant, but may be considered a scaling factor between the cost function E and an effective "temperature," T.

(5) Run the Metropolis algorithm for a large number of iterations, to allow the system to "equilibrate."

(6) Gradually reduce the effective temperature, T, according to an "annealing schedule," until the system becomes "frozen," *i.e.*, there is no further change in the arrangement of order of visiting the cities.

Press *et al.*[3] provide a computer program to explicitly treat the traveling salesperson problem by simulated annealing, as well as an excellent introduction to the concepts of simulated annealing and other optimization methods.

The beauty of the simulated annealing algorithm as shown in this example is that, unlike the direct minimization procedures mentioned above the algorithm is much less likely to become trapped in local minima. The reason for this is that the system of interest is allowed to explore a portion of phase space at each stage in the annealing schedule, and moreover, "uphill" moves are allowed, according to the Boltzmann distribution. Thus, even if the system finds itself within a local optimum during the procedure, it has a chance to "escape" to find the global optimum.

Kirkpatrick *et al.* confined their discussion to discrete systems, and simulated annealing has since been used extensively in VLSI design and for other discrete combinatorial problems. The implementation of a simulated annealing algorithm becomes more complicated in the case of a problem involving continuous variables. A large part of the problem lies in the optimal choice of step sizes and directions. This problem has been discussed at length by Vanderbilt and Louie.[7]

The problem of finding the global minimum energy of a cluster is, of course, a problem involving continuous variables. Monte Carlo approaches have been successfully used to such problems; see, for example, Ref. 8. Another approach is also used for physical systems, in which the molecular dynamics (MD) methods are used to simulate the actual motions of the atoms and molecules in the system. Various methods are used to adjust the system in order to produce a distribution kinetic energies corresponding to a

certain temperature. These methods are described in more detail in the following sections. Then, as in Monte Carlo methods, the temperature is reduced according to an annealing schedule, until the system freezes at the global minimum configuration.

The molecular dynamics approach to simulated annealing has the advantage of intuitive simplicity, and a closer resemblance to the actual behavior of a physical system. Thus it is the method of choice if one wishes to study such things as the dynamics of cluster formation. However, if one wishes to simply find equilibrium geometries, the molecular dynamics method would seem to be more computationally time-consuming than the Monte-Carlo approach. This is because to perform a MD simulation one must calculate forces as well as potentials in order to solve the equations of motion. The MD approach was used for the current studies because easily-adapted pre-existing MD programs were available, not because of considerations of computational efficiency. To our knowledge no systematic study has been made of the relative computational efficiency of the Monte Carlo and MD simulated annealing methods.

C3. Molecular dynamics simulated annealing

In this section we describe our approach to simulated annealing, which is based on a molecular dynamics program obtained from Prof. C. C. Martens (UC Irvine). First we review the basics of constant-energy (microcanonical) molecular dynamics simulation methods. We then discuss how to implement constant-temperature molecular dynamics (to simulate a canonical ensemble). Finally we discuss a modification of the latter method for simulated annealing.

C3.1 Microcanonical molecular dynamics

The basic problem that we will deal with here is how to simulate the motions of atoms which interact according to potential energy functions. Our treatment is drawn mainly from the book *Computer Simulation of Liquids*, by M. P. Allen and D. J. Tildesley,[9] which discusses these issues in much more depth, and is an essential resource for anyone interested in molecular dynamics simulations. We assume that the Born-Oppenheimer approximation is valid, and treat the atoms as points interacting according to classical mechanics. The problem then becomes how to solve the equations of motion of a set of i atoms with masses m_i and momenta $\mathbf{p}_i$ at positions $\mathbf{r}_i$ under the influence of the potential $V(\mathbf{r}_i)$ due to interactions with the other atoms. The equations of motion can then be represented by:[9]

$$\frac{d}{dt}\mathbf{r}_i = \frac{\mathbf{p}_i}{m_i} \tag{C1}$$

$$\frac{d}{dt}\mathbf{p}_i = -\nabla V(\mathbf{r}_i) = \mathbf{f}_i \tag{C2}$$

where $\mathbf{f}_i$ is the force on atom i.

Of the many algorithms available for solving the equations of motion, the most commonly used methods in molecular dynamics computations are the Gear predictor-corrector algorithm, which solves the set of first order differential equations given above, and the Verlet algorithm, which treats the problem in terms of a single set of second order differential equations. Both of these are finite difference methods, *i.e.* methods which start from an initial configuration and propagate the equations forward in time. For a discussion of the relative merits of these and other methods, see Allen and Tildesley.[9] Well-documented computer code to implement these algorithms as described

in Allen and Tildesley's book may be obtained from the website: http://toucan.phy.bris.ac.uk/AllenTildesley/home.html. The program used in this work employs the Gear predictor-corrector algorithm. Hence we confine our discussion to a brief description of this method.

As the name implies, the Gear algorithm takes place in two steps, a prediction step, and a correction step. In the first step, the configuration of the atoms are estimated from the previous configuration by a truncated Taylor expansion around the time, t. The program used in this work is a six-value Gear algorithm provided by the group of Prof. C. C. Martens.[10] We show here the first step of this algorithm, using the notation of Allen and Tildesley:

$$\begin{aligned}
\mathbf{r}^p(t+\delta t) &= \mathbf{r}(t)+\delta t\mathbf{v}(t)+\tfrac{1}{2}\delta t^2\mathbf{a}(t)+\tfrac{1}{6}\delta t^3\mathbf{b}(t)+\tfrac{1}{24}\delta t^4\mathbf{c}(t)\tfrac{1}{120}\delta t^5\mathbf{d}(t)\\
\mathbf{v}^p(t+\delta t) &= \mathbf{v}(t)+\delta t\mathbf{a}(t)+\tfrac{1}{2}\delta t^2\mathbf{b}(t)+\tfrac{1}{6}\delta t^3\mathbf{c}(t)+\tfrac{1}{24}\delta t^4\mathbf{d}(t)\\
\mathbf{a}^p(t+\delta t) &= \mathbf{a}(t)+\delta t\mathbf{b}(t)+\tfrac{1}{2}\delta t^2\mathbf{c}(t)+\tfrac{1}{6}\delta t^3\mathbf{d}(t)\\
\mathbf{b}^p(t+\delta t) &= \mathbf{b}(t)+\delta t\mathbf{c}(t)+\tfrac{1}{2}\delta t^2\mathbf{d}(t)\\
\mathbf{c}^p(t+\delta t) &= \mathbf{c}(t)+\delta t\mathbf{d}(t)\\
\mathbf{d}^p(t+\delta t) &= \mathbf{d}(t)
\end{aligned} \qquad \text{(A3)}$$

Here, **r**, **v**, and **a** stand for the set of positions, velocities and accelerations of all of the atoms, **b** and **c** stand for the first and second time derivatives, respectively, of the acceleration, and δt stands for the time step. The superscript p indicates that these are predicted values.

The equations of motion are introduced in the second, "corrector," step. In this step the forces are calculated from Eqs. (C1) and (C2) from the predicted positions **r** at time $t+\delta t$. Then the "correct" accelerations $\mathbf{a}^c$ are calculated from Equation (C3). The error in the predicted accelerations are then estimated from

$$\Delta\mathbf{a}(t+\delta t) = \mathbf{a}^c(t+\delta t) - \mathbf{a}^p(t+\delta t). \tag{C4}$$

Finally, the initially predicted values are corrected using this estimated error in the acceleration:

$$\begin{aligned}
\mathbf{r}^c(t+\delta t) &= \mathbf{r}^p(t+\delta t) + c_0\Delta\mathbf{a}(t+\delta t) \\
\mathbf{v}^c(t+\delta t) &= \mathbf{v}^p(t+\delta t) + c_1\Delta\mathbf{a}(t+\delta t) \\
\mathbf{a}^c(t+\delta t) &= \mathbf{a}^p(t+\delta t) + c_2\Delta\mathbf{a}(t+\delta t) \\
\mathbf{b}^c(t+\delta t) &= \mathbf{b}^p(t+\delta t) + c_3\Delta\mathbf{a}(t+\delta t) \\
\mathbf{c}^c(t+\delta t) &= \mathbf{c}^p(t+\delta t) + c_4\Delta\mathbf{a}(t+\delta t) \\
\mathbf{d}^c(t+\delta t) &= \mathbf{d}^p(t+\delta t) + c_5\Delta\mathbf{a}(t+\delta t)
\end{aligned} \tag{C5}$$

The coefficients $c_0,\ldots,c_4$ are chosen to optimize the stability and accuracy of the algorithm, and suggested coefficients given by Gear.

C3.2. Constant temperature molecular dynamics

The method in the above section is used to simulate atoms in the microcanonical ensemble, *i.e.*, with total energy conserved. The temperature of a cluster simulated in this way is not a well defined quantity, but follows a distribution, as the kinetic energy is not constant during the simulation. One method that has been used to simulate a canonical ensemble of atoms is to periodically rescale the velocities of all the particles by a factor:

$$\chi = \sqrt{\frac{T_{targ}}{T_{avg}}} \tag{C6}$$

where T_{targ} is the desired temperature of the system and T_{avg} is the average kinetic temperature of the system calculated from:

$$T_{avg} = \frac{2}{3Nk}\langle KE\rangle. \tag{C7}$$

where N is the total number of atoms, k is Boltzmann's constant, and $\langle KE \rangle$ is the average kinetic energy from the preceding time interval. This and other methods of constant temperature MD are described by Allen and Tildesley.[9].

C3.3. Using the rescaling of velocities for simulated annealing

Our implementation of simulated annealing using molecular dynamics was suggested by modification of the above constant-temperature MD algorithms described above in which the velocities are rescaled at each time step by a factor involving a time constant that governs the relaxation rate towards the desired "constant" temperature.[9] This suggested a method of setting the annealing schedule where the average kinetic energy of the cluster is decreased by rescaling the velocities of the atoms by the factor

$$\chi = \sqrt{1 + \frac{\Delta t}{t_{const}} \left(\frac{T_{targ}}{T_{avg}} - 1 \right)}, \qquad \text{(C8)}$$

where t_{const} is the time constant, and Δt is an interval of a number of time steps--in our application about 100-1000 time steps were used for Δt. The target temperature is then set to a number very small number, and the cluster is allowed to relax to this temperature. Whether or not this type of annealing schedule is optimal is open to debate, as many other schedules are possible.

C4. Documentation of the simulated annealing program "amain.f"

In this section we discuss in detail our implementation of the simulated annealing problem as applied to the Rg_nX systems, including examples of how to run the program,

as well as a discussion of the construction of the program for those who wish to modify it.

C4.1 The input file

In order to run the simulated annealing program one must supply a parameter file containing the pair potential parameters, flags to indicate which many body forces to include, and so on. A sample parameter file is given below, followed by a discussion of each of its components. This example is for the Ar_6I^- anion without any many-body forces included, titled "in1" :

```
1:  7.28       !* argon mass (me*10000)
2:  23.13      !* iodine atom mass (me*10000)
3:  0.03794    !* ar-ar epsilon for Lennard-Jones
4:  0.05935    !* ar-I epsilon for LJ
5:  64.34      !* ar-ar sigma for LJ
6:  68.35      !* ar-I sigma for LJ
7:  10000      !* nstep used in thermgen
8:  250        !* nskip used in thermgen
9:  250        !* ntconst (time constant for thermgen, in steps of h)
10:  1000      !* nave (number of steps for averaging run)
11:  2.        !* time step (au*100)
12:  0.00000001    !* desired final temp. in kelvin used in thermgen
13:  526878        !* dseed (seed for random number generator)
14:  0.000  !* escale (Initial energy in eV)
15:  7.     !* boxsize (Ang.)
16:  3.5    !* cutoff (Ang.)
17:  6      !* ncluster (number of rare gas atoms)
18:  2      !* nrgpot (Ar-Ar poten: 1 for LJ, 2 for Aziz)
19:  1   !* nhalpot (1 anion, 2 neut, 3 neut central diff, 4 anal neut)
20:  1            !* neigval (1, 2 for X; 3,4 for I; 5,6 for II)
21:  1.0d-5 !* delta (for numerical derivatives, Ang)
22:  0.01   !* gstepin (initial step for gradient minimization, Ang)
23:  1.0d-11   !* gsmall (convergence criterion for gradient min, eV)
24:  0      !* indflag (3 body induction, old model 1 on, 0 off)
25:  0      !* iexqflag(Charge-exchange quadrupole, Jansen model, 1 on,
            0 off)
26:  0      !* indi (iterated induction--dipoles only, 1 on, 0 off)
27:  0      !* iexqd (distributed dipole exchange quadrupole, 1 on, 0
            off)
28:  0      !* indq (charge induced Rg quadrupoles--not iterated,
            0=off, 1=on)
29:  0      !* iexg (Charge-Gaussian exchange, 0=off, 1=on)
30:  0      !* indqi(iterated dipoles & quadrupoles, 0=off, 1=on)
31:  0      !* iaxtel (Axilrod-Teller, 0=off, 1=on)
32:  2      !* ninit (1 pos. from file, 2 rand. pos save to file, 3 to
```

```
            restart, 4 to restart with average, 5 pos from file avg
            only,6 grad min)
33:  1.6419 !* polind (RG polarizability Ang^3 for 3-body ind)
34:  11.08  !* polrg (RG polarizability, a0^3, for iterated induction)
35:  52.7   !* polx (halide polarizability, a0^3, for iterated
            induction)
36:  27.11  !* pqrg (RG quadrupole polarizability,a0^5,= C Buckingham
            defn.)
37:  254.   !* pqx (halide quadrupole polarizability, a0^5) 254.
38:  0.936  !* betaexq (Exchange quadrupole range parameter, Ang^-1)
39:  6.5    !* cutexq (exchange quad cutoff distance, Ang)
40:  2086.  !* theta6 (Rg quadrupole dispersion coeff, e*a0^8)
41:  179.   !* c9anion (eV*Ang^9)
42:  130.   !* c9neut (eV*Ang^9)
43:  0.94268      !* soconst (eV)
44:  0.0458, 4.07, 5.70, 4.45, 1.08, 1.62, 279.5, 3537.  !* Anion MMSV
45:  0.0188, 3.95, 7.15, 6.18, 1.01, 1.62, 5234., 38032. !* X1/2 MMSV
46:  0.0139, 4.18, 7.25, 6.30, 1.04, 1.62, 7079., 51439. !* I3/2 MMSV
47:  0.0160, 4.11, 6.90, 6.40, 1.04, 1.64, 6150., 44688. !* II1/2 MMSV
48:  0.0123422,3.7565,10.77874743,1.8122004,2.26210716e5,1.10785136,
      0.56072459,0.34602794,1.36    !* Argon HFD-B parameters
49:  init_config_6
50:  output_file1_6
51:  restart_file1_6
52:  average_file1_6
53:  poten_file1_6
```

Note that the line numbers in boldface are not included in the actual file, but are shown here for reference purposes. All 53 lines of the file must be present, regardless of whether an anion or neutral simulation is being performed, or which many-body potential options are in use. The comments after the "!*" symbols are ignored by the program; in some cases the variable name used in the program is given in this comment field, as well as a brief explanation of its meaning. Note that comments are not allowed after lines containing file names (lines **49-53**), and will cause the program to crash.

Lines **1** and **2** are the masses of the rare gas and halide, respectively, in units of $10000m_e$, where m_e is the atomic unit of mass (*i.e.*, the electron mass). Lines **3-4** are Lennard Jones parameters, which were not used in the present application. One does have the option to use the Lennard Jones parameters for the Ar-Ar potential, however one may not use the many-body forces with this option.

Lines **7-14** provide information about the timescale of the simulation and the annealing schedule. The basic time step of the simulation is set by Line **8**, in $100\tau_0$ (where τ_0 is the atomic unit of time, equal to approximately 0.024 fs). Thus the value of 2 in the sample input file corresponds to a time step of about 5 fs. Line **7**, `nstep` is the total number of MD steps in the simulation. Line **8**, `nskip` is the number of MD time steps between energy rescalings. Line **9**, `ntconst`, is the time constant, in units of number of MD steps, that determines the factor by which the energies are `rescaled` after each `nskip` number of time steps, by the equation (See Chapter 4, Section 4.4.2):

$$fac = \sqrt{1 + \frac{\text{nskip}}{\text{ntconst}}\left(\frac{T_{targ}}{T_{avg}} - 1\right)} \qquad \text{(C9)}$$

Typically, when starting with a random configuration of atoms, we set `ntconst` equal to `nskip`, so that the rescaling factor is close to 0 (because for SA we normally set $KE_{targ} << KE_{avg}$) and energy is removed as fast as possible from the cluster. When the simulated annealing run, the cluster is allowed to run, and the average positions of the atoms are determined during this "averaging run." Line **10**, `nave`, indicates the number of time steps to perform this averaging. Line **12** is sets the desired final temperature for the simulation, which is used to directly calculate KE_{targ} in the above equation. When the program is used for simulated annealing this line should be set as arbitrarily close to 0 as possible. In other applications, *i.e.*, MD simulation of a system at a non-zero temperatures, this could be set to higher temperatures, or one could modify the program so that this variable could be used to set the annealing schedule. Line **13**, `dseed`, is the seed for the random number generator, and may be any integer. *Note that the sequence of "random" numbers produced by the program is entirely determined by the initial choice*

of seed. Thus if one does not change `dseed` *one will produce identical initial configurations of atoms, and if all the other parameters for the anneal are the same, one will arrive at identical final configurations. Therefore it is essential to change* `dseed` *every time one wishes to start with a "fresh" initial configuration.* Line **14**, `escale`, is the initial total kinetic energy, in eV, given to the atoms. This energy is distributed randomly among all the atoms. In general, for the first step when the atom positions are randomly chosen with a box, `escale` is set to 0, so that the initial kinetic energy of the atoms is determined only by the potential of the other atoms. If `escale` is set to a larger value, the atoms tend to evaporate out of the box.

Lines **15-17** determine the initial placement of the atoms in a box when the flag `ninit` (line **32**) is set to 2. When this flag is set, the atoms are placed randomly within a cubic box with the side dimension `boxsize` (line **15**) given in Angstroms. The initial configuration of the atoms is not allowed to contain any pairs closer together than `cutoff` (line **16**). The value shown, 3.5 Å, was found to be useful so that the initial configuration does not contain atoms in highly repulsive regions of the potential surface, which would lead to rapid evaporation. Line **17** gives the number of *rare gas* atoms (the number of *halide* atoms is always assumed to be one). This initial configuration is saved to the file named in line **49**, here called "init_config_6." The format of this file is explained in detail below.

Lines **18-21** are integer inputs that tell the program what kinds of pair potentials to use. Line **18** `nrgpot` may be set to 1 to use the Lennard-Jones form for the Rg-Rg interaction, or to 2 to use the more accurate Hartree-Fock Dispersion (HFD-B2) form.[11] The parameters for the Ar-Ar HFD-B2 potential are given in line **48**, listed in the same

order as in Ref. [11]. In all of the simulations in this work, the HFD-B2 form was used for the Ar-Ar pair potential. Line **19**, `nhalpot`, indicates whether to calculate the anion or neutral potential. If `nhalpot` is set to 1, the anion Rg_nX^- potential is used, with the anion MMSV parameters given in line **44** and many body terms as described below. If `nhalpot` is set to 2, 3 or 4, the neutral Rg_nX potential is calculated according to the method outlined in Chapter 4. If `nhalpot` is set to 2 or 3, the neutral eigenvalues are calculated by numerical diagonalization of the 6x6 matrix Eqn. (4.16) of Chapter 4. Line **21** specifies the displacement, `delta`, in Å, to be used for calculation of the forces from the potentials. For `nhalpot` equal to 2, the forces are calculated by finite difference, whereas if `nhalpot`=3, the forces are calculated by central difference, which in principle should give more accurate values of the forces.[3] However the calculation runs somewhat slower with `nhalpot`=3, and the precise values of the forces are not extremely significant for simulated annealing applications, so it is not recommended to run in the mode. *When* `nhalpot` *is set to 4, the neutral eigenvalues are found analytically, resulting in an approximate tenfold increase in speed. Therefore, this is the recommended mode of operation for calculation of neutral potentials.* The numerical mode of calculation may be used as a check on the analytical calculation.

Line **32**, `ninit`, specifies the mode of operation of the program. These modes are explained below. When ninit is equal to:

1. The initial positions and momenta of the atoms are read in from the file named in line **49**, here called "init_config_6," the final configuration is save in the file named in line **51**, here "restart_file1_6" the configuration after the annealing run

is save in the file named in line **52**, here "average_file1_6," and the final total potential is saved in the file on line **53**, here named "poten_file1_6;"

2. The atoms are placed at random in a box, and if `escale` is non-zero, are given initial kinetic energies totaling `escale`. This initial configuration is save in the file named in line **49**, and the output files are saved as described above for `ninit=1`;
3. The program is run with the initial configuration taken from the file named in line **51** ("restart_file1_6"), and the final configuration is saved to this same file; the "average" and "poten" files are also updated after running in this mode, but the "init_config" file is not modified; this mode of operation is used if one wishes to "restart" the simulation from the point where the previous simulation ended.;
4. The program is run as if `ninit=3`, above, except that the initial configuration is read in from line **52**, here "average_file1_6;" this mode is useful for rapid cooling of a cluster when it is near a minimum configuration;
5. In this mode, the initial positions and momenta are read in from the file named in line **49** ("init_config_6"), and only the averaging run is performed, the results of which are saved in "average_file1_6" (line **52**); the simulated annealing procedure is not performed in this mode; this mode is useful to get an idea of the standard deviations in atomic positions and potential energy of a given configuration;
6. In this mode, the initial positions are read in from the file specified in line **49**, and a gradient minimization procedure is performed; the final potential is saved in the

file named in line **53**, but the final configuration is not saved. This mode is used to quickly optimize the potential when one is very near the minimum.

In all modes of operation, an output file, named in line **50** is produced, which contains information about the cluster geometry, potential, etc., and is described in detail below.

Lines **24-31** are flags which turn on or off various many-body interactions. Except for the Axilrod-Teller interaction (line **31**), these flags are only applicable to the anionic clusters and are ignored by the program when a neutral cluster is being simulated.

Lines **24, 26, 28** and **30** specify various modes of implementation of the induction effects described in Chapter 4. When line **30**, `indqi`, is set to 1, the full iterative procedure including induced dipoles and quadrupoles is performed for finding the non-additive energy. This mode was used to obtain the results presented in Chapter 4. The other flags may be set to implement simpler (and faster running) models of the non-additive induction energy. If line **26**, `indi`, is set to 1, the iterated induction procedure is performed neglecting the quadrupole moments of the atoms. The inclusion of quadrupole induction effects is an extension of the method as originally introduced[12] and as usually implemented for simulations of polar solvents, as for example by Jedlovzky *et al.*,[13] which normally stop at the dipole term. However, it seems that the quadrupole non-additive portion contributes significantly to the non-additive energy in the case of highly polarizable anions and moderately polarizable rare gases such as Ar. It is probable that simulations involving highly polarizable rare gases, such as those of the Xe_nI^- clusters that are currently underway in the Neumark group, the quadrupole induction effect should be quite significant.

When `indflag` is set (line **24**) a simpler model of the many body interaction is used. In this case, we only consider the dipoles induced in the rare gases by the halide charge, and the pair interactions between these induced dipoles. Then for an Rg_nX^- cluster, the non-additive induction energy is approximated by

$$V_{ind} = \sum_{i<j} \frac{1}{R_{ij}^3} \left[\mu_i \cdot \mu_j - \frac{3(\mu_i \cdot \mathbf{R}_{ij})(\mu_i \cdot \mathbf{R}_{ij})}{R_{ij}^2} \right] \qquad \text{(C10)}$$

where $\mathbf{R}_{ij}$ is the vector *from* rare gas atom *i* *to* rare gas atom *j*, μ_i is the dipole moment induced by the halide charge on rare gas atom *i*, and the sum runs over all pairs of rare gas atoms. The induced dipole moments are found from

$$\mu_i = \frac{q\alpha_i}{R_i^3} \mathbf{R}_i \qquad \text{(C11)}$$

where q is the halide charge, α_i is the rare gas dipole polarizability, and $\mathbf{R}_i$ is the vector *from* the rare gas *to* the halide. This induction model may be considered a first order approximation to the full iterative solution of Eqn. (4.23) of Chapter 4. For an example of the application of this model with simulated annealing see the work of Asher *et al.*[14] on metal-argon clusters.

When `indq`, line **28**, is set to 1 the non-additive energy due to induced quadrupole moments is included in a non-iterative manner similar to the dipole induction model described above. When `indq` is set, the quadrupole moments induced in each rare gas atom by the halide charge are calculated, and the pair interactions of these quadrupoles are calculated, as well as the interactions of the quadrupoles with the dipoles calculated iteratively. For the formulas for the quadrupole-quadrupole and dipole-dipole interactions see Buckingham.[15] *In order for this to function properly,* `indi` *(line 26)*

must also be set to 1. *If* `indi` *is not set, the dipole-quadrupole interactions will not be included and the overall effect of the quadrupole will be severely underestimated.*

The flags **25**, **27** and **29** control which of the three models described in Chapter 4 for the "exchange quadrupole" effect is used. If `iexqflag` (line **25**) is set to 1, the non-additive exchange quadrupole energy is calculated from the interaction of the halide charge with the cylindrically symmetric quadrupole moments located at the midpoints between the nuclei of two rare gas atoms, $\Theta_{ex}^{(ij)}$, calculated from Equation 32 of Chapter 4. The total charge-exchange quadrupole energy for the cluster is then given by:

$$V_{ec}(\text{point quadrupole}) = \frac{q\Theta_{ex}^{(ij)}}{R_{c0}^3}\left(\frac{3}{2}\cos^2\phi - \frac{1}{2}\right) \qquad \text{(C12)}$$

where $\mathbf{R}_{c0}$ is the vector from the Rg-Rg midpoint to the halide nucleus, ϕ is the angle between the quadrupole and $\mathbf{R}_{c0}$. When line **27** (`iexqd`) is set to 1, the distributed dipole model of the exchange quadrupole introduced by Hutson *et al.*[16] is used. When line **29** (`iexg`), the "Gaussian exchange-charge," model developed in Chapter 4 is employed, and the non-additive energy is found from Equation 34 of Chapter 4.

Line **31**, `iaxtel`, is set to 1 to turn on the Axilrod-Teller three body dispersion interaction [see Eqn. (4.18) of Chapter 4]. This interaction may be used for both the anion or neutral calculation.

Lines **33-43** of the input file contain atomic data and interaction constants used in the calculation of the many-body potentials. These are listed below in the same notation as Chapter 4:

Line 33 `polind` α(Rg) Dipole polarizability of the rare gas atom in Å^3

Line 34	`polrg`	α(Rg)	Dipole polarizability of the rare gas atom in a_o^3
Line 35	`polx`	$\alpha(X^-)$	Dipole polarizability of the halide anion in a_o^3
Line 36	`pqrg`	C(Rg)*	Quadrupole polarizability of the rare gas in a_0^5.
Line 37	`pqx`	$C(X^-)$*	Quadrupole polarizability of the halide anion in a_0^5.

Line 38 `betaexq` β Gaussian range parameter in $Å^{-1}$ for exchange quadrupole

Line 39 `cutexq` A distance in Å which is used as the cutoff point for the exchange-charge calculation, to prevent non-physical behavior of Equation 34 of Chapter 4.

Line 40 `theta6` C_Θ Coefficient for calculating the dispersion induced quadrupole moment on a rare gas atom [see Eqn. (4.36), Chapter 4]

Line 41 `c9anion` $C_9(X^-\text{-Rg-Rg})$ Axilrod-Teller coefficient for halide anion-rare gas three-body dispersion interaction in $eV \cdot Å^9$.

Line 42 `c9neut` C_9(X-Rg-Rg) Axilrod-Teller coefficient for halogen neutral-rare gas three-body dispersion interaction in $eV \cdot Å^9$.

Line 43 `soconst` Δ Halogen spin-orbit constant in eV.

---

* Note that the definition of the quadrupole polarizability here is that of Buckingham,[15] which differs from other definitions (*e.g.* this is *not* the same α_q used in Chapter 3: see the footnote to Table VI of Chapter 4).

The MMSV pair potential parameters for the RgX^- anion are given in line **44**, and the MMSV parameters for the neutral $X1/2$, $I3/2$, and $II1/2$ states are given in lines **45, 46** and **47** respectively. The order and units of the MMSV parameters in each line the same as those in the input file for the Morse DVR program described in Appendix A (See Section A5.1).

C4.2 The atomic configuration files

This section describes the format of the files which describe the configurations of atoms, which are named in lines **49, 51** and **52** of the input file. A configuration of N rare gas atoms and one halogen atom would be represented in the following format:

1	X_1	Y_1	Z_1	$P_X^{(1)}$	$P_Y^{(1)}$	$P_Z^{(1)}$
				⋮		
N	X_N	Y_N	Z_N	$P_X^{(N)}$	$P_Y^{(N)}$	$P_Z^{(N)}$
N+1	X_{hal}	Y_{hal}	Z_{hal}	$P_X^{(hal)}$	$P_Y^{(hal)}$	$P_Z^{(hal)}$

The first N lines of the file give the rare gas configuration, and the final line is the halide configuration. The first column is an integer index. X_N , Y_N and Z_N are the Cartesian coordinates in units of $a_0/10$. $P_X^{(N)}$, etc. are the corresponding components of the momentum vector, with units $1000 m_e a_0/\tau_0$.

C4.3 Running the program: example of Ar_6I^-

To show how the simulated program is used we will show how to run the program using the input file "in1" shown in section C4.1. This input file is set up to find a minimum energy configuration of a Ar_6I^- cluster with no many body interaction, starting

from a random configuration of atoms. Below, we present the actual computer session. The user input is shown in boldface:

```
> amain
File name for parameters
in1
Parameter file name : in1
initial poten =   -0.21661377022324 eV
initial energy =  -0.21661377022324 eV

Time (ps)    fac^2     Temp. (K)       Target Temp.   Avg. Poten (eV)
1.21898128  0.00000000     46.81750652      0.00000001     -0.25897768
2.42828811  0.00000000     50.57195025      0.00000001     -0.30302060
3.63759493  0.00000000     27.21839795      0.00000001     -0.33263926
4.84690176  0.00000000     18.96968172      0.00000001     -0.36209274
6.05620858  0.00000000      9.81563601      0.00000001     -0.37959737
7.26551541  0.00000000      3.20781834      0.00000001     -0.38662604
8.47482223  0.00000001      0.92424026      0.00000001     -0.38892251
                                 :
                                 :
44.75402699  0.00000011      0.08753625      0.00000001     -0.38971920
45.96333382  0.00000012      0.08580413      0.00000001     -0.38972086
47.17264064  0.00000012      0.08425250      0.00000001     -0.38972236
48.38194747  0.00000012      0.08268986      0.00000001     -0.38972385
>
```

Here we see that because the factor for rescaling the velocities is practically zero the cluster cools quite rapidly. We can then examine the Rg-X and Rg-Rg bond distances given in the output file "output_file1_6," to ascertain the structure of this cluster. Alternately, one could plot the data from the averaged configuration file "average_file1_6," with a three-dimensional plotting program. We find that this configuration has the six argons all about 4 Å from the iodine ion, and arranged in a warped trapezoid-like arrangement, as shown in figure C1.

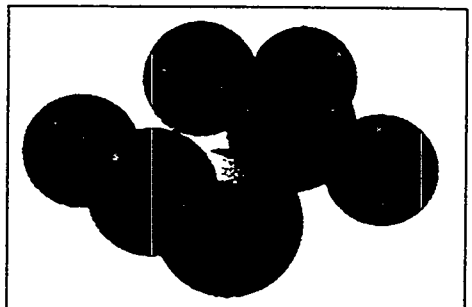

Figure C1. A local minimum energy geometry of Ar_6I.

C4.3.1 Gradient minimization when near a local minimum

To illustrate the use of the gradient minimization procedure for optimizing the binding energy, we modify the input "in1" to "in2," showing below only the lines of "in2" that are different from "in1:"

```
                                        :
32:    6        !* ninit (1 pos. from file, 2 rand. pos save to file, 3
       to restart, 4 to restart with average, 5 for pos from file,
       averaging only,6 grad min)
                                        :
49:    average_file1_6
50:    output_file2_6
51:    restart_file2_6
52:    average_file2_6
53:    poten_file2_6
```

Here we have changed the file name in line **49** to "average_file1_6," which contains the configuration output from the first run of the program, and set the flag `ninit` to 6 for gradient minimization. We perform the minimization as follows:

```
> amain
File name for parameters
in2
Parameter file name : in2
Initial potential:
                                    eV                          cm-1
  Total potential       = -0.38979933606317              -3143.943
  RG-X Contribution     = -0.27479441639689              -2216.366
  RG-RG Contribution    = -0.11500491966628               -927.577
```

```
Potential after gradient minimization:
                                    eV                          cm-1
  Total potential        = -0.38980032561780          -3143.951
  RG-X Contribution      = -0.27479155778784          -2216.343
  RG-RG Contribution     = -0.11500876782996           -927.608
>
```

We see that because the cluster is already quite cold, the gradient minimization in this case refines the minimum energy only by about 0.03 cm^{-1}.

C4.3.2 Reheating and annealing

Now that we have minimum energy configuration of the Ar_6I^-, we must perform the simulated annealing procedure to try to determine if this is this is the global minimum. To do this we make the following changes to the input file, and rename it "in3:"

```
                                    :
7:    75000    !* nstep used in thermgen
8:    500      !* nskip used in thermgen
9:    5000     !* ntconst (time constant for thermgen, in steps of h)
                                    :
14:   0.200    !* escale (Initial energy eV)
                                    :
32:   1        !* ninit (1 pos. from file, 2 rand. pos save to file, 3
      to restart, 4 to  restart with average, 5 for pos from file,
      averaging only,6 grad min)
                                    :
49:   average_file3_6
50:   output_file4_6
51:   restart_file4_6
52:   average_file4_6
53:   poten_file4_6
```

Here we have changed `nstep` so that the total duration of the SA run will be about 360 ps, and increased `nskip` by a factor of 2, so that the cluster will have more time to equilibrate between rescalings. Most importantly, the time constant `ntconst` has been set to be 10 times as large as nskip, so that the kinetic energies will be rescaled by a

factor of 0.9 and that the cooling will take place slowly. The cluster is heated initially by changing `escale`, the initial kinetic energy, to 200 meV, which will be distributed randomly to the atoms in the cluster. Experience has shown that in general one should not set `escale` to more than about one half of the potential of the minimum energy of the cluster (here *ca.* 400 meV), in order to prevent evaporation of atoms. The starting point is the configuration output from the gradient minimization in the file "average_file3_6," which file name is no place in line **49** of the input file. The flag `ninit` is changed to 1 to start the program with this initial configuration (plus the added kinetic energy). We now do the anneal:

```
> amain
File name for parameters
in3
Parameter file name : in3
initial poten =   -0.38979933606317 eV
initial energy =  -0.18979933606317 eV

Time (ps)     fac^2      Temp. (K)        Target Temp.    Avg. Poten (eV)
2.42828811   0.90000000    104.01207847       0.00000001      -0.28391709
4.84690176   0.90000000     57.03555081       0.00000001      -0.24823823
7.26551541   0.90000000     53.54463990       0.00000001      -0.25235669
9.68412906   0.90000000     67.22189146       0.00000001      -0.27198294
12.10274271  0.90000000     51.55501604       0.00000001      -0.26514639
                                  ⋮
55.63778842  0.90000000     41.73466731       0.00000001      -0.33538076
58.05640207  0.90000000     43.29693083       0.00000001      -0.34037302
60.47501572  0.90000000     37.04248189       0.00000001      -0.33744994
                                  ⋮
106.42867508  0.90000000    21.10488494       0.00000001      -0.37646628
108.84728873  0.90000000    20.05677511       0.00000001      -0.37843554
111.26590238  0.90000000    19.26204642       0.00000001      -0.38014508
                                  ⋮
357.96449475  0.90000001     0.16330365       0.00000001      -0.40021528
360.38310840  0.90000001     0.15893369       0.00000001      -0.40022694
362.80172205  0.90000001     0.14923592       0.00000001      -0.40023536
>
```

We see that at the end of the annealing procedure, the potential energy is lower than that of the previously located minimum, indicating that we have found a different

minimum on the potential energy surface. Examining the output files we find that this is indeed a different isomer of Ar_6I^- with the structure pictured in Figure C2.

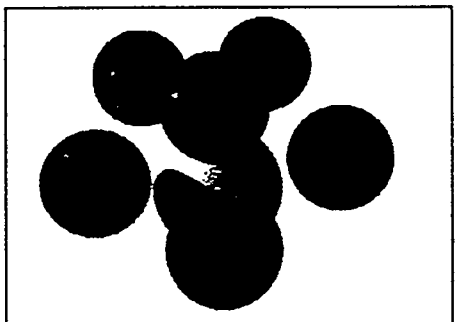

Figure C2. The global minimum energy geometry of Ar_nI^-.

This is the configuration that we recognize as the global minimum energy geometry from Figure 7 of Chapter 4. In this structure, as in the previously located minimum, all of the Ar atoms are in "contact" with the I^- atom. However, we see that in this structure, the number of Ar-Ar nearest neighbor "bonds" is 10, compared to 9 for the isomer pictured in Figure C1. Furthermore, the energies of these structures differ by 85 cm^{-1}, which is reasonably close to the Ar-Ar dimer bond energy (about 100 cm^{-1}).

We know from previous experience that this is the global minimum of Ar_6I^-. However, with an unknown system, a greater number of annealing runs must be performed to ensure that the global minimum has been found. It is not possible to give an exact answer to the question of how many times one must repeat the annealing process, because, as noted above, simulated annealing is not an exact method. The number of runs necessary will increase with the complexity of the system studied. In general, one should perform the process enough times to gain an understanding of the various categories of local minima for a given system and their relative energies.

C4.4. Outline of the program

The subroutines and functions used by the simulated annealing program are presented in Table C1 organized according to which file they reside in.

Table C1. Files, subroutines and functions used by the simulated annealing program

File	Subroutine or Function	Description
param.file		Contains constants and common blocks used by the program
amain.f		Main program; reads input and writes output file
aderiv.f	deriv	Function to calculate the forces on each atom and potential of the cluster for a given atomic configuration
initzandiat.f	initpos	Generates random initial positions for the atoms
	initzangdiat	Removes CM translation and sets overall angular momentum of cluster to zero
	ggubs	Random number generator
thermgen.f	thermgen	Propagates the MD equations of motion according to the scheme described above.
pforce.f	anmmsv	Function for anion MMSV pair potential and force
	hfd_b	Function for HFD-B2 (Rg-Rg) pair potential and force
	pfind	Calculates three-body induction potential and forces from Eq. (C10)
	pfexq	Calculates the three-body "exchange-quadrupole" potential and forces from Eq. (C12)

	porb	Calculates neutral many-body eigenfunctions numerically.
	rneummsv	Calculates neutral MMSV pair potential
	porba	Calculates neutral many-body eigenvalues analytically
	veqd	Calculated exchange charge effect using distributed dipole model
	vindi	Calculates many-body dipole induction potential using the iterative method.
	vindq	Calculates induced quadrupole-induced quadrupole and induced quadrupole-induced dipole potential; for use in conjunction with the subroutine "vindi"
	vinddq	Calculates many-body dipole and quadrupole potential, using the iterative method described in Chapter 4.
	exg	Charge exchange calculation of three body potential from Eq. (4.34)
	erf	Error function adapted from Ref. [3]
	vat	Calculates three-body Axilrod-Teller potential
averages.f	averages	Calculates average parameters during a constant energy MD run
ch.f	ch	EISPACK subroutine for finding the eigenvalues and eigenvectors of a complex Hermitian matrix

C5. Source code for the simulated annealing program.

C5.1. File "Makefile"

The makefile used for recompiling the simulated annealing program is shown below. To recompile the program one enters simply "make" because the makefile has the default name. The executable code will be save in the subdirectory "RUN" with the makefile as shown.

```
OBJ  = amain.o aderiv.o initzangdiat.o thermgen.o pforce.o averages.o
ch.o gmin.o

LIB =  -lgl_s -lm -lmpc
OPT = -O
OPT3 = -g


main : $(OBJ)
      f77 $(OPT) $(OBJ) -o RUN/amain

.f.o :
      f77 $(OPT) -c  $<
.c.o :
      cc  $(OPT) -c  $<
```

C5.2. File "param.file"

```
C
      IMPLICIT double precision (A-H,O-Z)
c      PARAMETER (ncluster=19,nmolec=1,natmax = ncluster+nmolec)
c      PARAMETER (NQ=6,NDIM=NQ*Natmax,NEQM=NDIM,NEQ=NDIM)
      parameter (ncl=25,nmolec=1,natmaxc = ncl+nmolec)
      parameter (NQ=6, NDIMC=NQ*natmaxc, NEQMC=NDIMC, NEQC=NDIMC)
      PARAMETER (ECONV=2196.d0, DCONV=0.0529d0)
      parameter(pi=3.1415926535898d0,a0=0.529177249d0,
     &  harev=27.2113961d0,evtocm=8065.5410d0)
      parameter(evtoj=1.60217733d-19,amutokg=1.6605402d-27,
     &  hztocm=3.335640952d-11,hbarev=6.5821220d-16,
     &  hbar=1.05457266d-34)

c
      common /nc/ncluster,natmax,NDIM,NEQM,NEQ
      common /param/rmc,rm1,eec,eeint,sigc,sigint,nrgpot
      common /param2/nhalpot,neigval,delta
      common /atoms/poten,rgrgpot,rgxpot,cddpot,exqpot,vindit,vexqd,
     &  rgdrgq,rgqq,rgqxd,viq,cdpot,cqpot,cdqpot,gexpot,ddispot,
     &  qdispot,gextot,atpot
```

```
      common /atoms1/rgxXpot,rgxIpot,rgxIIpot
      common /atoms2/potfin,tempfin
      common /threeb/indflag,iexqflag,indi,iexqd,indq,indqi,iexg,
     &  iaxtel

c ncluster = number of rare gas atoms
c nmolec = number of halide atoms
c Constants from Cohen & Taylor, Rev. Mod. Phys. Vol. 59, No. 4
c hbarev units eV*s
c hbar units J*s
```

C5.3. File "amain.f"

```
*
*  Simulated annealing program by Zhiming Li and Prof. CC Martens (UC
Irvine)
*  Modified and extended by Ivan Yourshaw
*
      include 'param.file'
      common /anion/ p(10),q(10)
      common /neutral/ px(10),p1(10),p2(10),soconst
      common /nonadpar/ polind,betaexq,cutexq,polrg,polx,theta6,
     &  pqrg,pqx,c9at
      common/box/boxsize,cutoff
C
      DIMENSION Y0(neqmc),dx(ncl),dy(ncl),dz(ncl),dist(ncl),
     &          ang(ncl,ncl),yave(neqc),rgx(2,ncl),rgrg(2,ncl,ncl),
     &          dip(natmaxc),qin(natmaxc)
      double precision p(10),q(10),px(10),p1(10),p2(10)
      character*30 aparam,initcond,outcond,restart,avrestart,potfile
      character*80 comment1,comment2
      integer iarr(3)
C
c     OPEN FILES
c
      write(*,*) 'File name for parameters'
      read(*,*) aparam
      write(*,*) 'Parameter file name : ',aparam
      open(7, file=aparam)
      read(7,*)rmc
      read(7,*)rm1
      read(7,*)eec
      read(7,*)eeint
      read(7,*)sigc
      read(7,*)sigint
c
      READ(7,*)nstepT
      READ(7,*)nskipT
      read(7,*)ntconst
      read(7,*)nave     !* Number of steps for averaging
      READ(7,*)h
      READ(7,*)ekelvin
c
      READ(7,*)dseed
```

```
      dseedint=dseed
      READ(7,*)escale          !* Initial kinetic energy, in eV
      escale = (escale/harev)*100.d0      !* Convert to hartree/100
      read(7,*)boxsize         !* In ang, for initpos
      boxsize = (boxsize/a0)*10.d0
      read(7,*)cutoff          !* In Ang, for initpot
      cutoff = (cutoff/a0)*10.d0
      read(7,*)ncluster        !* Number of rare gas atoms
      read(7,*)nrgpot          !* 1 for RG-RG LJ poten, 2 for Aziz
      read(7,*)nhalpot         !* 1 for anion, 2 for neutral (num.
forces)
*                           3 for neutral (forces by central
differences)
      read(7,*)neigval         !* eigenvalue # for neutral potential
*                           1 or 2 for X, 3 or 4 for I, 5 or 6 for II
      if (nhalpot.eq.4) then
        if ((neigval.eq.1).or.(neigval.eq.2)) then
          naval = 1
        elseif ((neigval.eq.3).or.(neigval.eq.4)) then
          naval = 2
        elseif ((neigval.eq.5).or.(neigval.eq.6)) then
          naval = 3
        endif
        neigval = naval
      endif
      read(7,*) delta          !* delta-r for numerical derivatives
(Ang)
      delta = delta*10.0d0/a0     !* convert to a0/10
      read(7,*) gstepin        !* initial step for gradient
minimization(A)
      gstepin = gstepin*10.0d0/a0   !* convert to a0/10
      read(7,*) gsmall         !* convergence test for gradient min (eV)
      gsmall = gsmall*100./harev    !* convert to hart/100
      read(7,*) indflag        !* Three-body induction, 0=off, 1=on
      read(7,*) iexqflag       !* Charge-Exchange Quadrupole, 0=off,
1=on
      read(7,*) indi           !* Iterated many-body (dipole) induction
*                       !* 0=off,1=on
      if (indi.eq.1) then
        indflag = 0
      endif
      read(7,*) iexqd          !* Distributed dipole exchange quadrupole
1=on
      if (iexqd.eq.1) then
        iexqflag = 0
      endif
      read(7,*) indq           !* Charge Induced Rg quadrupoles, 0=off,
1=on
      if (indq.eq.1) then
        indflag = 0
        indi = 1         !* Need dipole calculation to do quadrupoles
      endif
      read(7,*) iexg           !* Charge-Gaussian exchange, 0=off, 1=on
      if (iexg.eq.1) then
        iexqflag = 0
        iexqd = 0
      endif
```

```
      read(7,*) indqi          !* Iterated dipole and quadrupole
induction
      if (indqi.eq.1) then     !*  turns off other induction options
        indflag = 0
        indi = 0
        indq = 0
      endif
      read(7,*) iaxtel         !* Axilrod-Teller, 0=off, 1=on
      read(7,*) ninit   !* 1 to use initial positions from file
initcond,
*              2 to generate random initial positions and save to
initcond,
*                3 to restart from previous run
*                4 to restart with averages from prev run
*            5 to do averaging run only starting with initcond file
*            6 to do gradient potential minimization (pos saved to avg)
*            7 for Powell search
      read(7,*) polind         !* Rare gas polarizibility for 3-bod ind
*                            units Angstrom^3
      read(7,*) polrg          !* Rare gas polarizability for iterated
ind
*                            units a0^3
      read(7,*) polx           !* Halide polarizability for iterated ind
*                            units a0^3
      read(7,*) pqrg           !* Rare gas quadrupole polarizability
(a0^5)
      read(7,*) pqx            !* Halide quadrupole polarizability
(a0^5)
      read(7,*) betaexq        !* Range parameter for exchange
quadrupole
*                            units Angstrom^-1
      betaexq = betaexq*a0/10.d0     !* Converted to (a0/10)^-1
      read(7,*) cutexq         !* Cutoff distance for exchange quad.
(ang)
      cutexq = (cutexq/a0)*10.d0     !* Converted to (a0/10)^-1
      read(7,*) theta6         !* Quadrupole dispersion coeff. (e*a0^8)
      read(7,*) c9anion        !* Ax-Tel C9 for anion (eV*Ang^9)
      read(7,*) c9neut         !* Ax-Tel C9 for neutral (eV*Ang^9)
      if (nhalpot.eq.1) then
        c9at = c9anion/harev/a0**9  !* convert to au
      else
        c9at = c9neut/harev/a0**9
      endif
      read(7,*) soconst        !* Atomic spin-orbit const. in eV
      soconst = (soconst/harev)*100.d0     !* convert to hartree/100
      read(7,*) (p(i), i=1,8) !* RG-X anion MMSV parameters
      read(7,*) (px(i), i=1,8)       !* RG-X neutral X1/2 MMSV
parameters
      read(7,*) (p1(i), i=1,8)       !* RG-X neut I3/2 MMSV parameters
      read(7,*) (p2(i), i=1,8)       !* RG-x neut II1/2 MMSV parameters
      read(7,*) (q(i), i=1,9) !* RG-RG HFD-B parameters
      read(7,*) initcond       !* Initial condition file name
      read(7,*) outcond        !* output file name
      read(7,*) restart        !* restart file name
      read(7,*) avrestart      !* average restart file name
      read(7,*) potfile        !* potential save file name
      read(7,101) comment1
```

```
      read(7,101) comment2
101   format(a80)
c
      natmax = ncluster+nmolec
      NDIM=NQ*Natmax
      NEQM=NDIM
      NEQ=NDIM
c
      close(7)
c
      if ((ninit.eq.1).or.(ninit.eq.5).or.(ninit.eq.6).or.
     &(ninit.eq.7)) then
        open (7, file = initcond)
        do 123 n = 1,neq-5,6
        read(7,*)junk,y0(n),y0(n+1),y0(n+2),y0(n+3),
     .  y0(n+4),y0(n+5)
 123    continue
        close (7)
      elseif (ninit.eq.2) then
        call initpos(y0,dseed)
        open(7,file=initcond)
        icount = 0
        do n = 1,neq-5,6
          icount = icount+1
          write(7,*) icount,y0(n),y0(n+1),y0(n+2),y0(n+3),
     &              y0(n+4),y0(n+5)
        enddo
      elseif (ninit.eq.3) then
        open (7, file = restart)
        do n = 1,neq-5,6
          read(7,*)junk,y0(n),y0(n+1),y0(n+2),y0(n+3),
     .    y0(n+4),y0(n+5)
        enddo
        close (7)
      elseif (ninit.eq.4) then
        open (7, file = avrestart)
        do n = 1,neq-5,6
          read(7,*)junk,y0(n),y0(n+1),y0(n+2),y0(n+3),
     .    y0(n+4),y0(n+5)
        enddo
        close (7)
      endif
c
      if ((ninit.eq.2).or.(ninit.eq.1).or.(ninit.eq.5)) then
        call initzangdiat(y0,dseed,escale)
      endif
      if (ninit.lt.5) then
        call thermgen(y0,nstepT,nskipT,ntconst,h,ekelvin)
      endif

      iframe = 1
      call writeframe(y0,iframe)
c
c  write out the final positions
*  (and pos. and momenta to restart file)
c
      open (8, file = outcond)
```

```
      write(8,*) 'RGX Simulated Annealing Program'
      write(8,*)
      write(8,*) comment1
      write(8,*) comment2
      write(8,*)
      write(8,*) 'init cond file: ',initcond
      write(8,*) 'output file: ',outcond
      write(8,*) 'restart file: ',restart
      write(8,*) 'average config. file: ',avrestart
      write(8,*) 'potential file: ',potfile
      write(8,*)
      call idate(iarr)
      write(8,300) iarr(2),iarr(1),iarr(3)
      call itime(iarr)
      write(8,301) iarr(1),iarr(2),iarr(3)
300   format('Date: ',i2,'/',i2,1x,i4)
301   format('Time: ',i2,':',i2,':',i2)
      write(8,*)
      write(8,*) '******************Parameters******************'
      write(8,*) 'nstep = ',nstepT,'      nskip = ', nskipT
      write(8,*) 'ntconst = ',ntconst,'      nave = ',nave
      write(8,*) 'h = ',h
      write(8,*) 'initial dseed = ',dseedint
      write(8,*) 'escale (eV) = ',escale*harev/100.d0
      write(8,*) 'boxsize = ',boxsize*a0/10.,' Ang'
      write(8,*) 'cutoff = ',cutoff*a0/10.,' Ang'
      write(8,*) 'nrgpot = ',nrgpot,'      nhalpot = ',nhalpot
      if ((nhalpot.eq.2).or.(nhalpot.eq.3).or.(nhalpot.eq.4)) then
        write(8,*) 'neigval = ',neigval
        write(8,*) 'soconst = ',soconst*harev/100.
        write(8,*) 'delta = ',delta*a0/10.0d0
      endif
      if (nhalpot.eq.1) then
        if (indflag.eq.1) then
          write(8,981) polind
981       format('Dipole induction ON',3x,'polind =',f10.6,3x,'Ang^3')
        endif
        if (iexqflag.eq.1) then
          write(8,991) betaexq/a0*10.d0,cutexq*a0/10.d0
991       format('ExQ ON',3x,'betaexq =',f10.6,3x,
     &         'cutexq =',f10.6)
        endif
        if (indi.eq.1) then
          write(8,1001) polrg,polx
1001      format('Iter dipole Ind ON',3x,'Rg pol = ',f10.6,3x,
     &           'Hal pol = ',f10.6,3x,'a0^3')
        endif
        if (indq.eq.1) then
          write(8,1011) pqrg
1011      format('Rg charge induced quadrupole ON',3x,'Rg Quad. pol =',
     &      f10.6,3x,'a0^5')
        endif
        if (indqi.eq.1) then
          write(8,1085) polrg,polx,pqrg,pqx
1085      format('Dipole-Quadrupole iter. Induction ON',/,
     &      'Dipole polarizabilities, a0^3: Rg = ',f10.6,3x,'Hal = ',
     &      f10.6,/,'Quad. polarizabilities, a0^5: Rg = ',f10.6,3x,
```

```
     &       'Hal = ',f10.6)
          endif
          if (iexqd.eq.1) then
            write(8,1003) betaexq*10./a0,theta6
1003        format('Dist. Dip. ExQ ON',3x,'betaexq =',f10.6,3x,
     &       'theta6 = ',f15.3)
          endif
          if (iexg.eq.1) then
            write(8,1057) betaexq*10./a0,theta6
1057        format('Gaussian Ex. Chg. ON',3x,'beta = ',f10.6,3x,
     &     'theta6 = ',f15.3)
          endif
          write(8,*) 'Anion rg-x MMSV parameters:'
          write(8,173) (p(i),i=1,8)
        else
        write(8,*) 'Neutral rg-x MMSV Parameters (X,I,II diatom states):'
          write(8,173) (px(i),i=1,8)
          write(8,173) (p1(i),i=1,8)
          write(8,173) (p2(i),i=1,8)
        endif
        if (iaxtel.eq.1) then
          write(8,1083) c9at*harev*a0**9
1083      format('Axilrod-Teller ON',3x,'C9 (eV*Ang^9) = ',f10.6)
        endif
        if (nrgpot.eq.2) then
          write(8,*) 'RG-RG HFD-B Parameters:'
          write(8,173) (q(i),i=1,9)
        endif
173     format(2x,5g14.8,/,2x,5g14.8)
        write(8,1051) a0,harev,evtocm
1051    format('a0 = ',g18.12,1x,'harev = ',g18.12,1x,'evtocm = ',g18.12)
*
        if (ninit.ge.5) goto 5000
*
        write(8,*) '*********Configuration After Annealing**********'
        write(8,*)
        open (1, file =restart)
        icount = 0
c        write(8,*) '       Atom coordinates (a0/10)'
        do 124 n = 1,neq-5,6
        icount=icount+1
c        write(8,*)icount,y0(n),y0(n+1),y0(n+2)
        write(1,*)icount,y0(n),y0(n+1),y0(n+2),y0(n+3),y0(n+4),
     & y0(n+5)
124     continue
        close(1)
c        write(8,*)
        icount = 0
        do n=1,neq-11,6
          icount=icount+1
          dx(icount) = y0(n)-y0(neq-5)
          dy(icount) = y0(n+1)-y0(neq-4)
          dz(icount) = y0(n+2)-y0(neq-3)
          dist(icount) = (y0(n)-y0(neq-5))**2 + (y0(n+1)-y0(neq-4))**2
     &          + (y0(n+2)-y0(neq-3))**2
          dist(icount) = sqrt(dist(icount))
        enddo
```

```
      do i=1,ncluster
        do j=1,ncluster
          dot = dx(i)*dx(j)+dy(i)*dy(j)+dz(i)*dz(j)
          if (i.eq.j) then
            ang(i,j) = 0.
          else
            ang(i,j) = acos(dot/(dist(i)*dist(j)))
            ang(i,j) = ang(i,j)*180/pi
          endif
        enddo
      enddo
      write(8,*)
      write(8,*) '      RG-RG Angles (deg.)'
      do n=1,ncluster
        write(8,201) n,(ang(n,i),i=1,ncluster)
        if (mod(ncluster,10).ne.0) then
          write(8,*)
        endif
      enddo
200   format (i3, 3x, f8.4)
201   format (i2, 10f7.2,/,2x,10f7.2)

      write(8,*)
      write(8,*) 'Final temp. (K):',tempfin
      write(8,*)
c
c
5000  if (ninit.lt.6) then
        call averages(y0,nave,h,potave,potsd,pmin,prgrg,prgx,
     &   pind,pexq,pex,pddis,pqdis,pextot,pindi,pcd,pcq,pinddq,
     &   prgdrgq,prgqq,prgqxd,piq,
     &   pexqd,prgxX,prgxI,prgxII,ekinave,energy,yave,rgx,rgrg,
     &   dip,qin,paxtel)
       elseif (ninit.eq.6) then
        call gmin(y0,gstepin,gsmall,pmin,prgrg,prgx,pind,pcd,pcq,
     &    pinddq,pexq,pex,pddis,pqdis,pextot,pindi,prgdrgq,prgqq,
     &    prgqxd,piq,pexqd,prgxX,
     &    prgxI,prgxII,rgx,rgrg,dip,qin,paxtel)
       elseif (ninit.eq.7) then
        call pwmin(y0,gstepin,gsmall,pmin,prgrg,prgx,pind,pcd,pcq,
     &    pinddq,pexq,pex,pddis,pqdis,pextot,pindi,prgdrgq,prgqq,
     &    prgqxd,piq,pexqd,prgxX,
     &    prgxI,prgxII,rgx,rgrg,dip,qin,paxtel)
       endif
       open(9,file=potfile)
       if (indi.eq.1) then
         pind = pindi
       endif
       if (indqi.eq.1) then
         pind = pinddq
         prgdrgq = pcd
         prgqq = pcq
       endif
       if (iexqd.eq.1) then
         pexq = pexqd
       endif
       if (iexg.eq.1) then
```

```
        pexq = pextot
       endif

       write(9,501) ncluster,pmin*harev/100.,prgx*harev/100.,
      &  prgrg*harev/100.,pind*harev/100.,pexq*harev/100.,
      &  prgdrgq*harev/100.,prgqq*harev/100.,pddis*harev/100.,
      &  pqdis*harev/100.,pex*harev/100.,paxtel*harev/100.
       close(9)
501    format(i3,1x,11g16.8)
*
*  Fill in lower triangles
*
       if (ninit.lt.6) then
         do i=2,ncluster
           do j=1,i-1
             rgrg(1,i,j)=rgrg(1,j,i)
             rgrg(2,i,j)=rgrg(2,j,i)
           enddo
         enddo

         write(8,*)
         write(8,*) '***************Averaging Run******************'
         write(8,*)
         write(8,*) 'Avg. Potential (eV) = ',potave*harev/100.
         write(8,*) 'Std. Dev. of poten (eV) = ',potsd*harev/100.
         write(8,*)
         write(8,*) 'Potential at avg. configuration (eV) : '
       else
         write(8,*)
         write(8,*) '***********Gradient Minimization**************'
         write(8,*)
         write(8,*) 'gstep final = ',gstepin*a0/10.,' Ang'
         write(8,*) 'Convergence = ',gsmall*harev/100.,' ev'
         write(8,*)
       endif
       write(8,7000) pmin*harev/100.,pmin*harev/100.*evtocm
7000   format(33x,'eV',24x,'cm-1',/,
      &       '  Total potential       = ',g21.14,3x,f14.3)
       if (nhalpot.eq.1) then
         write(8,7005) prgx*harev/100.,prgx*harev/100.*evtocm
7005     format('  RG-X Contribution     = ',g21.14,3x,f14.3)
       elseif (nhalpot.lt.4) then
         if (neigval.le.2) then
           write (8,7010)prgx*harev/100.,prgx*harev/100.*evtocm,
      &     prgxI*harev/100.,prgxI*harev/100.*evtocm,
      &     prgxII*harev/100.,prgxII*harev/100.*evtocm
7010       format('  RG-X X State Contrib  = ',g21.14,3x,f14.3,/,
      &           ' (Vertical I contrib  = ',g21.14,3x,f14.3,')',/,
      &           ' (Vertical II contrib = ',g21.14,3x,f14.3,')')
         elseif (neigval.le.4) then
           write (8,7015)prgx*harev/100.,prgx*harev/100.*evtocm,
      &     prgxX*harev/100.,prgxX*harev/100.*evtocm,
      &     prgxII*harev/100.,prgxII*harev/100.*evtocm
7015       format('  RG-X I State Contrib  = ',g21.14,3x,f14.3,/,
      &           ' (Vertical X contrib  = ',g21.14,3x,f14.3,')',/,
      &           ' (Vertical II contrib = ',g21.14,3x,f14.3,')')
         else
```

```
          write (8,7020)prgx*harev/100.,prgx*harev/100.*evtocm,
     &     prgxX*harev/100.,prgxX*harev/100.*evtocm,
     &     prgxI*harev/100.,prgxI*harev/100.*evtocm
7020      format('  RG-X II State Contrib  = ',g21.14,3x,f14.3,/,
     &           ' (Vertical X contrib  = ',g21.14,3x,f14.3,')',/,
     &           ' (Vertical I contrib  = ',g21.14,3x,f14.3,')')
        endif
       elseif (nhalpot.eq.4) then
        if (neigval.eq.1) then
          write (8,7010)prgx*harev/100.,prgx*harev/100.*evtocm,
     &     prgxI*harev/100.,prgxI*harev/100.*evtocm,
     &     prgxII*harev/100.,prgxII*harev/100.*evtocm
        elseif (neigval.eq.2) then
          write (8,7015)prgx*harev/100.,prgx*harev/100.*evtocm,
     &     prgxX*harev/100.,prgxX*harev/100.*evtocm,
     &     prgxII*harev/100.,prgxII*harev/100.*evtocm
        elseif (neigval.eq.3) then
          write (8,7020)prgx*harev/100.,prgx*harev/100.*evtocm,
     &     prgxX*harev/100.,prgxX*harev/100.*evtocm,
     &     prgxI*harev/100.,prgxI*harev/100.*evtocm
        endif
       endif
       write(8,7025)prgrg*harev/100.,prgrg*harev/100.*evtocm
7025   format('  RG-RG Contribution    = ',g21.14,3x,f14.3)
       if (indflag.eq.1) then
         write(8,7030) pind*harev/100., pind*harev/100.*evtocm
7030     format('  Charge-dipole-dipole   = ',g21.14,3x,f14.3)
       endif
       if (iexqflag.eq.1) then
         write(8,7035) pexq*harev/100., pexq*harev/100.*evtocm
7035     format('  Charge-Ex. Quadrupole  = ',g21.14,3x,f14.3)
       endif
       if (indi.eq.1) then
         write(8,7040) pindi*harev/100.,pindi*harev/100.*evtocm
7040     format('  Dipole Induction       = ',g21.14,3x,f14.3)
       endif
       if (indq.eq.1) then
         write(8,7045)prgdrgq*harev/100.,prgdrgq*harev/100.*evtocm,
     &    prgqq*harev/100.,prgqq*harev/100.*evtocm,
     &    prgqxd*harev/100.,prgqxd*harev/100.*evtocm,
     &    piq*harev/100.,piq*harev/100.*evtocm
7045     format('  Rg Ind Qp - Rg Ind Dip = ',g21.14,3x,f14.3,/,
     &           '  Rg Ind Qp - Rg Ind Qp  = ',g21.14,3x,f14.3,/,
     &           '  Rg Ind Qp - X Ind Dip  = ',g21.14,3x,f14.3,/,
     &           ' (Total Ind. Qp. = ',g21.14,3x,f14.3,')')
       endif
       if (indqi.eq.1) then
         write(8,7050)pcd*harev/100.,pcd*harev/100.*evtocm,
     &    pcq*harev/100.,pcq*harev/100.*evtocm,
     &    pinddq*harev/100.,pinddq*harev/100.*evtocm
7050     format('  Dipole Induction      = ',g21.14,3x,f14.3,/,
     &           '  Quadrupole Induction  = ',g21.14,3x,f14.3,/,
     &           ' (Total Induction       = ',g21.14,3x,f14.3,')')
       endif
       if (iexqd.eq.1) then
         write(8,7055) pexqd*harev/100.,pexqd*harev/100.*evtocm
7055     format('  Ex. Quad. (dist dip)  = ',g21.14,3x,f14.3)
```

```
      endif
      if (iexg.eq.1) then
        write(8,7060)pex*harev/100.,pex*harev/100.*evtocm,
     &   pddis*harev/100.,pddis*harev/100.*evtocm,
     &   pqdis*harev/100.,pqdis*harev/100.*evtocm,
     &   pextot*harev/100.,pextot*harev/100.*evtocm
7060    format('  Ex. Gaussian Charge    = ',g21.14,3x,f14.3,/,
     &         '  Disp. Dipole           = ',g21.14,3x,f14.3,/,
     &         '  Disp. Quadrupole       = ',g21.14,3x,f14.3,/,
     &         ' (Total Ex. + Disp. MP   = ',g21.14,3x,f14.3,')')
      endif
      if (iaxtel.eq.1) then
        write(8,7065)paxtel*harev/100.,paxtel*harev/100.*evtocm
7065    format('  Axilrod-Teller         = ',g21.14,3x,f14.3)
      endif
      if (ninit.ne.6) then
        write(8,*)
        write(8,*) 'Avg. Kinetic energy (eV) = ',ekinave*harev/100.
        write(8,*) 'Total Energy (eV) = ',energy*harev/100.
        write(8,*)
      endif
      open(9,file=avrestart)
      icount = 0
      do i = 1,neq-5,6
        icount = icount +1
        if (ninit.lt.6) then
          write(9,*) icount,yave(i),yave(i+1),yave(i+2),0.,0.,0.
        else
          write(9,*) icount,y0(i),y0(i+1),y0(i+2),0.,0.,0.
        endif
      enddo

      if (ninit.lt.6) then
        write(8,*) 'Average / Std. Dev. of RG-X distances (Angst):'
        write(8,*)
        do n=1,ncluster
          write(8,350) n,rgx(1,n)*a0/10.,rgx(2,n)*a0/10.
        enddo
350     format(i3,3x,f8.4,g12.4)

        write(8,*)
        write(8,*) 'Average of RG-RG distances (Angst.):'
        write(8,*)

        do i=1,ncluster
          write(8,351) i,(rgrg(1,i,j)*a0/10.,j=1,ncluster)
          if (mod(ncluster,5).ne.0) then
            write(8,*)
          endif
        enddo
      endif

351   format(i3,3x,5f8.4,/,6x,5f8.4,/,6x,5f8.4,/,6x,5f8.4)
353   format (i3,3x,5g12.4,/,6x,5g12.4,/,6x,5g12.4,/,6x,5g12.4)

      if (ninit.ge.6) then
        icount = 0
```

```
        do n=1,neq-11,6
          icount=icount+1
          dx(icount) = y0(n)-y0(neq-5)
          dy(icount) = y0(n+1)-y0(neq-4)
          dz(icount) = y0(n+2)-y0(neq-3)
          dist(icount) = (y0(n)-y0(neq-5))**2 +
     &      (y0(n+1)-y0(neq-4))**2 + (y0(n+2)-y0(neq-3))**2
          dist(icount) = sqrt(dist(icount))
        enddo
        do i=1,ncluster
          do j=1,ncluster
            dot = dx(i)*dx(j)+dy(i)*dy(j)+dz(i)*dz(j)
            if (i.eq.j) then
              ang(i,j) = 0.
            else
              ang(i,j) = acos(dot/(dist(i)*dist(j)))
              ang(i,j) = ang(i,j)*180/pi
            endif
          enddo
        enddo
        write(8,*)
        write(8,*) '      RG-X (angs)'
        do n=1,ncluster
          write(8,200) n,dist(n)*a0/10.
        enddo
        write(8,*)
        write(8,*) '      RG-RG Angles (deg.)'
        do n=1,ncluster
          write(8,201) n,(ang(n,i),i=1,ncluster)
          if (mod(ncluster,10).ne.0) then
            write(8,*)
          endif
        enddo
      endif

      if (ninit.le.6) then
      if ((indi.eq.1).or.(indqi.eq.1)) then
        write(8,*) 'Rg Dipole Moments, e*a0 :'
        write(8,*)
        write(8,2005) (dip(i),i=1,ncluster)
2005    format(8(2x,f8.6))
        write(8,*)
        write(8,2010) dip(ncluster+1)
2010    format('Halide dipole moment : ',f12.10)
      endif

      if ((indq.eq.1).or.(indqi.eq.1)) then
        write(8,*)
        write(8,*) 'Rg Quadrupole Moments, e*a0^2 :'
        write(8,*)
        write(8,2005) (qin(i),i=1,ncluster)
      endif
      if (indqi.eq.1) then
        write(8,*)
        write(8,2045) qin(ncluster+1)
2045    format('Halide quadrupole moment : ',f12.10)
      endif
```

```
      endif

      close(8)

      stop
      end
```

C5.4. File "aderiv.f"

```
c     Written by Zhiming Li
*     Changed by IY to use anion MMSV potential, and Aziz potential
*     for RG-RG & open shell neutral potential
c     subroutine that calculates forces in a rare cluster
c     with a halogen anion or neutral impurity
c
      subroutine deriv(force,y)
      include 'param.file'
      common /anion/ p(10),q(10)
      common /neutral/ px(10),p1(10),p2(10),soconst
      common /nonadpar/ polind,betaexq,cutexq,polrg,polx,theta6,
     &  pqrg,pqx,c9at
      common /dipoles/dx(natmaxc),dy(natmaxc),dz(natmaxc),
     &  qind(natmaxc),dpol(natmaxc,3),qpol(natmaxc,3,3)
c
      double precision p(10)  !* anion MMSV parameters
      double precision q(10)
      double precision eval(6),evala(3)
      dimension px(10),p1(10),p2(10)
      dimension force(neqc),y(neqc)
      dimension RX(ncl+nmolec), RY(ncl+nmolec),
     .RZ(ncl+nmolec),
     .FX(ncl+nmolec), FY(ncl+nmolec),
     .FZ(ncl+nmolec)
      double precision
dx(natmaxc),dy(natmaxc),dz(natmaxc),dx2(natmaxc),
     & dy2(natmaxc),dz2(natmaxc),dhalpair(ncl),drgpair(ncl),
     & qhalpair(ncl),qrgpair(ncl)
c
C
**********************************************************************
c
c      write(*,*) 'aderiv'
       DO 100 I = 1, ncluster+nmolec
c
           FX(I) = 0.0d0
           FY(I) = 0.0d0
           FZ(I) = 0.0d0

100     CONTINUE
c
      icount = 0
c
      do 10 n = 1,neq-5,6
      icount = icount + 1
```

```
      rx(icount) = y(n)
      ry(icount) = y(n+1)
      rz(icount) = y(n+2)
c
 10   continue
c
        poten = 0.0d0          !* Total potential
        rgxpot = 0.0d0         !* RG-X contribution
        rgrgpot = 0.0d0        !* RG-RG contribution
        cddpot = 0.0d0         !* induction (3-body) contribution
        exqpot = 0.0d0         !* Charge- exchange quadrupole
contribution
c
c      calculate cluster-cluster (RG-RG) pairwise interactions
c
      if (nrgpot.eq.1) then          !* Lennard-Jones
        DO 200 I = 1, ncluster-1
c
           RXI = RX(I)
           RYI = RY(I)
           RZI = RZ(I)
           FXI = FX(I)
           FYI = FY(I)
           FZI = FZ(I)
c
c
           DO 199 J = I + 1, ncluster
c
              RXIJ = RXI - RX(J)
              RYIJ = RYI - RY(J)
              RZIJ = RZI - RZ(J)
c

              RIJSQ = RXIJ ** 2 + RYIJ ** 2 + RZIJ ** 2
c
                 SR2    = sigc*sigc / RIJSQ
                 SR6    = SR2 * SR2 * SR2
                 SR12   = SR6 ** 2
                 VIJ    = 4.0d0*eec*(SR12 - SR6)

                 rgrgpot = rgrgpot + VIJ
                 WIJ    =  24.0d0*eec*(2.0d0*sr12 - sr6)/(sigc**2)
                 FIJ    = WIJ * sr2
                 FXIJ   = FIJ * RXIJ
                 FYIJ   = FIJ * RYIJ
                 FZIJ   = FIJ * RZIJ
                 FXI    = FXI + FXIJ
                 FYI    = FYI + FYIJ
                 FZI    = FZI + FZIJ
                 FX(J)  = FX(J) - FXIJ
                 FY(J)  = FY(J) - FYIJ
                 FZ(J)  = FZ(J) - FZIJ
c
c
199        CONTINUE
c
C     ** INNER LOOP ENDS **
```

```
            FX(I) = FXI
            FY(I) = FYI
            FZ(I) = FZI
c
200      CONTINUE

C     ** OUTER LOOP ENDS **

       elseif (nrgpot.eq.2) then           !* Aziz potential (Ar-Ar)

         DO 250 I = 1, ncluster-1
c
            RXI = RX(I)
            RYI = RY(I)
            RZI = RZ(I)
            FXI = FX(I)
            FYI = FY(I)
            FZI = FZ(I)
c
c
            DO 159 J = I + 1, ncluster
c
               RXIJ = RXI - RX(J)
               RYIJ = RYI - RY(J)
               RZIJ = RZI - RZ(J)
c

               RIJ = sqrt(RXIJ ** 2 + RYIJ ** 2 + RZIJ ** 2)
c
                  call hfd_b(q(1),q(2),q(3),q(4),q(5),q(6),q(7),
     &                       q(8),q(9),RIJ,VIJ,FIJ)
                  rgrgpot     = rgrgpot + VIJ
                  FIJ    = FIJ/RIJ
                  FXIJ   = FIJ * RXIJ
                  FYIJ   = FIJ * RYIJ
                  FZIJ   = FIJ * RZIJ
                  FXI    = FXI + FXIJ
                  FYI    = FYI + FYIJ
                  FZI    = FZI + FZIJ
                  FX(J) = FX(J) - FXIJ
                  FY(J) = FY(J) - FYIJ
                  FZ(J) = FZ(J) - FZIJ
c
c
159         CONTINUE
c
C        ** INNER LOOP ENDS **

            FX(I) = FXI
            FY(I) = FYI
            FZ(I) = FZI
c
250      CONTINUE

C     ** OUTER LOOP ENDS **
```

```
      else
        pause 'Error in deriv'
      endif

      poten = poten + rgrgpot

c
c      calculate RG-Halogen pairwise interaction
c
      if (nhalpot.eq.1) then  !* Anion

        i=ncluster+1
c
           RXI = RX(I)
           RYI = RY(I)
           RZI = RZ(I)
           FXI = FX(I)
           FYI = FY(I)
           FZI = FZ(I)
c
        do 299 j=1,ncluster
c
           RXIJ = RXI - RX(J)
           RYIJ = RYI - RY(J)
           RZIJ = RZI - RZ(J)
c
           RIJ = sqrt(RXIJ ** 2 + RYIJ ** 2 + RZIJ ** 2)
c
              call anmmsv(p(1),p(2),p(3),p(4),p(5),p(6),
     &                    p(7),p(8),RIJ,VIJ,FIJ)
              rgxpot     = rgxpot + VIJ
              FIJ = FIJ / RIJ
              FXIJ  = FIJ * RXIJ
              FYIJ  = FIJ * RYIJ
              FZIJ  = FIJ * RZIJ
              FXI   = FXI + FXIJ
              FYI   = FYI + FYIJ
              FZI   = FZI + FZIJ
              FX(J) = FX(J) - FXIJ
              FY(J) = FY(J) - FYIJ
              FZ(J) = FZ(J) - FZIJ
c
 299    continue

        poten = poten + rgxpot
        FX(I) = FXI
        FY(I) = FYI
        FZ(I) = FZI
c
c  Calculate Three body interactions involving halide & 2 RGs
c
        if ((indflag+iexqflag).gt.0) then
          x0 = rx(ncluster+1)
          y0 = ry(ncluster+1)
          z0 = rz(ncluster+1)
          do i = 2 , ncluster
            do j = 1,i-1
```

```
              if (indflag.eq.1) then
                call pfind(polind,x0,y0,z0,rx(i),ry(i),rz(i),
     &                     rx(j),ry(j),rz(j),vcdd,f0x,f0y,
     &                     f0z,f1x,f1y,f1z,f2x,f2y,f2z)
                cddpot = cddpot + vcdd
                fx(ncluster+1) = fx(ncluster+1) + f0x
                fy(ncluster+1) = fy(ncluster+1) + f0y
                fz(ncluster+1) = fz(ncluster+1) + f0z
                fx(i) = fx(i) + f1x
                fy(i) = fy(i) + f1y
                fz(i) = fz(i) + f1z
                fx(j) = fx(j) + f2x
                fy(j) = fy(j) + f2y
                fz(j) = fz(j) + f2z
              endif
              if (iexqflag.eq.1) then
                call pfexq(betaexq,cutexq,x0,y0,z0,rx(i),ry(i),
     &                     rz(i),rx(j),ry(j),rz(j),vexq,f0x,f0y,
     &                     f0z,f1x,f1y,f1z,f2x,f2y,f2z)
                exqpot = exqpot + vexq
                fx(ncluster+1) = fx(ncluster+1) + f0x
                fy(ncluster+1) = fy(ncluster+1) + f0y
                fz(ncluster+1) = fz(ncluster+1) + f0z
                fx(i) = fx(i) + f1x
                fy(i) = fy(i) + f1y
                fz(i) = fz(i) + f1z
                fx(j) = fx(j) + f2x
                fy(j) = fy(j) + f2y
                fz(j) = fz(j) + f2z
              endif
            enddo
          enddo
          poten = poten + cddpot + exqpot
        endif

*
*
*  Many body induction (by iteration) and new exchange quadrupole
model,
*  & quadrupole induction
*
        if (((indi.eq.1).or.(iexqd.eq.1).or.(indq.eq.1)).and.
     &  (indqi.eq.0)) then
          if (indi.eq.1) then
            call vindi(polrg,polx,rx,ry,rz,dx,dy,dz,dhalpair,
     &                 drgpair,vindit)
c            write(*,1001) dsqrt(dx(1)**2+dy(1)**2+dz(1)**2),
c     &        dsqrt(dx(2)**2+dy(2)**2+dz(2)**2),
c     &        dsqrt(dx(3)**2+dy(3)**2+dz(2)**2)
c1001        format(3(f15.12,1x))
          else
            vindit = 0.0d0
          endif
          if (indq.eq.1) then
            call vindq(pqrg,pqx,rx,ry,rz,dx,dy,dz,dhalpair,drgpair,
     &                 qind,rgdrgq,rgqq,rgqxd,viq)
          else
```

```
        xdrgq = 0.0d0
        rgqq = 0.0d0
        rgqxd = 0.0d0
        viq = 0.0d0
      endif
      if (iexqd.eq.1) then
        call veqd(betaexq,theta6,rx,ry,rz,dx,dy,dz,dx2,dy2,dz2,
&                 vexqd)
      else
        vexqd = 0.0d0
      endif

      vindexqd = vindit + vexqd + viq

      do i = 1,ncluster+1        !* Forces calculated numerically

        rxi = rx(i)
        ryi = ry(i)
        rzi = rz(i)
        rx(i) = rxi + delta

        if (indi.eq.1) then
          call vindi(polrg,polx,rx,ry,rz,dx,dy,dz,dhalpair,
&                    drgpair,vinditx)
        else
          vinditx = 0.0d0
        endif

        if (indq.eq.1) then
          call vindq(pqrg,pqx,rx,ry,rz,dx,dy,dz,dhalpair,drgpair,
&                    qind,rgdrgqdum,rgqqdum,rgqxddum,viqx)
        else
          viqx = 0.0d0
        endif

        if (iexqd.eq.1) then
          call veqd(betaexq,theta6,rx,ry,rz,dx,dy,dz,dx2,dy2,dz2,
&                   vexqdx)
        else
          vexqdx = 0.0d0
        endif

        rx(i) = rxi
        ry(i) = ryi + delta

        if (indi.eq.1) then
          call vindi(polrg,polx,rx,ry,rz,dx,dy,dz,dhalpair,
&                    drgpair,vindity)
        else
          vindity = 0.0d0
        endif

        if (indq.eq.1) then
          call vindq(pqrg,pqx,rx,ry,rz,dx,dy,dz,dhalpair,drgpair,
&                    qind,rgdrgqdum,rgqqdum,rgqxddum,viqy)
        else
          viqy = 0.0d0
```

```
            endif

            if (iexqd.eq.1) then
              call veqd(betaexq,theta6,rx,ry,rz,dx,dy,dz,dx2,dy2,dz2,
     &                  vexqdy)
            else
              vexqdy = 0.0d0
            endif

            ry(i) = ryi
            rz(i) = rzi + delta

            if (indi.eq.1) then
              call vindi(polrg,polx,rx,ry,rz,dx,dy,dz,dhalpair,
     &                   drgpair,vinditz)
            else
              vinditz = 0.0d0
            endif

            if (indq.eq.1) then
              call vindq(pqrg,pqx,rx,ry,rz,dx,dy,dz,dhalpair,drgpair,
     &                   qind,rgdrgqdum,rgqqdum,rgqxddum,viqz)
            else
              viqz = 0.0d0
            endif

            if (iexqd.eq.1) then
              call veqd(betaexq,theta6,rx,ry,rz,dx,dy,dz,dx2,dy2,dz2,
     &                  vexqdz)
            else
              vexqdz = 0.0d0
            endif

            rz(i) = rzi
            fx(i) = fx(i) - (vinditx+vexqdx+viqx-vindexqd)/delta
            fy(i) = fy(i) - (vindity+vexqdy+viqy-vindexqd)/delta
            fz(i) = fz(i) - (vinditz+vexqdz+viqz-vindexqd)/delta

          enddo

          poten = poten + vindexqd

        endif
*
*  Iterated dipoles and quadrupoles, Gaussian Exchange & Axilrod-Teller
*
        if ((indqi.eq.1).or.(iexg.eq.1).or.(iaxtel.eq.1)) then

          if (indqi.eq.1) then
            call vinddq(polrg,polx,pqrg,pqx,rx,ry,rz,dpol,dhalpair,
     &      drgpair,qhalpair,qrgpair,qpol,cdpot,cqpot,cdqpot)
          else
            cdpot = 0.
            cqpot = 0.
            cdqpot = 0.
          endif
```

```
      if (iexqd.eq.1) then
        call veqd(betaexq,theta6,rx,ry,rz,dx,dy,dz,dx2,dy2,dz2,
&       vexqd)
      elseif (iexg.eq.1) then
        call exg(betaexq,theta6,rx,ry,rz,gexpot,ddispot,qdispot,
&       gextot)
        vexqd = 0.0d0
      else
        vexqd = 0.
        gexpot = 0.
        ddispot = 0.
        qdispot = 0.
        gextot = 0.
      endif

      if (iaxtel.eq.1) then
        call vat(c9at,rx,ry,rz,atpot)
      else
        atpot = 0.
      endif

      vindexqd = cdqpot + vexqd + gextot + atpot

      do i = 1,ncluster+1        !* Forces calculated numerically

        rxi = rx(i)
        ryi = ry(i)
        rzi = rz(i)
        rx(i) = rxi + delta

        if (indqi.eq.1) then
          call vinddq(polrg,polx,pqrg,pqx,rx,ry,rz,dpol,dhalpair,
&         drgpair,qhalpair,qrgpair,qpol,cddum,cqdum,cdqpotx)
        else
          cdqpotx = 0.
        endif

        if (iexqd.eq.1) then
          call veqd(betaexq,theta6,rx,ry,rz,dx,dy,dz,dx2,dy2,dz2,
&         vexqdx)
        elseif (iexg.eq.1) then
          call exg(betaexq,theta6,rx,ry,rz,gexdum,ddisdum,qdisdum,
&         gextotx)
          vexqdx = 0.0d0
        else
          gextotx = 0.
          vexqdx = 0.0d0
        endif

        if (iaxtel.eq.1) then
          call vat(c9at,rx,ry,rz,atpotx)
        else
          atpotx = 0.
        endif

        rx(i) = rxi
        ry(i) = ryi + delta
```

```
      if (indqi.eq.1) then
        call vinddq(polrg,polx,pqrg,pqx,rx,ry,rz,dpol,dhalpair,
     &    drgpair,qhalpair,qrgpair,qpol,cddum,cqdum,cdqpoty)
      else
        cdqpoty = 0.
      endif

      if (iexqd.eq.1) then
        call veqd(betaexq,theta6,rx,ry,rz,dx,dy,dz,dx2,dy2,dz2,
     &    vexqdy)
      elseif (iexg.eq.1) then
        call exg(betaexq,theta6,rx,ry,rz,gexdum,ddisdum,qdisdum,
     &    gextoty)
        vexqdy = 0.0d0
      else
        gextoty = 0.
        vexqdy = 0.0d0
      endif

      if (iaxtel.eq.1) then
        call vat(c9at,rx,ry,rz,atpoty)
      else
        atpoty = 0.
      endif

      ry(i) = ryi
      rz(i) = rzi + delta

      if (indqi.eq.1) then
        call vinddq(polrg,polx,pqrg,pqx,rx,ry,rz,dpol,dhalpair,
     &    drgpair,qhalpair,qrgpair,qpol,cddum,cqdum,cdqpotz)
      else
        cdqpotz = 0.
      endif

      if (iexqd.eq.1) then
        call veqd(betaexq,theta6,rx,ry,rz,dx,dy,dz,dx2,dy2,dz2,
     &    vexqdz)
      elseif (iexg.eq.1) then
        call exg(betaexq,theta6,rx,ry,rz,gexdum,ddisdum,qdisdum,
     &    gextotz)
        vexqdz = 0.0d0
      else
        gextotz = 0.
        vexqdz = 0.0d0
      endif

      if (iaxtel.eq.1) then
        call vat(c9at,rx,ry,rz,atpotz)
      else
        atpotz = 0.
      endif

      rz(i) = rzi
      fx(i) = fx(i) - (cdqpotx+vexqdx+gextotx+atpotx
     &                 -vindexqd)/delta
```

```
        fy(i) = fy(i) - (cdqpoty+vexqdy+gextoty+atpoty
     &                   -vindexqd)/delta
        fz(i) = fz(i) - (cdqpotz+vexqdz+gextotz+atpotz
     &                   -vindexqd)/delta

      enddo

      poten = poten + vindexqd

    endif


   elseif (nhalpot.eq.2) then    !* Neutral

    if (iaxtel.eq.1) then
      pause 'Ax-Tel not implemented for numerical neutral'
    endif

    call porb(rx,ry,rz,eval)
    rgxpot = eval(neigval)
    rgxXpot = eval(1)
    rgxIpot = eval(3)
    rgxIIpot = eval(5)
*
*     Forces calculated numerically
*
    do i = 1,ncluster+1   !* Forces on rare gases and halogen
      rxi = rx(i)
      ryi = ry(i)
      rzi = rz(i)
      rx(i) = rxi + delta
      call porb(rx,ry,rz,eval)
      fx(i) = fx(i) - (eval(neigval)-rgxpot)/delta
      rx(i) = rxi
      ry(i) = ryi + delta
      call porb(rx,ry,rz,eval)
      fy(i) = fy(i) - (eval(neigval)-rgxpot)/delta
      ry(i) = ryi
      rz(i) = rzi + delta
      call porb(rx,ry,rz,eval)
      fz(i) = fz(i) - (eval(neigval)-rgxpot)/delta
      rz(i) = rzi
    enddo
    poten = poten + rgxpot

   elseif (nhalpot.eq.3) then    !* Neutral, Central Difference

    if (iaxtel.eq.1) then
      pause 'Ax-Tel not implemented for numerical neutral'
    endif

    call porb(rx,ry,rz,eval)
    rgxpot = eval(neigval)
    rgxXpot = eval(1)
    rgxIpot = eval(3)
    rgxIIpot = eval(5)
```

```
        do i = 1,ncluster+1   !* Forces on rare gases and halogen
*                           Using Central difference approximation
          rxi = rx(i)
          ryi = ry(i)
          rzi = rz(i)

          rx(i) = rxi + delta/2.0d0
          call porb(rx,ry,rz,eval)
          evalplus = eval(neigval)
          rx(i) = rxi - delta/2.0d0
          call porb(rx,ry,rz,eval)
          fx(i) = fx(i) - (evalplus-eval(neigval))/delta
          rx(i) = rxi

          ry(i) = ryi + delta/2.0d0
          call porb(rx,ry,rz,eval)
          evalplus = eval(neigval)
          ry(i) = ryi - delta/2.0d0
          call porb(rx,ry,rz,eval)
          fy(i) = fy(i) - (evalplus-eval(neigval))/delta
          ry(i) = ryi

          rz(i) = rzi + delta/2.0d0
          call porb(rx,ry,rz,eval)
          evalplus = eval(neigval)
          rz(i) = rzi - delta/2.0d0
          call porb(rx,ry,rz,eval)
          fz(i) = fz(i) - (evalplus-eval(neigval))/delta
          rz(i) = rzi

        enddo

        poten = poten + rgxpot

      elseif (nhalpot.eq.4) then    !* Neutral, calculated analytically

        call porba(rx,ry,rz,evala)
        rgxpot = evala(neigval)
        rgxXpot = evala(1)
        rgxIpot = evala(2)
        rgxIIpot = evala(3)

        if (iaxtel.eq.1) then
          call vat(c9at,rx,ry,rz,atpot)
        else
          atpot = 0.
        endif
*
*       Forces calculated numerically
*
        do i = 1,ncluster+1   !* Forces on rare gases and halogen
          rxi = rx(i)
          ryi = ry(i)
          rzi = rz(i)
          rx(i) = rxi + delta
          call porba(rx,ry,rz,evala)
          if (iaxtel.eq.1) then
```

```
            call vat(c9at,rx,ry,rz,atpotx)
          else
            atpotx = 0.
          endif
          fx(i) = fx(i) - (evala(neigval)+atpotx-rgxpot
     &                     -atpot)/delta
          rx(i) = rxi
          ry(i) = ryi + delta
          call porba(rx,ry,rz,evala)
          if (iaxtel.eq.1) then
            call vat(c9at,rx,ry,rz,atpoty)
          else
            atpoty = 0.
          endif
          fy(i) = fy(i) - (evala(neigval)+atpoty-rgxpot
     &                     -atpot)/delta
          ry(i) = ryi
          rz(i) = rzi + delta
          call porba(rx,ry,rz,evala)
          if (iaxtel.eq.1) then
            call vat(c9at,rx,ry,rz,atpotz)
          else
            atpotz = 0.
          endif
          fz(i) = fz(i) - (evala(neigval)+atpotz-rgxpot
     &                     -atpot)/delta
          rz(i) = rzi
        enddo
        poten = poten + rgxpot + atpot

      endif
c
c
      icount = 0
c
      do 1000 n = 1,6*ncluster-5,6
c
        icount = icount + 1
        force(n) = y(n+3)/rmc
        force(n+1) = y(n+4)/rmc
        force(n+2) = y(n+5)/rmc
        force(n+3) = fx(icount)
        force(n+4) = fy(icount)
        force(n+5) = fz(icount)
c
 1000 continue
c
      icount = icount + 1
c
      n = 6*(ncluster+1) - 5
c
      force(n) = y(n+3)/rm1
      force(n+1) = y(n+4)/rm1
      force(n+2) = y(n+5)/rm1
      force(n+3) = fx(icount)
      force(n+4) = fy(icount)
      force(n+5) = fz(icount)
```

```
c
      RETURN
      END
c
      subroutine kinetic(y,ekin)
      include 'param.file'
c
      dimension y(neqmc)
      ekin = 0.0d0
c
      do 100 n = 1,6*ncluster-5,6
      ekin = ekin
     . + (y(n+3)**2 + y(n+4)**2 + y(n+5)**2)/(2.0d0*rmc)
 100  continue

      n = 6*(ncluster+1) - 5
      ekin = ekin
     .       + (y(n+3)**2 + y(n+4)**2 + y(n+5)**2)/(2.0d0*rm1)
c
      return
      end
```

C5.5. File "initzangdiat.f"

```
*==========================================
      subroutine initpos(y0,dseed)
*==========================================
*
*  Generate random initial positions
*
      include 'param.file'
      common/box/boxsize,cutoff
      dimension y0(neqc),x(ncl+nmolec),y(ncl+nmolec),z(ncl+nmolec),
     &          dist(ncl+1,ncl+1)
      logical tooclose

10    do n=1,ncluster+1
        call ggubs(dseed,1,r)
        x(n)=boxsize*r
        call ggubs(dseed,1,r)
        y(n)=boxsize*r
        call ggubs(dseed,1,r)
        z(n)=boxsize*r
      enddo
*
*  Check that no atoms are too close together
*
      tooclose=.false.
      do n=1,ncluster+1
        do m=n+1,ncluster+1
          dist(n,m)=sqrt((x(n)-x(m))**2+(y(n)-y(m))**2+(z(n)-z(m))**2)
          if (dist(n,m).lt.cutoff) then
            tooclose=.true.
          endif
        enddo
```

```
      enddo

      if (tooclose) go to 10

      do n=1,ncluster+1
        i = 6*(n-1)
        y0(i+1)=x(n)
        y0(i+2)=y(n)
        y0(i+3)=z(n)
      enddo

      return
      end


*=======================================================================
c
      subroutine initzangdiat(y0,dseed,escale)
      include 'param.file'
c
      dimension y0(neqc)
      dimension px(3000),py(3000),pz(3000)
      dimension x(3000),y(3000),z(3000)
      dimension rmass(3000)
      dimension r(1)
      dimension rmom(3,3),rinv(3,3)
c
      rmtot = rmc*ncluster + rm1
c
      do 150 n = 1,ncluster
      rmass(n) = rmc
150   continue
      rmass(ncluster+1) = rm1
c
c     zero out moment of inertia tensor
c
      do 10 i=1,3
      do 11 j=1,3
      rmom(i,j) = 0.0d0
11    continue
10    continue
c
c
c      read in initial configuration and calculate the momemt of
c      inertia tensor
c
      xcm = 0.0d0
      ycm = 0.0d0
      zcm = 0.0d0
c
      do 20 n = 1,ncluster + 1
c
      i = 6*(n-1)
      x(n) = y0(i+1)
      y(n) = y0(i+2)
      z(n) = y0(i+3)
      xcm = xcm + rmass(n)*x(n)
```

```
      ycm = ycm + rmass(n)*y(n)
      zcm = zcm + rmass(n)*z(n)
 20   continue
c
      xcm = xcm/rmtot
      ycm = ycm/rmtot
      zcm = zcm/rmtot
c
      do 21 n = 1,ncluster + 1
c
      x(n) = x(n) - xcm
      y(n) = y(n) - ycm
      z(n) = z(n) - zcm
c
      rmom(1,1) = rmom(1,1) + rmass(n)*(y(n)**2 + z(n)**2)
      rmom(1,2) = rmom(1,2) - rmass(n)*x(n)*y(n)
      rmom(1,3) = rmom(1,3) - rmass(n)*x(n)*z(n)
      rmom(2,2) = rmom(2,2) + rmass(n)*(x(n)**2 + z(n)**2)
      rmom(2,3) = rmom(2,3) - rmass(n)*y(n)*z(n)
      rmom(3,3) = rmom(3,3) + rmass(n)*(x(n)**2 + y(n)**2)
c
 21   continue
c
      rmom(2,1) = rmom(1,2)
      rmom(3,1) = rmom(1,3)
      rmom(3,2) = rmom(2,3)
c
c      invert the moment of inertia tensor
c
      call invert(rmom,rinv)
c
c     calculate random momenta and subtract CM momentum
c
c
      pxtot = 0.0d0
      pytot = 0.0d0
      pztot = 0.0d0
c
      do 50 n = 1,ncluster + 1
      call ggubs(dseed,1,r)
      px(n) = 2.0d0*(r(1) - 0.5d0)
      call ggubs(dseed,1,r)
      py(n) = 2.0d0*(r(1) - 0.5d0)
      call ggubs(dseed,1,r)
      pz(n) = 2.0d0*(r(1) - 0.5d0)
      pxtot = pxtot + px(n)
      pytot = pytot + py(n)
      pztot = pztot + pz(n)
 50   continue
      do 55 n = 1,ncluster + 1
      px(n) = px(n) - rmass(n)*pxtot/rmtot
      py(n) = py(n) - rmass(n)*pytot/rmtot
      pz(n) = pz(n) - rmass(n)*pztot/rmtot
55    continue
c
c      calculate angular momenta and angular velocity
c
```

```
      angx = 0.0d0
      angy = 0.0d0
      angz = 0.0d0
c
      do 57 n = 1,ncluster + 1
c
      angx = angx + y(n)*pz(n) - z(n)*py(n)
      angy = angy + z(n)*px(n) - x(n)*pz(n)
      angz = angz + x(n)*py(n) - y(n)*px(n)
c
57    continue
c
      wx = rinv(1,1)*angx + rinv(1,2)*angy + rinv(1,3)*angz
      wy = rinv(2,1)*angx + rinv(2,2)*angy + rinv(2,3)*angz
      wz = rinv(3,1)*angx + rinv(3,2)*angy + rinv(3,3)*angz
c
c      adjust particle momenta to remove overall angular momentum
c      also, calculate the total kinetic energy
c
      ekin = 0.0d0
c
      do 60 n = 1,ncluster + 1
c
      dpx = rmass(n)*(wy*z(n) - wz*y(n))
      dpy = rmass(n)*(wz*x(n) - wx*z(n))
      dpz = rmass(n)*(wx*y(n) - wy*x(n))
c
      px(n) = px(n) - dpx
      py(n) = py(n) - dpy
      pz(n) = pz(n) - dpz
c
      ekin = ekin + (px(n)**2 + py(n)**2
     . + pz(n)**2)/(2.0d0*rmass(n))
c
60    continue
c
      angx = 0.0d0
      angy = 0.0d0
      angz = 0.0d0
      pxtot = 0.0d0
      pytot = 0.0d0
      pztot = 0.0d0
c

      do 75 n = 1,ncluster + 1
c
      pxtot = pxtot + px(n)
      pytot = pytot + py(n)
      pztot = pztot + pz(n)
      angx = angx + y(n)*pz(n) - z(n)*py(n)
      angy = angy + z(n)*px(n) - x(n)*pz(n)
      angz = angz + x(n)*py(n) - y(n)*px(n)
c
75    continue
c
      fac = dsqrt(escale/ekin)
c
```

```
      pxf = 0.0d0
      pyf = 0.0d0
      pzf = 0.0d0
c
      do 70 n = 1,ncluster + 1
      px(n) = px(n)*fac
      py(n) = py(n)*fac
      pz(n) = pz(n)*fac
      pxf = pxf + px(n)
      pyf = pyf + py(n)
      pzf = pzf + pz(n)
c
70    continue
      do 100 n = 1,ncluster + 1
      i = 6*(n-1)
      y0(i+1) = x(n)
      y0(i+2) = y(n)
      y0(i+3) = z(n)
      y0(i+4) = px(n)
      y0(i+5) = py(n)
      y0(i+6) = pz(n)
 100  continue
c
      return
      end
c
c
      subroutine invert(rmom,rinv)
      implicit real*8(a-h,o-z)
      dimension rmom(3,3), rinv(3,3),cf(3,3),rident(3,3)
c
c      inverts the three by three rmom matrix  must be symmetric!
c
      cf(1,1) = rmom(2,2)*rmom(3,3) - rmom(2,3)*rmom(3,2)
      cf(1,2) = rmom(2,1)*rmom(3,3) - rmom(2,3)*rmom(3,1)
      cf(1,3) = rmom(2,1)*rmom(3,2) - rmom(2,2)*rmom(3,1)
      cf(2,1) = cf(1,2)
      cf(2,2) = rmom(1,1)*rmom(3,3) - rmom(1,3)*rmom(3,1)
      cf(2,3) = rmom(1,1)*rmom(3,2) - rmom(1,2)*rmom(3,1)
      cf(3,1) = cf(1,3)
      cf(3,2) = cf(2,3)
      cf(3,3) = rmom(1,1)*rmom(2,2) - rmom(1,2)*rmom(2,1)
c
      det = rmom(1,1)*cf(1,1) - rmom(1,2)*cf(1,2) + rmom(1,3)*cf(1,3)
c
c
      do 10 i=1,3
      do 20 j=1,3
      ifac = (-1)**(i+j)
      rinv(i,j) = dfloat(ifac)*cf(j,i)/det
20    continue
10    continue
c
c     check if this is the inverse
c
      do 30 i = 1,3
      do 40 j = 1,3
```

```
      rident(i,j) = 0.0d0
      do 50 k = 1,3
      rident(i,j) = rident(i,j) + rmom(i,k)*rinv(k,j)
50    continue
40    continue
30    continue
      do 60 i = 1,3
c      write(*,*)(rident(i,j),j=1,3)
60    continue
      return
      end

C   IMSL ROUTINE NAME   - GGUBS
C
C-----------------------------------------------------------------------
C
C   COMPUTER            - IBM77/SINGLE
C
C   LATEST REVISION     - JUNE 1, 1980
C
C   PURPOSE             - BASIC UNIFORM (0,1) PSEUDO-RANDOM NUMBER
C                           GENERATOR
C
C   USAGE               - CALL GGUBS (DSEED,NR,R)
C
C   ARGUMENTS    DSEED  - INPUT/OUTPUT DOUBLE PRECISION VARIABLE
C                           ASSIGNED AN INTEGER VALUE IN THE
C                           EXCLUSIVE RANGE (1.D0, 2147483647.D0).
C                           DSEED IS REPLACED BY A NEW VALUE TO BE
C                           USED IN A SUBSEQUENT CALL.
C                NR     - INPUT NUMBER OF DEVIATES TO BE GENERATED.
C                R      - OUTPUT VECTOR OF LENGTH NR CONTAINING THE
C                           PSEUDO-RANDOM UNIFORM (0,1) DEVIATES
C
C   PRECISION/HARDWARE  - SINGLE/ALL
C
C   REQD. IMSL ROUTINES - NONE REQUIRED
C
C   NOTATION            - INFORMATION ON SPECIAL NOTATION AND
C                           CONVENTIONS IS AVAILABLE IN THE MANUAL
C                           INTRODUCTION OR THROUGH IMSL ROUTINE UHELP
C
C   COPYRIGHT           - 1980 BY IMSL, INC. ALL RIGHTS RESERVED.
C
C   WARRANTY            - IMSL WARRANTS ONLY THAT IMSL TESTING HAS BEEN
C                           APPLIED TO THIS CODE. NO OTHER WARRANTY,
C                           EXPRESSED OR IMPLIED, IS APPLICABLE.
C
C-----------------------------------------------------------------------
C
      SUBROUTINE GGUBS (DSEED,NR,R)
C                                  SPECIFICATIONS FOR ARGUMENTS
c     note...I have changed this to be real*8 for all vars.  CCM
1/28/90
      INTEGER             NR
      REAL*8           r(NR),dseed
C                                  SPECIFICATIONS FOR LOCAL VARIABLES
```

```
      real*8   D2P31M,D2P31
C                                   D2P31M=(2**31) - 1
C                                   D2P31 =(2**31)(OR AN ADJUSTED VALUE)
      DATA               D2P31M/2147483647.D0/
      DATA               D2P31/2147483648.D0/
C                                   FIRST EXECUTABLE STATEMENT
      DO 5 I=1,NR
         DSEED = DMOD(16807.D0*DSEED,D2P31M)
    5 R(I) = DSEED / D2P31
      RETURN
      END
```

C5.6. File "thermgen.f"

```
      subroutine thermgen(y0,nstep,nskip,ntconst,h,ekintarget)
      include 'param.file'

      DIMENSION Y(neqc),D1Y(neqc),D2Y(neqc),D3Y(neqc),D4Y(neqc),
     .ZY(neqc),Y0(neqc),Y1(neqc),Y2(neqc),Y3(neqc),YH(neqc),Y3P(neqc),
     .F0(neqc),F1(neqc),F2(neqc),F3(neqc),FH(neqc),ZF(neqc)
c
C-------------------------------------------------------------------------
--------
C     DATA FOR INTEGRATOR ALGORITHMS
C-------------------------------------------------------------------------
--------
C     DATA FOR RUNGE KUTTA INTEGRATION
      B11= 1.0D0/3.0D0
      B21=-1.0D0/3.0D0
      B22= 1.0D0
      B31= 1.0D0
      B32=-1.0D0
      B33= 1.0D0
       W1= 1.0D0/8.0D0
       W2= 3.0D0/8.0D0
       W3= 3.0D0/8.0D0
       W4= 1.0D0/8.0D0
C
C     DATA FOR HYBRID GEAR ROUTINE
      GA02= 153.0D0/128.0D0
      GA01= 25.0D0/16.0D0
      GA00=-225.0D0/128.0D0
      GB02= 45.0D0/128.0D0
      GB01= 75.0D0/32.0D0
      GB00= 225.0D0/128.0D0
C
C     ALPHA1=-0.5 FOR STABILITY
C
      GA12= (15.0D0/16.0D0)-0.5D0*(29.0D0/32.0D0)
      GA11= (-1.0D0)-0.5D0*(1.0D0)
      GA10= (17.0D0/16.0D0)-0.5D0*(-61.0D0/32.0D0)
      GB12= (5.0D0/16.0D0)-0.5D0*(43.0D0/160.0D0)
      GB11= (11.0D0/12.0D0)-0.5D0*(41.0D0/24.0D0)
      GB10= (-11.0D0/16.0D0)-0.5D0*(31.0D0/32.0D0)
      GG10= (4.0D0/3.0D0)-0.5D0*(-2.0D0/15.0D0)
```

```
C
C     ALPHA2=1.0, BETA=9/56
C
      BETA= 9.0D0/56.0D0
      GA22= 1.5D0*(29.0D0/32.0D0)+BETA*(-45.0D0/4.0D0)
      GA21= 1.5D0*(1.0D0)+BETA*(0.0D0)
      GA20= 1.5D0*(-61.0D0/32.0D0)+BETA*(45.0D0/4.0D0)
      GB22= 1.5D0*(43.0D0/160.0D0)+BETA*(-71.0D0/20.0D0)
      GB21= 1.5D0*(41.0D0/24.0D0)+BETA*(-16.0D0)
      GB20= 1.5D0*(31.0D0/32.0D0)+BETA*(-3.0D0/4.0D0)
      GG20= 1.5D0*(-2.0D0/15.0D0)+BETA*(-16.0D0/5.0D0)
      GG21= BETA*1.0D0
C
      open(7, file='therm.out',status='unknown')
c
      call deriv(f0,y0)
c
      call kinetic(y0,ekin)
      energy = ekin + poten
      write(*,*)'initial poten = ',poten*harev/100.,' eV'
      WRITE(*,*)'initial energy =',energy*harev/100.,' eV'
      write(*,*)
      write(*,667)
667   format(5x,'Time (ps)',8x,'fac^2',10x,'Temp. (K)',7x,'Target
Temp.',
     &  3x,'Avg. Poten (eV)')
      therm = 2.0d0*ekin/(3.0d0*(ncluster+nmolec))
      therm = therm/0.0003167d0
      write(7,*)0.0d0,therm
C
C-------------------------------------------------------------------------
--------
C     SIXTEEN STEP RUNGE KUTTA INTEGRATION TO START
C-------------------------------------------------------------------------
--------
C
      DH=H/8.0D0
      ICOUNT=0
      DO 99 I=1,NEQ
      Y(I)=Y0(I)
99    CONTINUE
C
      CALL DERIV(F0,Y)
C
      DO 1000 ISTEP=1,16
C
      ICOUNT=ICOUNT+1
C
      CALL DERIV(ZF,Y)
C
      DO 100 I=1,NEQ
      D1Y(I)=DH*ZF(I)
100   CONTINUE
C
      DO 200 I=1,NEQ
      ZY(I)=Y(I)+B11*D1Y(I)
200   CONTINUE
```

```
C
      CALL DERIV(ZF,ZY)
C
      DO 300 I=1,NEQ
      D2Y(I)=DH*ZF(I)
300   CONTINUE
C
      DO 400 I=1,NEQ
      ZY(I)=Y(I)+B21*D1Y(I)+B22*D2Y(I)
400   CONTINUE
C
      CALL DERIV(ZF,ZY)
C
      DO 500 I=1,NEQ
      D3Y(I)=DH*ZF(I)
500   CONTINUE
C
      DO 600 I=1,NEQ
      ZY(I)=Y(I)+B31*D1Y(I)+B32*D2Y(I)+B33*D3Y(I)
600   CONTINUE
C
      CALL DERIV(ZF,ZY)
C
      DO 700 I=1,NEQ
      D4Y(I)=DH*ZF(I)
700   CONTINUE
C
      DO 800 I=1,NEQ
      Y(I)=Y(I)+W1*D1Y(I)+W2*D2Y(I)+W3*D3Y(I)+W4*D4Y(I)
800   CONTINUE
C
      IF(ICOUNT.EQ.8)THEN
      CALL DERIV(F1,Y)
      DO 810 I=1,NEQ
      Y1(I)=Y(I)
810   CONTINUE
      ELSE IF(ICOUNT.EQ.16)THEN
      CALL DERIV(F2,Y)
      DO 820 I=1,NEQ
      Y2(I)=Y(I)
820   CONTINUE
      END IF
C
1000  CONTINUE
C
C
C-------------------------------------------------------------------------
--------
C     ENTER MAIN INTEGRATION LOOP
C-------------------------------------------------------------------------
--------
      ICOUNT=2
      itime = 2
      iave = 0
      ekinave = 0.0d0
      potave = 0.0d0
c
```

```
      DO 2000 ISTEP=1,NSTEP
      ICOUNT=ICOUNT+1
      itime = itime + 1
      iave = iave + 1
C
      DO 1100 I=1,NEQ
      J=I
      YH(I)=GA02*Y0(I)+GA01*Y1(I)+GA00*Y2(I)+
     .H*(GB02*F0(I)+GB01*F1(I)+GB00*F2(I))
1100  CONTINUE
C
      CALL DERIV(FH,YH)
C
C     CALCULATE PREDICTED ARRAY Y3P
C
      DO 1300 I=1,NEQ
      Y3P(I)=GA12*Y0(I)+GA11*Y1(I)+GA10*Y2(I)+
     .H*(GB12*F0(I)+GB11*F1(I)+GB10*F2(I))+
     .H*GG10*FH(I)
1300  CONTINUE
C
      CALL DERIV(F3,Y3P)
C
C     CALCULATE CORRECTED ARRAY Y3
C     SET LOCAL TRUNCATION ERROR ERRLOC EQUAL TO ZERO
      ERRLOC=0.0D0
C
      DO 1500 I=1,NEQ
      Q=GA22*Y0(I)+GA21*Y1(I)+GA20*Y2(I)+
     .H*(GB22*F0(I)+GB21*F1(I)+GB20*F2(I))+
     .H*(GG20*FH(I)+GG21*F3(I))
c
      Y3(I)=Y3P(I)+Q
1500  CONTINUE
*
*     Recalculate F3 with corrected array Y3
*
      call deriv(F3,Y3)
      potave = potave + poten
C
C     RESET ARRAYS FOR NEXT STEP
C

      DO 1600 I=1,NEQ
      Y0(I)=Y1(I)
      Y1(I)=Y2(I)
      Y2(I)=Y3(I)
      F0(I)=F1(I)
      F1(I)=F2(I)
      F2(I)=F3(I)
1600  CONTINUE
      call kinetic(y2,ekin)

      ekinave = ekinave + ekin
      if(iave.eq.nskip) then
      ekinave = ekinave/(1.0d0*nskip)
      potave = potave/(1.0d0*nskip)
```

```
      therm = 2.0d0*ekinave/(3.0d0*(nmolec+ncluster))
      therm = therm/0.0003167d0
      write(7,*)h*itime/413.47,therm

      fac = dsqrt(1.+(dble(nskip)/dble(ntconst))
     &              *(ekintarget/therm-1.))

      ptave=potave*harev/100.
      write(*,2100)h*itime/413.46,fac**2,therm,ekintarget,ptave
      do 1601 n = 1,ncluster+nmolec
      do 1602 l=4,6
      y0(6*(n-1)+l) = fac*y0(6*(n-1)+l)
      y1(6*(n-1)+l) = fac*y1(6*(n-1)+l)
      y2(6*(n-1)+l) = fac*y2(6*(n-1)+l)
1602  continue
1601  continue
c
      iave = 0
      ekinave = 0.0d0
      potfin = potave
      tempfin = therm
      potave = 0.0d0
      endif
C
2000  CONTINUE
2100  format(5f16.8)
C
3000  close(7)
C
      do i = 1, neq
      y0(I)=y2(i)
      end do
c
      return
      END
```

C5.7. File "pforce.f"

```
*-----------------------------------------------------------------
      subroutine anmmsv(depth,rmin,beta1,
     &  beta2,x1,x2,c4,c6,rtin,pot,force)
c-----------------------------------------------------------------
c     Scaled MMSV for anion
*     In:     rtin  R in a0/10 units
*     Out:    pot   potential (hartree/100)
*             force (hartree/100)/(a0/10), negative when attractive
c
       implicit double precision(a-h,o-z)
       parameter(pi=3.14159265359d0,a0=0.529177249d0,
     &  harev=27.2113961d0)

       rt = a0*rtin/10.        !*  Convert to Angstroms
       xt = rt/rmin

        if (xt.le.1.) then
```

```
          e2 = exp(2.*beta1*(1.-xt))
          e1 = exp(beta1*(1.-xt))
          pot = depth*(e2-2.*e1)
          force = -(depth/rmin)*2.*beta1*(e1-e2)

        else if ((xt.gt.1).and.(xt.le.x1)) then

          e2 = exp(2.*beta2*(1.-xt))
          e1 = exp(beta2*(1.-xt))
          pot = depth*(e2-2.*e1)
          force = -(depth/rmin)*2.*beta2*(e1-e2)

        else if ((xt.gt.x1).and.(xt.lt.x2)) then

          p21=pi/(x2-x1)
          pt=p21*(xt-x1)
          swxt=0.5*(cos(pt)+1.)
          dswxt=-0.5*p21*sin(pt)/rmin

          e2 = exp(2.*beta2*(1.-xt))
          e1 = exp(beta2*(1.-xt))
          pmorse2 = e2-2.*e1
          dmorse2 = 2.*beta2*(e1-e2)/rmin

          pvdw = -(c4*rt**(-4)+c6*rt**(-6))
          dvdw = 4.*c4*rt**(-5)+6.*c6*rt**(-7)

          pot = depth*(swxt*pmorse2+(1.-swxt)*pvdw)
          force = -depth*(dswxt*(pmorse2-pvdw)+swxt*dmorse2
     &                    +(1.-swxt)*dvdw)

        else

          pot = -depth*(c4*rt**(-4)+c6*rt**(-6))
          force = -depth*(4.*c4*rt**(-5)+6.*c6*rt**(-7))

        end if
        pot = 100.*pot/harev
        force = 10.*force*a0/harev
      return
      end

*-----------------------------------------------------------------------
      subroutine hfd_b(depth,rmin,alpha,beta,
     &  acoef,c6,c8,c10,damp,rtin,pot,force)
*-----------------------------------------------------------------------
*
*  Hartree-Fock Dispersion-B Potential
*  Ref: Aziz & Slaman Mol. Phys. v.58, p.679
*
*  Input:   9 potential parameters
*           rtin  R in a0/10 units
*  Output:  pot   potential in hartree/100
*           force force in (hartree/100)/(a0/10)
*
      implicit double precision(a-h,o-z)
```

```
      parameter(a0=0.529177249d0,harev=27.2113961d0)

      rt = a0*rtin/10.        !*  Convert to Angstroms
      xt=rt/rmin
*
*  Dispersion damping function
*
      if (xt.lt.damp) then
        fdamp=exp(-(damp/xt-1)**2)
        ddamp=(2.*damp/xt**2)*(damp/xt-1.)*fdamp
      else
        fdamp=1.
        ddamp=0.
      endif
*
*  Repulsive part
*
      repul=acoef*exp(-alpha*xt+beta*xt*xt)
      drepul=(-alpha+2.*beta*xt)*repul
*
*  Dispersion terms
*
      disp=c6*xt**(-6)+c8*xt**(-8)+c10*xt**(-10)
      ddisp=-6.*c6*xt**(-7)-8.*c8*xt**(-9)-10.*c10*xt**(-11)

      pot=depth*(repul-fdamp*disp)
      force=-depth*(drepul-fdamp*ddisp-ddamp*disp)/rmin

      pot = 100.*pot/harev
      force = 10.*force*a0/harev

      return
      end

*=============================================================
      subroutine pfind(pol,x0,y0,z0,x1,y1,z1,x2,y2,z2,poten,
     &  f0x,f0y,f0z,f1x,f1y,f1z,f2x,f2y,f2z)
*=============================================================
*
*  Charge-Ind Dipole-Ind Dipole Three body potential & force
*
*  Input:   pol         Rare gas polarizibility (A^3)
*           x0,y0,z0    Halide cartesian coordinates (a0/10)
*           x1,y1,z1, x2,y2,z2      Rare gas coordinates (a0/10)
*
*  Output:  poten       Potential (hartree/100)
*           f0x,y,z           Force on halide (hartree/100)/(a0/10)
*           f1x,y,z;f2x,y,z   Force on rare gases
*
      implicit double precision(a-h,o-z)
      parameter(pi=3.14159265359d0,a0=0.529177249d0,
     &  harev=27.2113961d0)

      const = (pol/(a0**3)*1.d3)**2


      r10sqr = (x1-x0)**2+(y1-y0)**2+(z1-z0)**2
```

```
      r20sqr = (x2-x0)**2+(y2-y0)**2+(z2-z0)**2
      r12sqr = (x1-x2)**2+(y1-y2)**2+(z1-z2)**2

      dot1020 = (x1-x0)*(x2-x0)+(y1-y0)*(y2-y0)+(z1-z0)*(z2-z0)
      dot1012 = (x1-x0)*(x1-x2)+(y1-y0)*(y1-y2)+(z1-z0)*(z1-z2)
      dot1220 = (x1-x2)*(x2-x0)+(y1-y2)*(y2-y0)+(z1-z2)*(z2-z0)

      t1 = dsqrt(r10sqr)
      t2 = r10sqr**2
      t6 = dsqrt(r12sqr)
      t7 = r12sqr**2
      t10 = dsqrt(r20sqr)
      t11 = r20sqr**2
      poten = const*t1/t2*t6/t7*t10/t11*(dot1020-
     & 3*dot1012*dot1220/r12sqr)
      poten = poten*1.d3

*
* dv/dx0
*
      t1 = dsqrt(r10sqr)
      t2 = r10sqr**2
      t7 = dsqrt(r12sqr)
      t8 = r12sqr**2
      t10 = t7/t8
      t12 = dsqrt(r20sqr)
      t13 = r20sqr**2
      t15 = t12/t13
      t20 = 1/r12sqr
      t22 = dot1020-3*dot1012*dot1220*t20
      t29 = const*t1/t2
      t39 = -x1+x2
      t47 = -3.D0/2.D0*const*t1/t2/r10sqr*t10*t15*t22*(-2*x1+2*x0)-
3.D0/
     #2.D0*t29*t10*t12/t13/r20sqr*t22*(-2*x2+2*x0)+t29*t10*t15*(-
x2+2*x0
     #-x1-3*t39*dot1220*t20-3*dot1012*t39*t20)
      f0x = -t47*1.d3
*
* dv/dy0
*
      t26 = 1/r12sqr
      t28 = dot1020-3*dot1012*dot1220*t26
      t35 = const*t1/t2
      t45 = -y1+y2
      t53 = -3.D0/2.D0*const*t1/t2/r10sqr*t10*t15*t28*(-2*y1+2*y0)-
3.D0/
     #2.D0*t35*t10*t12/t13/r20sqr*t28*(-2*y2+2*y0)+t35*t10*t15*(-
y2+2*y0
     #-y1-3*t45*dot1220*t26-3*dot1012*t45*t26)
      f0y = -t53*1.d3
*
* dv/dz0
*
      t45 = -z1+z2
      t53 = -3.D0/2.D0*const*t1/t2/r10sqr*t10*t15*t28*(-2*z1+2*z0)-
3.D0/
```

```
     #2.D0*t35*t10*t12/t13/r20sqr*t28*(-2*z2+2*z0)+t35*t10*t15*(-
z2+2*z0
     #-z1-3*t45*dot1220*t26-3*dot1012*t45*t26)
      f0z = -t53*1.d3
*
* dv/dx1
*
      t9 = 1/t8
      t10 = t7*t9
      t17 = x2-x0
      t25 = dot1012*dot1220
      t29 = t15*((x1-x0)*t17+(y1-y0)*(y2-y0)+(z1-z0)*(z2-z0)-3*t25*t26)
      t40 = 2*x1-2*x2
      t54 = -3.D0/2.D0*const*t1/t2/r10sqr*t10*t29*(2*x1-2*x0)-
3.D0/2.D0*
     #t35*t7/t8/r12sqr*t29*t40+t35*t10*t15*(x2-x0-3*(2*x1-x2-
x0)*dot1220
     #*t26-3*dot1012*t17*t26+3*t25*t9*t40)
      f1x = -t54*1.d3
*
*  dv/dy1
*
      t20 = y2-y0
      t29 = t15*((x1-x0)*(x2-x0)+(y1-y0)*t20+(z1-z0)*(z2-z0)-3*t25*t26)
      t40 = 2*y1-2*y2
      t54 = -3.D0/2.D0*const*t1/t2/r10sqr*t10*t29*(2*y1-2*y0)-
3.D0/2.D0*
     #t35*t7/t8/r12sqr*t29*t40+t35*t10*t15*(y2-y0-3*(2*y1-y2-
y0)*dot1220
     #*t26-3*dot1012*t20*t26+3*t25*t9*t40)
      f1y = -t54*1.d3
*
*  dv/dz1
*
      t23 = z2-z0
      t29 = t15*((x1-x0)*(x2-x0)+(y1-y0)*(y2-y0)+(z1-z0)*t23-3*t25*t26)
      t40 = 2*z1-2*z2
      t54 = -3.D0/2.D0*const*t1/t2/r10sqr*t10*t29*(2*z1-2*z0)-
3.D0/2.D0*
     #t35*t7/t8/r12sqr*t29*t40+t35*t10*t15*(z2-z0-3*(2*z1-z2-
z0)*dot1220
     #*t26-3*dot1012*t23*t26+3*t25*t9*t40)
      f1z = -t54*1.d3
*
*  dv/dx2
*
      t5 = const*t1/t2
      t6 = dsqrt(r12sqr)
      t7 = r12sqr**2
      t28 = dot1020-3*t25*t26
      t30 = -2*x1+2*x2
      t33 = 1/t7
      t34 = t6*t33
      t55 = -3.D0/2.D0*t5*t6/t7/r12sqr*t15*t28*t30-
3.D0/2.D0*t5*t34*t12/
     #t13/r20sqr*t28*(2*x2-2*x0)+t5*t34*t15*(x1-x0-3*(-
x1+x0)*dot1220*t2
```

```
     #6-3*dot1012*(-2*x2+x0+x1)*t26+3*t25*t33*t30)
      f2x = -t55*1.d3
*
*  dv/dy2
*
      t30 = -2*y1+2*y2
      t55 = -3.D0/2.D0*t5*t6/t7/r12sqr*t15*t28*t30-
3.D0/2.D0*t5*t34*t12/
     #t13/r20sqr*t28*(2*y2-2*y0)+t5*t34*t15*(y1-y0-3*(-
y1+y0)*dot1220*t2
     #6-3*dot1012*(-2*y2+y0+y1)*t26+3*t25*t33*t30)
      f2y = -t55*1.d3
*
*  dv/dz2
*
      t30 = -2*z1+2*z2
      t55 = -3.D0/2.D0*t5*t6/t7/r12sqr*t15*t28*t30-
3.D0/2.D0*t5*t34*t12/
     #t13/r20sqr*t28*(2*z2-2*z0)+t5*t34*t15*(z1-z0-3*(-
z1+z0)*dot1220*t2
     #6-3*dot1012*(-2*z2+z0+z1)*t26+3*t25*t33*t30)
      f2z = -t55*1.d3

      return
      end


*=====================================================================
      subroutine pfexq(beta,cutoff,x0,y0,z0,x1,y1,z1,x2,y2,z2,poten,
     &  f0x,f0y,f0z,f1x,f1y,f1z,f2x,f2y,f2z)
*=====================================================================
*
*  Exchange Quadrupole, potential & forces
*
*  Input:   beta        range parameter ((a0/10)^-1)
*           cutoff            (a0/10)
*           x0,y0,z0    Halide coords (a0/10)
*           x1,...            RG 1 coords
*           x2,...            RG 2 coords
*
*  Output:  poten       potential (hartree/100)
*           f0x,...           Forces (hartree/100)(a0/10)
*
      implicit double precision(a-h,o-z)
      parameter(pi=3.14159265359d0,a0=0.529177249d0,
     &  harev=27.2113961d0,evtocm=8065.5410d0)

      r12sqr = (x1-x2)**2+(y1-y2)**2+(z1-z2)**2
      if (r12sqr.gt.cutoff**2) then
        poten = 0.
        f0x = 0.
        f0y = 0.
        f0z = 0.
        f1x = 0.
        f1y = 0.
        f1z = 0.
        f2x = 0.
```

```
        f2y = 0.
        f2z = 0.
        return
       endif
       rcsqr = (x1/2.-x0+x2/2.)**2+(y1/2.-y0+y2/2.)**2
      &        +(z1/2.-z0+z2/2.)**2
       dotc12 = (x1/2-x0+x2/2)*(x1-x2)+(y1/2-y0+y2/2)*(y1-y2)
      &         +(z1/2-z0+z2/2)*(z1-z2)
       ex = exp(-r12sqr*beta**2/2.d0)
       quad = -r12sqr*ex/(1.d0-ex)/2
       rcthir = rcsqr**1.5d0
       vexq = -(quad/rcthir)*(3.d0*dotc12**2/rcsqr/r12sqr-1.d0)/2.d0

       poten = vexq*1.d3
*
*  F0x
*
       t1 = beta**2
       t3 = dexp(-r12sqr*t1/2.0d0)
       t4 = r12sqr*t3
       t6 = 1/(1-t3)
       t8 = dsqrt(rcsqr)
       t9 = rcsqr**2
       t13 = dotc12**2
       t14 = 1/rcsqr
       t16 = 1/r12sqr
       t20 = -x1+2*x0-x2
       t23 = 1/t9
       t36 = -3.D0/4.D0*t4*t6*t8/t9/rcsqr*(3.D0/2.D0*t13*t14*t16-
1.D0/2.D
     #0)*t20+t4*t6*t8*t23*(3*dotc12*t14*t16*(-x1+x2)-
3.D0/2.D0*t13*t23*t
     #16*t20)/2
       f0x = -t36*1.0d3
*
*  F0y
*
       t20 = -y1+2*y0-y2
       t36 = -3.D0/4.D0*t4*t6*t8/t9/rcsqr*(3.D0/2.D0*t13*t14*t16-
1.D0/2.D
     #0)*t20+t4*t6*t8*t23*(3*dotc12*t14*t16*(-y1+y2)-
3.D0/2.D0*t13*t23*t
     #16*t20)/2
       f0y = -t36*1.0d3
*
*  F0z
*
       t20 = -z1+2*z0-z2
       t36 = -3.D0/4.D0*t4*t6*t8/t9/rcsqr*(3.D0/2.D0*t13*t14*t16-
1.D0/2.D
     #0)*t20+t4*t6*t8*t23*(3*dotc12*t14*t16*(-z1+z2)-
3.D0/2.D0*t13*t23*t
     #16*t20)/2
       f0z = -t36*1.0d3
*
*  F1x
*
```

```
      t1 = 2*x1-2*x2
      t2 = beta**2
      t4 = dexp(-r12sqr*t2/2)
      t6 = 1-t4
      t7 = 1/t6
      t8 = dsqrt(rcsqr)
      t9 = rcsqr**2
      t10 = 1/t9
      t11 = t8*t10
      t12 = t7*t11
      t13 = dotc12**2
      t14 = 1/rcsqr
      t15 = t13*t14
      t16 = 1/r12sqr
      t18 = 3.D0/2.D0*t15*t16-1.D0/2.D0
      t24 = t11*t18
      t27 = t4**2
      t29 = t6**2
      t35 = r12sqr*t4
      t41 = x1/2-x0+x2/2
      t51 = r12sqr**2
      t58 = t1*t4*t12*t18/2-r12sqr*t1*t2*t4*t7*t24/4-
r12sqr*t27/t29*t24*
     #t1*t2/4-
3.D0/4.D0*t35*t7*t8/t9/rcsqr*t18*t41+t35*t12*(3*dotc12*t14
     #*t16*(x1-x0)-3.D0/2.D0*t13*t10*t16*t41-3.D0/2.D0*t15/t51*t1)/2
      f1x = -t58*1.0d3
*
*  F1y
*
      t1 = 2*y1-2*y2
      t41 = y1/2-y0+y2/2
      t58 = t1*t4*t12*t18/2-r12sqr*t1*t2*t4*t7*t24/4-
r12sqr*t27/t29*t24*
     #t1*t2/4-
3.D0/4.D0*t35*t7*t8/t9/rcsqr*t18*t41+t35*t12*(3*dotc12*t14
     #*t16*(y1-y0)-3.D0/2.D0*t13*t10*t16*t41-3.D0/2.D0*t15/t51*t1)/2
      f1y = -t58*1.0d3
*
*  F1z
*
      t1 = 2*z1-2*z2
      t41 = z1/2-z0+z2/2
      t58 = t1*t4*t12*t18/2-r12sqr*t1*t2*t4*t7*t24/4-
r12sqr*t27/t29*t24*
     #t1*t2/4-
3.D0/4.D0*t35*t7*t8/t9/rcsqr*t18*t41+t35*t12*(3*dotc12*t14
     #*t16*(z1-z0)-3.D0/2.D0*t13*t10*t16*t41-3.D0/2.D0*t15/t51*t1)/2
      f1z = -t58*1.0d3
*
*  F2x
*
      t1 = -2*x1+2*x2
      t41 = x1/2-x0+x2/2
      t58 = t1*t4*t12*t18/2-r12sqr*t1*t2*t4*t7*t24/4-
r12sqr*t27/t29*t24*
```

```
     #t1*t2/4-
3.D0/4.D0*t35*t7*t8/t9/rcsqr*t18*t41+t35*t12*(3*dotc12*t14
     #*t16*(-x2+x0)-3.D0/2.D0*t13*t10*t16*t41-3.D0/2.D0*t15/t51*t1)/2
      f2x = -t58*1.0d3
*
*  F2y
*
      t1 = -2*y1+2*y2
      t41 = y1/2-y0+y2/2
      t58 = t1*t4*t12*t18/2-r12sqr*t1*t2*t4*t7*t24/4-
r12sqr*t27/t29*t24*
     #t1*t2/4-
3.D0/4.D0*t35*t7*t8/t9/rcsqr*t18*t41+t35*t12*(3*dotc12*t14
     #*t16*(-y2+y0)-3.D0/2.D0*t13*t10*t16*t41-3.D0/2.D0*t15/t51*t1)/2
      f2y = -t58*1.0d3
*
*  F2z
*
      t1 = -2*z1+2*z2
      t41 = z1/2-z0+z2/2
      t58 = t1*t4*t12*t18/2-r12sqr*t1*t2*t4*t7*t24/4-
r12sqr*t27/t29*t24*
     #t1*t2/4-
3.D0/4.D0*t35*t7*t8/t9/rcsqr*t18*t41+t35*t12*(3*dotc12*t14
     #*t16*(-z2+z0)-3.D0/2.D0*t13*t10*t16*t41-3.D0/2.D0*t15/t51*t1)/2
      f2z = -t58*1.0d3
*
*
      return
      end



*=================================================================
      subroutine porb(qx,qy,qz,eval)
*=================================================================
*
*  Calculate p-orbital splitting, including SO coupling.
*  Ref.: Lawrence & Apkarian, JCP v.101, p.1820.
*  Note the matrix element <1/2,-1/2|V|3/2,1/2> is incorrect
*  in the paper.
*
*  Input:   qx(1..nrg+1)      1..nrg x positions of rare gas atoms
*                          nrg+1 x pos of halogen atom (a0/10)
*           qy(        )    y positions
*           qz(        )      z positions
*
*  Output:  eval(1..6)  Eigenvalues (hartree/100)
*
*  Uses:    poten2.f: poten
*
      include 'param.file'
      parameter(nb=2,np=2,x=1,y=2,z=3)
      common /neutral/ px(10),p1(10),p2(10),soconst
*
*  Note soconst must be in hartree/100
```

```
*
      integer ncluster,i,j,k,ierr
      double precision qx(ncl+1),qy(ncl+1),qz(ncl+1),
     &  px(10),p1(10),p2(10)
      double precision vr(-1:1,-1:1),vi(-1:1,-1:1),
     &  vjr(6,6),vji(6,6),eval(6),evecr(6,6),eveci(6,6),fv1(6),
     &  fv2(6),fm1(2,6)
      double precision rneummsv,rt,soconst
      double precision sr2,sr3,xhal,yhal,zhal,
     &  xk,yk,zk,v0k,v2k,c1,rsq,vx0,vi0,vii0

      sr2 = dsqrt(2.0d0)
      sr3 = dsqrt(3.0d0)
*
      xhal = qx(ncluster+1)    !* Halogen atom coordinates
      yhal = qy(ncluster+1)
      zhal = qz(ncluster+1)
*
*  Calculate matrix elements in spinless basis |lm>
*
      do i=-1,1
        do j=-1,1
          vr(i,j)=0.
          vi(i,j)=0.
        enddo
      enddo

      do k=1,ncluster
        xk = qx(k) - xhal
        yk = qy(k) - yhal
        zk = qz(k) - zhal

        rsq=xk**2+yk**2+zk**2
        rt = sqrt(rsq)

        vx0=rneummsv(px(1),px(2),px(3),px(4),px(5),px(6),px(7),
     &               px(8),rt)
        vi0=rneummsv(p1(1),p1(2),p1(3),p1(4),p1(5),p1(6),p1(7),
     &               p1(8),rt)
        vii0=rneummsv(p2(1),p2(2),p2(3),p2(4),p2(5),p2(6),p2(7),
     &               p2(8),rt)

        v0k = (vx0+vii0+vi0)/3.
        v2k = 5.*(vx0+vii0-2.*vi0)/3.      !* Assuming constant soconst

        c1 = (3.*zk**2-rsq)*v2k/rsq

        vr(0,0) = vr(0,0) + v0k + (1./5.)*c1

        vr(1,1) = vr(1,1) + v0k - (1./10.)*c1

        vr(1,0) = vr(1,0) -3.*zk*xk*v2k/(5.*sr2*rsq)
        vi(1,0) = vi(1,0) +3.*zk*yk*v2k/(5.*sr2*rsq)

        vr(1,-1) = vr(1,-1)-3.*(xk**2-yk**2)*v2k/(10.*rsq)
        vi(1,-1) = vi(1,-1)+3.*xk*yk*v2k/(5.*rsq)
```

```
      enddo

      vr(-1,-1) = vr(1,1)

      vr(-1,1) = vr(1,-1)
      vi(-1,1) = -vi(1,-1)

      vr(0,1) = vr(1,0)
      vi(0,1) = -vi(1,0)

      vr(0,-1) = -vr(1,0)
      vi(0,-1) = -vi(1,0)

      vr(-1,0) = -vr(1,0)
      vi(-1,0) = vi(1,0)
*
*  Calculate matrix elements in coupled basis |JM>
*
      vjr(1,1) = 2.*vr(1,1)/3.+vr(0,0)/3.+2.*soconst/3.
      vji(1,1) = 0.

      vjr(1,2) = 0.
      vji(1,2) = 0.

      vjr(1,3) = (sr2/3.)*(vr(1,1)-vr(0,0))
      vji(1,3) = 0.

      vjr(1,4) = (2.*vr(1,0)-vr(0,-1))/3.
      vji(1,4) = (2.*vi(1,0)-vi(0,-1))/3.

      vjr(1,5) = -vr(0,1)/sr3
      vji(1,5) = -vi(0,1)/sr3

      vjr(1,6) = vr(1,-1)*sr2/sr3
      vji(1,6) = vi(1,-1)*sr2/sr3

      vjr(2,1) = 0.
      vji(2,1) = 0.

      vjr(2,2) = vjr(1,1)
      vji(2,2) = 0.

      vjr(2,3) = (vr(0,1)-2.*vr(-1,0))/3.   !* Incorrect in Apkarian's
paper
      vji(2,3) = (vi(0,1)-2.*vi(-1,0))/3.   !*  "

      vjr(2,4) = (vr(0,0)-vr(-1,-1))*sr2/3.
      vji(2,4) = 0.

      vjr(2,5) = -vr(-1,1)*sr2/sr3
      vji(2,5) = -vi(-1,1)*sr2/sr3

      vjr(2,6) = vr(0,-1)/sr3
      vji(2,6) = vi(0,-1)/sr3

      vjr(3,1) = vjr(1,3)
      vji(3,1) = 0.
```

```
vjr(3,2) = vjr(2,3)
vji(3,2) = -vji(2,3)

vjr(3,3) = 2.*vr(0,0)/3.+vr(1,1)/3.-soconst/3.
vji(3,3) = 0.

vjr(3,4) = 0.
vji(3,4) = 0.

vjr(3,5) = vr(0,1)*sr2/sr3
vji(3,5) = vi(0,1)*sr2/sr3

vjr(3,6) = vr(1,-1)/sr3
vji(3,6) = vi(1,-1)/sr3

vjr(4,1) = vjr(1,4)
vji(4,1) = -vji(1,4)

vjr(4,2) = vjr(2,4)
vji(4,2) = 0.

vjr(4,3) = 0.
vji(4,3) = 0.

vjr(4,4) = vjr(3,3)
vji(4,4) = 0.

vjr(4,5) = vr(-1,1)/sr3
vji(4,5) = vi(-1,1)/sr3

vjr(4,6) = vr(0,-1)*sr2/sr3
vji(4,6) = vi(0,-1)*sr2/sr3

vjr(5,1) = vjr(1,5)
vji(5,1) = -vji(1,5)

vjr(5,2) = vjr(2,5)
vji(5,2) = -vji(2,5)

vjr(5,3) = vjr(3,5)
vji(5,3) = -vji(3,5)

vjr(5,4) = vjr(4,5)
vji(5,4) = -vji(4,5)

vjr(5,5) = vr(1,1)-soconst/3.
vji(5,5) = 0.

vjr(5,6) = 0.
vji(5,6) = 0.

vjr(6,1) = vjr(1,6)
vji(6,1) = -vji(1,6)

vjr(6,2) = vjr(2,6)
vji(6,2) = -vji(2,6)
```

```
      vjr(6,3) = vjr(3,6)
      vji(6,3) = -vji(3,6)

      vjr(6,4) = vjr(4,6)
      vji(6,4) = -vji(4,6)

      vjr(6,5) = 0.
      vji(6,5) = 0.

      vjr(6,6) = vjr(5,5)
      vji(6,6) = 0.
*
*
*
*  Find eigenvalues (complex hermitian routine from Eispack)
*
      call ch(6,6,vjr,vji,eval,1,evecr,eveci,fv1,fv2,fm1,ierr)
      if (ierr.ne.0) then
        write (*,*) 'Error in Eispack ch.f subroutine!'
        write (*,*) 'ierr=',ierr
      endif
      eval(1) = eval(1) + soconst/3.
      eval(2) = eval(2) + soconst/3.
      eval(3) = eval(3) + soconst/3.
      eval(4) = eval(4) + soconst/3.
      eval(5) = eval(5) - 2.*soconst/3.
      eval(6) = eval(6) - 2.*soconst/3.

      return
      end

c -----------------------------------------------------------------
      subroutine showarr3(ndim,a,nra,nca)
c
c     displays the matrix A
c
      implicit double precision (a-h,o-z)
      dimension a(ndim,ndim)

      do 10 ir = 1, nra
         write(*,100) (a(ir,ic), ic = 1,nca)
100      format(20(f15.8,1x),/)
10    continue
      return
      end

*-----------------------------------------------------------------
      double precision function rneummsv(depth2,rmin2,pmbeta12,
     &  pmbeta22,xrstar12,xrstar22,c6vdw2,c8vdw2,rtin)
c-----------------------------------------------------------------
*
c    Scaled MMSV
c    R in a0/10
c    Poten in hartree/100
c
      implicit double precision(a-h,o-z)
```

```
      parameter(pi=3.14159265359d0,a0=0.529177249d0,
     &  harev=27.2113961d0)

      rt=rtin*a0/10.d0  !* Convert a0/10 to Ang.
      xt=rt/rmin2
        if (xt.le.1.) then
        Tsum = depth2*(exp(2.*pmbeta12*(1.-xt))-2.*exp(
     &            pmbeta12*(1.-xt)))
        elseif ((xt.gt.1).and.(xt.le.xrstar12)) then
         Tsum = depth2*(exp(2.*pmbeta22*(1.-xt))-2.*exp(
     &            pmbeta22*(1.-xt)))
        elseif ((xt.gt.xrstar12).and.(xt.lt.xrstar22)) then
         swxt=0.5*(cos(pi*(xt-xrstar12)/(xrstar22-xrstar12))+1.)
         pmorse2=exp(2.*pmbeta22*(1.-xt))-2.*exp(
     &            pmbeta22*(1.-xt))
         pvdw=-1.*(c6vdw2*rt**(-6)+c8vdw2*rt**(-8))
         Tsum = depth2*(swxt*pmorse2+(1.-swxt)*pvdw)
        else
         Tsum = -1.*depth2*(c6vdw2*rt**(-6)+c8vdw2*rt**(-8))
        endif
      rneummsv=(Tsum/harev)*100.d0  !* Convert eV to hartree/100

      return
      end

*=======================================================================
===
      subroutine porba(qx,qy,qz,eval)
*=======================================================================
===
*
*  Calculate p-orbital states analytically
*
*  Input:   qx,qy,qz(nrg+1)         coordinates of rgs & halogen
(a0/10)
*  Output:  eval(1..3)        eigenvalues X, I, II (hartree/100)
*
      include 'param.file'
      common /neutral/ px(10),p1(10),p2(10),soconst
*
*  Note soconst must be in hartree/100
*
      double precision av,so,v0,kv,gv,fv,hv
      double precision qx(ncl+1),qy(ncl+1),qz(ncl+1),
     &  px(10),p1(10),p2(10),eval(3)
      double precision soconst
      double complex t60,t100,t124,t148,t151,t152,t153,t157,
     &  t158,t159,t160

      sr2=dsqrt(2.d0)

      xhal = qx(ncluster+1)   !* Halogen atom coordinates
      yhal = qy(ncluster+1)
      zhal = qz(ncluster+1)

      av = 0.d0
      kv = 0.d0
```

```
      gv = 0.d0
      fv = 0.d0
      hv = 0.d0
      v0 = 0.d0

      do k=1,ncluster
        xk = qx(k) - xhal
        yk = qy(k) - yhal
        zk = qz(k) - zhal
        rsq=xk**2+yk**2+zk**2

        rt = sqrt(rsq)
        vx0=rneummsv(px(1),px(2),px(3),px(4),px(5),px(6),px(7),
     &              px(8),rt)
        vi0=rneummsv(p1(1),p1(2),p1(3),p1(4),p1(5),p1(6),p1(7),
     &              p1(8),rt)
        vii0=rneummsv(p2(1),p2(2),p2(3),p2(4),p2(5),p2(6),p2(7),
     &              p2(8),rt)
        v0k = (vx0+vii0+vi0)/3.
        v0 = v0 + v0k
        v2k = 5.*(vx0+vii0-2.*vi0)/3.     !* Assuming constant soconst
        av = av + (3.*zk**2-rsq)*v2k/rsq
        fv = fv - 3.*zk*xk*v2k/(5.*sr2*rsq)
        gv = gv + 3.*zk*yk*v2k/(5.*sr2*rsq)
        hv = hv - 3.*(xk**2-yk**2)*v2k/(10.*rsq)
        kv = kv + 3.*xk*yk*v2k/(5.*rsq)
      enddo

      so = soconst
*
*  Maple generated code
*
* fortran(ev[1],optimized);
*
      t1 = av**2
      t2 = so**2
      t3 = kv**2
      t4 = gv**2
      t5 = fv**2
      t6 = hv**2
      t7 = v0**2
      t10 = t1*av
      t11 = t2*so
      t14 = hv*t4
      t16 = t6*av
      t19 = t4*av
      t20 = t3*av
      t21 = t5*av
      t27 = t3*t6
      t31 = t1*t3
      t33 = t1*t6
      t36 = t2**2
      t38 = t5*t6
      t42 = t5*fv
      t45 = t5*t3
      t51 = t5**2
      t55 = av*hv
```

```
      t60 = -360000*t27*t5-1620*hv*t10*t5-1800*t31*t2-1800*t33*t2-
27000*
     #t33*t4-300*t36*t1-2340000*t38*t4-
360000*t27*t4+3240000*hv*gv*kv*t4
     #2+2520000*t45*t4+60*t10*t11-5400*t5*t4*t1-27000*t38*t1-
162000*t51*
     #hv*av-27000*t45*t1+162000*t55*t45+1620*t10*t4*hv-2700*t51*t1
      t65 = t1**2
      t67 = fv*kv
      t68 = t4*gv
      t71 = t3**2
      t75 = av*gv
      t82 = t6*hv
      t95 = t2*t3
      t98 = t4**2
      t100 = 450000*t6*t51-120000*fv*gv*t11*kv-27*t2*t65-
324000*t67*t68*
     #av+5400*t1*t71+324000*fv*t3*kv*t75-3240000*fv*hv*t68*kv-3240*t67*
     #t10*gv+162000*t21*t82-324000*t42*gv*kv*av-180000*t71*t4-243*t65*t
     #6-3600*t1*t4*t2-3600*t1*t5*t2-20000*t36*t4-20000*t36*t5-
120000*t95
     #*t4-30000*t2*t71-120000*t2*t98
      t102 = t6**2
      t116 = t2*t6
      t124 = -30000*t2*t102-10000*t36*t3+324000*t67*t75*t6-
120000*t2*t51
     #-180000*t4*t102-360000*t3*t98+162000*t55*t98-90000*t71*t6-
90000*t3
     #*t102-180000*t71*t5-30000*t71*t3-243*t65*t3-
60000*t116*t3+6000*t19
     #*t11-180000*t5*t102-720000*t98*t5-720000*t4*t51-360000*t3*t51-
1200
     #00*t116*t4
      t148 = -240000*t2*t5*t4-10000*t36*t6-120000*t95*t5-
120000*t116*t5-
     #162000*t20*t14-162000*av*t82*t4-30000*t102*t6-240000*t51*t5+45000
     #0*t98*t6-240000*t98*t4+10800*t33*t3+6000*t21*t11-27000*t31*t4-
6000
     #0*t5*t11*hv+5400*t1*t102-2700*t98*t1+60000*t4*t11*hv-
6000*t16*t11-
     #6000*t20*t11
      t151 = sqrt(t60+t100+t124+t148)
      t152 = -(-3.D0/100.D0*t1-t2/3-t3-2*t4-2*t5-t6+3*t7)*v0/2+t10/1000
     #+t11/27-v0*t5-3.D0/200.D0*v0*t1+t14-t6*v0/2-t16/10-t3*v0/2-t4*
     #v0+t19/10-t20/10+t21/10-hv*t5-2*gv*kv*fv-v0*t2/6+3.D0/2.D0*t7*v0
     #+t151/900
      t153 = t152**(1.D0/3.D0)
      t158=(-t1/100-t2/9-t3/3-2.D0/3.D0*t4-2.D0/3.D0*t5-t6/3)/t153
      t157 = t153-t158+v0

c      write(*,*)'ev1',t157
c      write(*,*)'ev1',dble(t157)-2.*soconst/3.

      eval1 = dble(t157)
*
* hand generated code
*
```

```
      t158=(-t1/100-t2/9-t3/3-2.D0/3.D0*t4-2.D0/3.D0*t5-t6/3)/t153
      t159 = cmplx(0.d0,1.d0)*sqrt(3.d0)*(t153+t158)
      t160 = -t153/2.d0+t158/2.d0+v0
      eval2 = dble(t160+t159/2.d0)
      eval3 = dble(t160-t159/2.d0)

*
*  Sort eigenvalues in order X,I,II
*
      eval(1) = dmin1(eval1,eval2,eval3) + soconst/3.d0
      eval(3) = dmax1(eval1,eval2,eval3) - 2.d0*soconst/3.d0
      eval(2) = eval1+eval2+eval3-eval(1)-eval(3)

      return
      end

*=====================================================================
      subroutine veqd(betain,theta6,qx,qy,qz,dxin,dyin,dzin,dxout,
     & dyout,dzout,potout)
*=====================================================================
*
*  Exchange quadrupole-dipole & dispersion quadrupole
*  Ref: Ernesti & Hutson, Phys. Rev. A v.51,p.239
*
*  Input:   betain            Exchange quadrupole range parameter
(a0/10)^-1
*           theta6            quadrupole dispersion coefficient
(e*a0^8)
*           qx,qy,qz(ncl+1) Rg, halide coordinates (a0/10)
*           dxin,dyin,dzin(ncl+1) Dipoles input
*
*  Output:  potout            potential (hartree/100)
*           dxout,dyout,dzout Dipoles with exchange & disp
*                             contribution added
*
      include 'param.file'
      double precision betain,theta6
      double precision qx(ncl+1),qy(ncl+1),qz(ncl+1),dxin(ncl+1),
     & dyin(ncl+1),dzin(ncl+1),dxout(ncl+1),dyout(ncl+1),dzout(ncl+1),
     & dx(ncl+1),dy(ncl+1),dz(ncl+1),
     & rijx(ncl+1,ncl+1),rijy(ncl+1,ncl+1),rijz(ncl+1,ncl+1),
     & rij2(ncl+1,ncl+1),rij3(ncl+1,ncl+1),rij1(ncl+1,ncl+1)
*
*
      beta = betain*10.0d0
      nhal = ncluster+1
*
      do i = 1,ncluster       !* Relative Rg-Rg and Rg-X vectors
        do j = i+1,ncluster+1 !* (rij is vector from Rg j to Rg i)
          rijx(i,j) = (qx(i) - qx(j))/10.0d0    !* Convert to a0
          rijy(i,j) = (qy(i) - qy(j))/10.0d0
          rijz(i,j) = (qz(i) - qz(j))/10.0d0
          rijx(j,i) = -rijx(i,j)
          rijy(j,i) = -rijy(i,j)
          rijz(j,i) = -rijz(i,j)
          rij2(i,j) = rijx(i,j)**2+rijy(i,j)**2+rijz(i,j)**2
          rij1(i,j) = dsqrt(rij2(i,j))
```

```
            rij3(i,j) = rij1(i,j)**3
            rij1(j,i) = rij1(i,j)
            rij2(j,i) = rij2(i,j)
            rij3(j,i) = rij3(i,j)
          enddo
        enddo
*
* Zero out dipoles
*
        do i = 1,ncluster
          dx(i) = 0.0d0
          dy(i) = 0.0d0
          dz(i) = 0.0d0
        enddo
*
*  Add contribution to Rg dipoles from exchange quadrupole and
*  dispersion quadrupole between each pair of Rgs
*
        do i = 1,ncluster-1
          do j = i+1,ncluster
            ex = exp(-(beta**2)*rij2(i,j)/2.0d0)
            exquad = -rij2(i,j)*ex/(1.0d0-ex)/2
     &               +theta6/(rij3(i,j)**2)
            exdip = exquad/rij2(i,j)/2.0d0
            dx(i) = dx(i)+exdip*rijx(i,j)
            dy(i) = dy(i)+exdip*rijy(i,j)
            dz(i) = dz(i)+exdip*rijz(i,j)
            dx(j) = dx(j)+exdip*rijx(j,i)
            dy(j) = dy(j)+exdip*rijy(j,i)
            dz(j) = dz(j)+exdip*rijz(j,i)
          enddo
        enddo
*
*
*
*  Add to input dipoles
*
        do i = 1,ncluster
          dxout(i) = dxin(i)+dx(i)
          dyout(i) = dyin(i)+dy(i)
          dzout(i) = dzin(i)+dz(i)
        enddo
*
*  Compute charge-dipole interactions
*
        vcd = 0.0d0
        do i = 1,ncluster
          dotprod = dx(i)*rijx(nhal,i)+dy(i)*rijy(nhal,i)
     &              +dz(i)*rijz(nhal,i)
          vcd = vcd - dotprod/rij3(nhal,i)
        enddo

        potout = vcd*1.0d2       !* Convert to hartree/100

        return
        end
```

B2.2. Running the program

One runs the program in the usual Unix fashion, by piping in the input file. If one is calculating standard deviations as well as performing a fit it is usually desirable to run the program in the background, as in the following example using the input file "xebr_x_rfin" discussed above:

```
> nice +19 rfit < xebr_x_rfin > xebr_x_rfout &
>
```

The output file "xebr_x_rfout" then contains the optimized parameters and standard deviations, and the convoluted spectrum is saved to the file "xebr_x_fit" named in the input file.

B2.3. Outline of the program

The program "rfit" is contained in only one file, "rfit.f." There is no makefile; the program is recompiled with a command such as:

```
> f77 -O -o rfit rfit.f -C
```

The subroutines and functions used by the program are listed in Table B3.

Table B3. Subroutines and functions used by the rotational fitting program "rfit".

Subroutine or Function	Description
rfit	main program
optchi	general-purpose subroutine implementing the gradient optimization method
chisquar	calculates χ^2 for a given set of fitting parameters
rsticks	calculates a rotational stick spectrum given one or two vibrational stick spectra, and the fitting parameters.
be	function to calculate the rotational constant
boltz	calculates the Boltzmann factor for a given rotational temperature and anion rotational state
conzeke	convolutes the rotational stick spectrum with the ZEKE line shape
cp2	compares the experimental and convoluted spectra and calculates χ^2

B3. Source code for the rotational fitting program "rfit"

```
      program rfit


      implicit undefined(a-z)
*
      real frac,evtocm
      parameter (frac=0.1,evtocm=8065.541)
      real vstk(2,2,100)
      real temp,org,baseln
      real beneu,bean,reneu(2),rean
      real mass1,mass2,rmass
      integer ninfil,i,j,k,state(2),j,nvstk(2),rct
      integer mpts
      character*25 infile(2), outfile,especfil
      real relfcf(2),scoef(2)
      real fwhmcm,fwhmev,delta,deltawn
      real smooth(2,100000),espec(2,100000),espec2(2,100000),
     &  rstk(2,500000)
      integer nspec
      real dfwhmcm,dtemp,dscoef(2),drelfcf(2),dorg,dbaseln
      real mfwhmcm,mtemp,mscoef(2),mrelfcf(2),morg,mbaseln
      real xfwhmcm,xtemp,xscoef(2),xrelfcf(2),xorg,xbaseln
      real s1fwhm,s1temp,s1scoef(2),s1relfcf(2),s1org,s1baseln
      real s2fwhm,s2temp,s2scoef(2),s2relfcf(2),s2org,s2baseln
      logical ufwhm,utemp,uscoef(2),urelfcf(2),uorg,ubaseln
      real chisq,xinc,chiopt
      character*1 ans
      logical vscoef(2),vrelfcf(2),vtemp,vfwhm,vorg,vbaseln
      integer ncomp
      real a(30),da(30),ma(30),xa(30),aopt(30),abest(30)
      real s1a(30),s2a(30)
      logical va(30),ua(30),v1
      integer nparm,mode,tout
      logical pr
      real ncounts,chilast,slast,slope
*
      common /conv1/ fwhmcm,fwhmev,delta,deltawn
      common /conv2/ mpts
      common /func/ bean,beneu
      common /rvars/ mass1,mass2,rmass,rean,temp,org,baseln
      common /ivars/ rct,ninfil
      common/arrays/reneu(2),state(2),nvstk(2),
     &  relfcf(2),scoef(2)
      common /comp/ chisq,ncomp
      common /comp2/ ncounts
      common /opti/ nparm
      common /optr/ a(30),abest(30)
      common /sticks/ vstk(2,2,100),rstk(2,500000)
      common /spec1/ smooth(2,100000),espec(2,100000)
      common /spec2/ nspec
      common /pr1/ pr,tout
*-----------------------------------------------------------------
*  Read in constants and fitting parameters
```

```
        vpair = vpair + drg2/(2*rij2(nhal,i))

      enddo

      potout = (vpol-vpair)*1.0d2   !* Convert to hartree/100

      return

      end


*=====================================================================
      subroutine vindq(pqrg,pqx,qx,qy,qz,dx,dy,dz,dhalpair,
     &  drgpair,qind,viqrgd,viqq,viqxd,potout)
*=====================================================================
*
*  Calculate induced quadrupole-induced dipole & ind quad-ind quad
*  potentials
*
*  Input:   pqrg,pqx    Rare gas, halide quadrupole
*                       polarizabilities (a0^5)
*           qx,qy,qz(ncl+1)   Rg, halide coordinates (a0/10)
*           dx,dy,dz(ncl+1)   Induced dipoles from vindi (au)
*           dhalpair(ncl)     Induced halogen dipole in pair potens
*           drgpair(ncl)      Induced Rg dipole in pair potens
*
*  Output:  viqrgd            Ind Rg Q - Ind Rg dip poten (hart/100)
*           viqq         Ind Rg Q - Ind Rg Q (hart/100)
*           viqxd        Ind X dip - Ind Rg Q (3-body
contrib.)(hart/100)
*           potout             Total Potential (hartree/100)
*           qind(ncl+1) Quadrupoles on Rgs & halide (e*a0^2)
*
      include 'param.file'
*
      common /distances/ rijx(ncl+1,ncl+1),rijy(ncl+1,ncl+1),
     & rijz(ncl+1,ncl+1),rij2(ncl+1,ncl+1),rij3(ncl+1,ncl+1),
     & rij1(ncl+1,ncl+1),dip(ncl+1)
*
      double precision pqrg,pqx,potout
      double precision qx(ncl+1),qy(ncl+1),qz(ncl+1),dx(ncl+1),
     & dy(ncl+1),dz(ncl+1),rij1(ncl+1,ncl+1),
     & rijx(ncl+1,ncl+1),rijy(ncl+1,ncl+1),rijz(ncl+1,ncl+1),
     & rij2(ncl+1,ncl+1),rij3(ncl+1,ncl+1),qind(ncl+1),dip(ncl+1),
     & dhalpair(ncl),drgpair(ncl)
*
      nhal = ncluster+1
*
*  Compute quadrupoles on Rgs due to halide charge
*
      do i = 1,ncluster
        qind(i) = 2.0d0*pqrg/rij3(i,nhal)
      enddo
*
*  Compute ind quadrupole-ind Rg dipole energy, and ind q-ind q.
*
```

```
      viqrgd = 0.0d0
      viqq = 0.0d0
      do i = 1,ncluster       !* Quadrupole at i
        do j = 1,ncluster     !* Dipole at j

          if (i.ne.j) then    !* Q-dip: double counted

            dotq = rijx(nhal,i)*rijx(j,i) + rijy(nhal,i)*rijy(j,i)
     &             + rijz(nhal,i)*rijz(j,i)
            cthetaq = dotq/(rij1(nhal,i)*rij1(j,i))
            if (cthetaq.gt.1.0d0) then
              cthetaq = 1.0d0
            elseif (cthetaq.lt.-1.0d0) then
              cthetaq = -1.0d0
            endif
            thetaq = acos(cthetaq)  !* angle between Q (Rg-X), Rg-Rg

            dotd = dx(j)*rijx(i,j)+dy(j)*rijy(i,j)+dz(j)*rijz(i,j)
            cthetad = dotd/(rij1(i,j)*dip(j))
            if (cthetad.gt.1.0d0) then
              cthetad = 1.0d0
            elseif (cthetad.lt.-1.0d0) then
              cthetad = -1.0d0
            endif
            thetad = acos(cthetad)  !* angle between dip, Rg-Rg

            dothjij = rijx(i,j)*rijx(nhal,j) + rijy(i,j)*rijy(nhal,j)
     &                + rijz(i,j)*rijz(nhal,j)
            cchi = dothjij/(rij1(i,j)*rij1(nhal,j))
            rpar = cchi*rij1(nhal,j)
            rparx = rpar*rijx(i,j)/rij1(i,j)
            rpary = rpar*rijy(i,j)/rij1(i,j)
            rparz = rpar*rijz(i,j)/rij1(i,j)
            ux = rijx(nhal,j) - rparx           !* u=vector in Rhj,Rij
plane
            uy = rijy(nhal,j) - rpary           !* perpendicular to Rij
            uz = rijz(nhal,j) - rparz
            dpar = dip(j)*cthetad
            dparx = dpar*rijx(i,j)/rij1(i,j)
            dpary = dpar*rijy(i,j)/rij1(i,j)
            dparz = dpar*rijz(i,j)/rij1(i,j)

            vx = dx(j) - dparx                  !* v=vector in
dipole,Rij plane
            vy = dy(j) - dpary                  !* perpendicular to Rij
            vz = dz(j) - dparz
            dotuv = ux*vx+uy*vy+uz*vz
            cphi = dotuv/dsqrt((ux**2+uy**2+uz**2)
     &                         *(vx**2+vy**2+vz**2))
*                   !* angle between dipole,Rij and Rhi,Rij planes

            if (cphi.gt.1.0d0) then
              cphi = 1.0d0
            elseif (cphi.lt.-1.0d0) then
              cphi = -1.0d0
            endif
```

```
          viqrgd = viqrgd + 1.5d0*dip(j)*qind(i)
     &      *(cthetad*(3.0d0*cthetaq**2-1.0d0)+2.0d0*sin(thetad)
     &      *sin(thetaq)*cthetaq*cphi)/(rij2(i,j)**2)

        endif

        if (i.lt.j) then    !* Q-Q not double counted

          dotq2 = rijx(nhal,j)*rijx(i,j) + rijy(nhal,j)*rijy(i,j)
     &            + rijz(nhal,j)*rijz(i,j)
          cthetaq2 = dotq2/(rij1(nhal,j)*rij1(i,j))
          if (cthetaq2.gt.1.0d0) then
            cthetaq2 = 1.0d0
          elseif (cthetaq2.lt.-1.0d0) then
            cthetaq2 = -1.0d0
          endif
          thetaq2 = acos(cthetaq2)

          phiqq = 0.0d0     !* both Qs in same plane, pointing at X
          cphiqq = cos(phiqq)

          viqq = viqq + 0.75d0*qind(i)*qind(j)*(1.0d0
     &      - 5.0d0*cthetaq**2 - 5.0d0*cthetaq2**2
     &      + 17.0d0*(cthetaq**2)*(cthetaq2**2)
     &      + 2.0d0*(sin(thetaq)**2)*(sin(thetaq2)**2)*cphiqq**2
     &      + 16.0d0*sin(thetaq)*cthetaq*sin(thetaq2)*cthetaq2
     &        *cphiqq)/(rij3(i,j)*rij2(i,j))
        endif

      enddo
    enddo
*
*  Compute halogen dipole - Rg quadrupole energy Total (pair+3B)
*  minus pair contrib = 3B part
*
    viqxd = 0.0d0
    vpair = 0.0d0
    do i = 1,ncluster

      thetaq = 0.0d0
      cthetaq = cos(thetaq)

      dotd = rijx(i,nhal)*dx(nhal) + rijy(i,nhal)*dy(nhal)
     &       + rijz(i,nhal)*dz(nhal)
      cthetad = dotd/(rij1(i,nhal)*dip(nhal))
      if (cthetad.gt.1.0d0) then
        cthetad = 1.0d0
      elseif (cthetad.lt.-1.0d0) then
        cthetad = -1.0d0
      endif
      thetad = acos(cthetad)

      phi = 0.0d0
      cphi = cos(phi)

      viqxd = viqxd + 1.5d0*dip(nhal)*qind(i)
     &  *(cthetad*(3.0d0*cthetaq**2-1.0d0)+2.0d0*sin(thetad)
```

```
     &     *sin(thetaq)*cthetaq*cphi)/(rij2(nhal,i)**2)

*        !* Pair contribution

         vpair = vpair + 3.0d0*dhalpair(i)*qind(i)
     &                   /(rij2(nhal,i)**2)

       enddo
       viqxd = viqxd - vpair

       viqrgd = viqrgd*1.0d2
       viqq = viqq*1.0d2
       viqxd = viqxd*1.0d2
       vpair = vpair*1.0d2
       potout = viqrgd + viqq + viqxd
       return
       end

*=====================================================================
=======
       subroutine vinddq(polrg,polx,pqrg,pqx,qx,qy,qz,dp,dhalpair,
     &  drgpair,qhalpair,qrgpair,qp,vcd,vcq,potout)
*=====================================================================
=======
*
*  Iterative dipoles & quadrupoles
*  Using eqns from Buckingham, Adv Chem Phys.
*
*  Input:   polrg,polx  RG, halide polarizabilities, a0^3
*           pqrg,pqx    RG, halide quadrupole polarizabilities, a0^5
*                       ="C" as defined by Buckingham for sph. symm.
*           qx,qy,qz(ncl+1)   RG, halide coordinates (a0/10)
*           dp(ncl+1,3) Initial dipoles (e*a0)
*           dhalpair(ncl)     Induced halide dipole in pair potens
*           drgpair(ncl)      Induced Rg dipole in pair potens
*           qhalpair(ncl)     Induced halide quadrupole in pair
potens,initial
*                       (scalar = Qzz = -2Qxx = -2Qyy, units e*a0^2)
*           qrgpair(ncl)      Induced Rg quadrupole in pair potens,
initial
*           qp(ncl+1,3,3)     Initial quadrupoles
*
*  Output:  vcd         Charge-Induced, iterated dipole
poten(hart/100),
*                       three-body part
*           vcq         Charge-Induced,iterated quadrupole
poten(ht/100)
*                       ,three body part
*           potout            Total potential, three body part
(hart/100)
*           dp(ncl+1,3) Iterated dipole vectors (e*a0)
*           qp(ncl+1,3,3)     Iterated quadrupole tensors (e*a0^2, 3x3)
*
       include 'param.file'
       implicit undefined(a-z)
       double precision conv,small,small2,catas,catas2
       parameter(conv=1.0d-10,small = 1.0d-6,
```

```
     & small2=small*small,catas = 10.,catas2=catas*catas)

      double precision rmc,eec,eeint,sigc,sigint,delta,poten,
     & rgrgpot,rgxpot,cddpot,exqpot,vindit,vexqd,rgqq,rgqxd,
     & viq,rgxxpot,rgxipot,rgxiipot,potfin,tempfin,exqpot,rm1,
     & rgdrgq,cdpot,cqpot,cdqpot,gexpot,ddispot,qdispot,gextot,
     & atpot
      double precision dab,dag,dbg,dad,dbd,dgd,fgrg,ferg,vcdt,
     & vcqt,fex,fgx,vcdpair,vcqpair,drgold,dhalold,qrgold,qhalold
*
      double precision polrg,polx,pqrg,pqx,vcd,vcq,potout
      double precision qx(ncl+1),qy(ncl+1),qz(ncl+1),dp(ncl+1,3),
     & dhalpair(ncl),drgpair(ncl),qhalpair(ncl),qrgpair(ncl),
     & qp(ncl+1,3,3),rij(ncl+1,ncl+1,3),rij1(ncl+1,ncl+1),
     & rij2(ncl+1,ncl+1),rij3(ncl+1,ncl+1),rij5(ncl+1,ncl+1),
     & rij7(ncl+1,ncl+1),rij9(ncl+1,ncl+1),t1(ncl+1,ncl+1,3),
     & t2(ncl+1,ncl+1,3,3),t3(ncl+1,ncl+1,3,3,3),rij4(ncl+1,ncl+1),
     & t4(ncl+1,ncl+1,3,3,3,3),fe(ncl+1,3),fg(ncl+1,3,3),
     & dmag(ncl+1),qmag(ncl+1),dmagp(ncl+1),qmagp(ncl+1)

      integer ncluster,natmax,ndim,neqm,neq,nrgpot,nhalpot,neigval,
     &indflag,iexqflag,indi,iexqd,indq,indqi,iexg,iaxtel
      integer nhal,i,j,k,iexit,niter,alp,bet,gam,del,iexsave

      nhal = ncluster+1
      niter = 0
*
*  Calc initial dipole and quadrupole magnitudes
*
      do i = 1,ncluster+1
        dmagp(i) = dp(i,1)**2+dp(i,2)**2+dp(i,3)**2
        qmagp(i) = 0.0d0
        do alp = 1,3
          do bet = 1,3
            qmagp(i) = qmagp(i) + qp(i,alp,bet)**2
          enddo
        enddo
      enddo
*
*  Set up position vectors and distances
*
      do i = 1,ncluster       !* Relative Rg-Rg  and Rg-X coords
        do j = i+1,ncluster+1 !* (rij is vector from atom j to atom i)
          rij(i,j,1) = (qx(i) - qx(j))/10.0d0   !* Convert to a0
          rij(i,j,2) = (qy(i) - qy(j))/10.0d0
          rij(i,j,3) = (qz(i) - qz(j))/10.0d0
          rij(j,i,1) = -rij(i,j,1)
          rij(j,i,2) = -rij(i,j,2)
          rij(j,i,3) = -rij(i,j,3)
          rij2(i,j) = rij(i,j,1)**2+rij(i,j,2)**2+rij(i,j,3)**2
          rij1(i,j) = dsqrt(rij2(i,j))
          rij3(i,j) = rij2(i,j)*rij1(i,j)
          rij4(i,j) = rij2(i,j)**2
          rij5(i,j) = rij3(i,j)*rij2(i,j)
          rij7(i,j) = rij5(i,j)*rij2(i,j)
          rij9(i,j) = rij7(i,j)*rij2(i,j)
          rij1(j,i) = rij1(i,j)
```

```
          rij2(j,i) = rij2(i,j)
          rij3(j,i) = rij3(i,j)
          rij4(j,i) = rij4(i,j)
          rij5(j,i) = rij5(i,j)
          rij7(j,i) = rij7(i,j)
          rij9(j,i) = rij9(i,j)
        enddo
      enddo
*
*
*  Set up interaction tensors (gradients of 1/R)
*
      do i = 1,ncluster                    !* To atom i
        do j = i+1,ncluster+1              !* from atom j
          do alp = 1,3                     !* Alpha
            t1(i,j,alp) = -rij(i,j,alp)/rij3(i,j)
            t1(j,i,alp) = -t1(i,j,alp)
            do bet = 1,3                   !* Beta
              if (alp.eq.bet) then
                dab = 1.0d0                !* Delta function
              else
                dab = 0.0d0
              endif
              t2(i,j,alp,bet) = (3.0d0*rij(i,j,alp)*rij(i,j,bet)
     &           - rij2(i,j)*dab)/rij5(i,j)
              t2(j,i,alp,bet) = t2(i,j,alp,bet)
              do gam = 1,3                 !* Gamma
                if (alp.eq.gam) then
                  dag = 1.0d0
                else
                  dag = 0.0d0
                endif
                if (bet.eq.gam) then
                  dbg = 1.0d0
                else
                  dbg = 0.0d0
                endif
                t3(i,j,alp,bet,gam) =  -3.0d0*
     &          (5.0d0*rij(i,j,alp)*rij(i,j,bet)*rij(i,j,gam)
     &           - rij2(i,j)*(rij(i,j,alp)*dbg + rij(i,j,bet)*dag
     &           + rij(i,j,gam)*dab))/rij7(i,j)
                t3(j,i,alp,bet,gam) = -t3(i,j,alp,bet,gam)
                do del = 1,3               !* Delta
                  if (alp.eq.del) then
                    dad = 1.0d0
                  else
                    dad = 0.0d0
                  endif
                  if (bet.eq.del) then
                    dbd = 1.0d0
                  else
                    dbd = 0.0d0
                  endif
                  if (gam.eq.del) then
                    dgd = 1.0d0
                  else
                    dgd = 0.0d0
```

```
                endif
                t4(i,j,alp,bet,gam,del) = 3.0d0*
     &          (((rij2(i,j)*dab-5.0d0*rij(i,j,alp)*rij(i,j,bet))*dgd
     &           +(rij2(i,j)*dag-5.0d0*rij(i,j,alp)*rij(i,j,gam))*dbd
     &           +(rij2(i,j)*dad-5.0d0*rij(i,j,alp)*rij(i,j,del))*dbg
     &           )/rij7(i,j) - 5.0d0*
     &           (( rij(i,j,gam)*rij(i,j,del)*dab
     &             +rij(i,j,bet)*rij(i,j,del)*dag
     &             +rij(i,j,bet)*rij(i,j,gam)*dad
     &            )*rij2(i,j) - 7.0d0*
     &                 rij(i,j,alp)*rij(i,j,bet)*rij(i,j,gam)
     &             *rij(i,j,del) )/rij9(i,j) )
                t4(j,i,alp,bet,gam,del) = t4(i,j,alp,bet,gam,del)
              enddo
            enddo
          enddo
        enddo
      enddo
*
*
*  Compute electric field, fe, & field gradient, fg, at atoms from
other atoms
*
500   do alp = 1,3
        fe(nhal,alp) = 0.0d0
        do bet = 1,3
          fg(nhal,alp,bet) = 0.0d0
        enddo
      enddo

      do i = 1,ncluster       !* Contribution from halide charge to Rgs
        do alp = 1,3
          fe(i,alp) = t1(i,nhal,alp)      !* q = -1
          fg(i,alp,alp) = t2(i,nhal,alp,alp)
        enddo
        do alp = 1,2
          do bet = alp,3
            fg(i,alp,bet) = t2(i,nhal,alp,bet)  !* q = -1
            fg(i,bet,alp) = fg(i,alp,bet)
          enddo
        enddo
      enddo
*
      do i = 1,ncluster+1     !* Field produced by atom j at atom i
        do j = 1,ncluster+1
          if (i.ne.j) then

            do alp = 1,3
              do bet = 1,3                !* field from dipoles
                fe(i,alp) = fe(i,alp)
     &          + t2(i,j,alp,bet)*dp(j,bet)
                do gam = 1,3              !* field from quadrupoles
                  fe(i,alp) = fe(i,alp)
     &            - t3(i,j,alp,bet,gam)*qp(j,bet,gam)/3.0d0
                enddo
              enddo
```

```
            enddo

            do alp = 1,2
              do gam = 1,3          !* Gradient from dipoles
                fg(i,alp,alp) = fg(i,alp,alp)
     &          + t3(i,j,alp,alp,gam)*dp(j,gam)
                do del = 1,3        !* Gradient from quadrupoles
                  fg(i,alp,alp) = fg(i,alp,alp)
     &            - t4(i,j,alp,alp,gam,del)*qp(j,gam,del)/3.0d0
                enddo
              enddo
              do bet = alp+1,3
                do gam = 1,3        !* Gradient from dipoles
                  fg(i,alp,bet) = fg(i,alp,bet)
     &            + t3(i,j,alp,bet,gam)*dp(j,gam)
                  do del = 1,3            !* Gradient from quadrupoles
                    fg(i,alp,bet) = fg(i,alp,bet)
     &              - t4(i,j,alp,bet,gam,del)*qp(j,gam,del)/3.0d0
                  enddo
                enddo
                fg(i,bet,alp) = fg(i,alp,bet)
              enddo
            enddo
            fg(i,3,3) = -fg(i,1,1)-fg(i,2,2)

          endif

        enddo           !* j
*
*       Compute new induced dipoles & quadrupoles for atom i from E-
field
*       & field gradient "on the fly"
*
        if (i.eq.nhal) then
          dp(nhal,1) = polx*fe(nhal,1)
          dp(nhal,2) = polx*fe(nhal,2)
          dp(nhal,3) = polx*fe(nhal,3)
          qp(nhal,1,1) = fg(nhal,1,1)*pqx
          qp(nhal,2,2) = fg(nhal,2,2)*pqx
          qp(nhal,3,3) = fg(nhal,3,3)*pqx
          do alp = 1,2
            do bet = alp+1,3
              qp(nhal,alp,bet) = fg(nhal,alp,bet)*pqx
              qp(nhal,bet,alp) = qp(nhal,alp,bet)
            enddo
          enddo
        else
          dp(i,1) = polrg*fe(i,1)
          dp(i,2) = polrg*fe(i,2)
          dp(i,3) = polrg*fe(i,3)
          qp(i,1,1) = fg(i,1,1)*pqrg
          qp(i,2,2) = fg(i,2,2)*pqrg
          qp(i,3,3) = fg(i,3,3)*pqrg
          do alp = 1,2
            do bet = alp+1,3
              qp(i,alp,bet) = fg(i,alp,bet)*pqrg
              qp(i,bet,alp) = qp(i,alp,bet)
```

```
            enddo
          enddo
        endif

      enddo

*
*  Check D-dot-D and Q-double dot-Q for convergence
*
      iexit = 1
      do i = 1,ncluster+1
        qmag(i) = qp(i,1,1)**2+qp(i,2,2)**2+qp(i,3,3)**2
     &    +2.*qp(i,1,2)**2+2.*qp(i,1,3)**2+2.*qp(i,2,3)**2
        dmag(i) = dp(i,1)**2+dp(i,2)**2+dp(i,3)**2
        if (dmag(i).gt.0.) then
          if ((dabs((dmag(i)-dmagp(i))/dmag(i)).gt.conv).and.
     &        (dmag(i).gt.small2)) then
            iexit = 0
          endif
        endif
        if (qmag(i).gt.0.) then
          if ((dabs((qmag(i)-qmagp(i))/qmag(i)).gt.conv).and.
     &        (qmag(i).gt.small2)) then
            iexit = 0
          endif
        endif
        if (niter.eq.0) then
          iexit = 0
        endif
*
*       Check for polarization catastrophe
*
        if ((dmag(i).gt.catas2).or.(qmag(i).gt.catas2)) iexit=2

      enddo

      if (iexit.eq.2) then
        write(*,*) 'dipoles : '
        write(*,1000) (dmag(i), i=1,ncluster+1)
        write(*,*) 'quadrupoles : '
        write(*,1000) (qmag(i), i=1,ncluster+1)
      endif
1000  format(7(f10.6,1x))

      niter = niter + 1
*
*  If converged or catastrophe, skip out of loop
*
      if (iexit.ne.0) goto 2000
*
*  Otherwise reset dipole & quadrupole magnitudes & loop back
*
      do i = 1,nhal
        qmagp(i) = qmag(i)
        dmagp(i) = dmag(i)
      enddo
```

```
      goto 500
*
*
2000   k=0
*
*  Converged, now calculate pair RG-X quadrupole and
*  dipole interaction energy. Note this is the total induction energy
*  including charge-dipole, charge-quadrupole, dipole-dipole,
*  dipole-quadrupole, quadrupole-quadrupole, and dipole and quadrupole
*  self energies.  However, the latter 5 terms cancel out.
*  (See Ahlstrom et at, Mol. Phys. v.68, p.563.)
*
      vcdt = 0.
      vcqt = 0.
      do j = 1,ncluster       !* Loop over Rgs
        do alp = 1,3
          vcdt = vcdt - t1(j,nhal,alp)*dp(j,alp)      !* Charge-dipole
          do bet = 1,3
            vcqt = vcqt - t2(nhal,j,alp,bet)*qp(j,alp,bet)  !* Chg-
Quad.
          enddo
        enddo
      enddo
      vcdt = vcdt/2.
      vcqt = vcqt/6.
*
*  Calculate pair induced dipoles and quadrupoles, iteratively.
*  ferg= electric field (z) at Rg, fgrg = field gradient (zz) at Rg,
*  fex, fgx are those at halide
*
      iexsave = iexit
      vcdpair = 0.
      vcqpair = 0.
      do i=1,ncluster         !* Loop over Rgs
5000    ferg = -1./rij2(nhal,i)+2.*dhalpair(i)/rij3(nhal,i)
     &         +3.*qhalpair(i)/rij4(nhal,i)
        fgrg = 2./rij3(nhal,i)-6.*dhalpair(i)/rij4(nhal,i)
     &         -12.*qhalpair(i)/rij5(nhal,i)
        drgold = drgpair(i)
        drgpair(i) =  polrg*ferg
        qrgold = qrgpair(i)
        qrgpair(i) = pqrg*fgrg
        fex = 2.*drgpair(i)/rij3(nhal,i)-3.*qrgpair(i)/rij4(nhal,i)
        fgx = 6.*drgpair(i)/rij4(nhal,i)-12.*qrgpair(i)/rij5(nhal,i)
        dhalold = dhalpair(i)
        dhalpair(i) = polx*fex
        qhalold = qhalpair(i)
        qhalpair(i) = pqx*fgx
        iexit = 1
        if (drgpair(i).ne.0.) then
          if ((dabs((drgold-drgpair(i))/drgpair(i)).gt.conv).and.
     &        (dabs(drgpair(i)).gt.small)) iexit=0
        endif
        if (dhalpair(i).ne.0.) then
          if ((dabs((dhalold-dhalpair(i))/dhalpair(i)).gt.conv).and.
     &        (dabs(dhalpair(i)).gt.small))iexit=0
        endif
```

```
        if (qrgpair(i).ne.0.) then
          if ((dabs((qrgold-qrgpair(i))/qrgpair(i)).gt.conv).and.
     &         (dabs(qrgpair(i)).gt.small))iexit=0
        endif
        if (qhalpair(i).ne.0.) then
          if ((dabs((qhalold-qhalpair(i))/qhalpair(i)).gt.conv).and.
     &         (dabs(qhalpair(i)).gt.small))iexit=0
        endif
        if ((drgpair(i).gt.catas).or.(dhalpair(i).gt.catas).or.
     &      (qrgpair(i).gt.catas).or.(qhalpair(i).gt.catas)) then
          iexit = 2
        endif
        if (iexit.eq.0) goto 5000
        vcdpair = vcdpair + drgpair(i)/(2.*rij2(nhal,i))
        vcqpair = vcqpair - qrgpair(i)/(2.*rij3(nhal,i))
      enddo
      if (iexsave.eq.2) iexit = 2

*
*  Subtract pair from total to get nonadditive parts
*
      vcd = (vcdt - vcdpair)*1.0d2
      vcq = (vcqt - vcqpair)*1.0d2
      potout = vcd+vcq

      if (iexit.eq.2) then    !* Reset multipoles if catastrophe has
happened
        write(*,*) '****** Polarization catastrophe *******'
        do i=1,ncluster
          write(*,*) 'pair dipoles rg, x',drgpair(i),dhalpair(i)
          drgpair(i) = 0.
          dhalpair(i) = 0.
          write(*,*) 'pair quadrupoles rg,x',qrgpair(i),qhalpair(i)
          qrgpair(i) = 0.
          qhalpair(i) = 0.
          do alp = 1,3
            dp(j,alp) = 0.
            do bet = 1,3
              qp(j,alp,bet) = 0.
            enddo
          enddo
        enddo
      endif

      return
      end

*=====================================================================
===
      subroutine exg(betain,th6,qx,qy,qz,vexx,vddisp,vqdisp,potout)
*=====================================================================
===
*
*  Exchange quadrupole-charge energy calculated exactly from Gaussian
*  overlaps
*
*  Input:   betain            Gaussian range parameter (a0/10)^-1
```

```
*           th6          Dispersion quadrupole coeff (e*a0^8)
*           qx,qy,qz(ncl+1)   Rg, halide coordinates (a0/10)
*
*  Output:  potout             potential (hart/100)
*
*  Uses:    erf          error function
*
      include 'param.file'
      parameter(toosmall = 1.0d-1)
      double precision betain,th6,potout,erf
      double precision qx(ncl+1),qy(ncl+1),qz(ncl+1),
     & rxi(ncl),rxix(ncl),rxiy(ncl),rxiz(ncl),coef(ncl),
     & disd(ncl,3),disq(ncl,3,3),t2(3,3)
      integer alp,bet
*
*
      beta = betain*10.0d0
      beta2 = beta**2
      thq = -th6/10.          !* quadrupole coeff
      thd = 3.*th6/5.         !* dipole coeff
      nhal = ncluster+1
      potout = 0.
      vexx = 0.
      vddisp = 0.
      vqdisp = 0.

      do i = 1,ncluster
        rxix(i) = (qx(i)-qx(nhal))/10.    !* RG-X distances
        rxiy(i) = (qy(i)-qy(nhal))/10.
        rxiz(i) = (qz(i)-qz(nhal))/10.
        rij2 = rxix(i)**2+rxiy(i)**2+rxiz(i)**2
        rxi(i) = dsqrt(rij2)
        coef(i) = 0.
        do alp=1,3
          disd(i,alp)=0.
          do bet=1,3
            disq(i,alp,bet)=0.
          enddo
        enddo
      enddo

      do i = 1,ncluster-1
        do j = i+1,ncluster
          rijx = (qx(i)-qx(j))/10.  !* Rg-Rg distances
          rijy = (qy(i)-qy(j))/10.
          rijz = (qz(i)-qz(j))/10.
          rij2 = rijx**2+rijy**2+rijz**2
          t2(1,1) = (3.*rijx**2/rij2 - 1) !* Quadrupole int tensor
          t2(2,2) = (3.*rijy**2/rij2 - 1)
          t2(3,3) = (3.*rijz**2/rij2 - 1)
          t2(1,2) = 3.*rijx*rijy/rij2
          t2(1,3) = 3.*rijx*rijz/rij2
          t2(2,3) = 3.*rijy*rijz/rij2
          t2(2,1) = t2(1,2)
          t2(3,1) = t2(1,3)
          t2(3,2) = t2(2,3)
          rij6 = rij2**3
```

```
          rij8 = rij6*rij2
          qco = thq/rij6/2.
          dco = thd/rij8
          disd(i,1) = disd(i,1)+dco*rijx  !* Dispersion dipoles
          disd(i,2) = disd(i,2)+dco*rijy
          disd(i,3) = disd(i,3)+dco*rijz
          disd(j,1) = disd(j,1)-dco*rijx
          disd(j,2) = disd(j,2)-dco*rijy
          disd(j,3) = disd(j,3)-dco*rijz
          do alp = 1,3                    !* Dispersion quadrupoles
            do bet = 1,3
              disq(i,alp,bet)=disq(i,alp,bet)+qco*t2(alp,bet)
              disq(j,alp,bet)=disq(j,alp,bet)+qco*t2(alp,bet)
            enddo
          enddo
          sov2 = exp(-beta2*rij2/2.)
          fac = sov2/(1-sov2)
          rcx = (rxix(i)+rxix(j))/2.      !* X-(Rg-Rg center) distances
          rcy = (rxiy(i)+rxiy(j))/2.
          rcz = (rxiz(i)+rxiz(j))/2.
          rc = dsqrt(rcx**2+rcy**2+rcz**2)
          if (rc.gt.toosmall) then
            vexx = vexx-2.*fac*erf(beta*rc)/rc  !* Contrib from + chg
            coef(i) = coef(i) + fac             !* at Rg-Rg center
            coef(j) = coef(j) + fac
          endif
        enddo
      enddo

      do i = 1,ncluster       !* Contribs from - charges at Rg nuclei
        vexx = vexx + coef(i)*erf(beta*rxi(i))/rxi(i)
        rij2 = rxi(i)**2
        t2(1,1) = (3.*rxix(i)**2/rij2 - 1)      !* Quadrupole int
tensor
        t2(2,2) = (3.*rxiy(i)**2/rij2 - 1)
        t2(3,3) = (3.*rxiz(i)**2/rij2 - 1)
        t2(1,2) = 3.*rxix(i)*rxiy(i)/rij2
        t2(1,3) = 3.*rxix(i)*rxiz(i)/rij2
        t2(2,3) = 3.*rxiy(i)*rxiz(i)/rij2
        t2(2,1) = t2(1,2)
        t2(3,1) = t2(1,3)
        t2(3,2) = t2(2,3)
        vddisp = vddisp + (rxix(i)*disd(i,1)+rxiy(i)*disd(i,2)
     &                     +rxiz(i)*disd(i,3))/(rxi(i)**3)
        qsum2 = 0.
        trace = 0.
        do alp = 1,3
          do bet = 1,3
            vqdisp = vqdisp
     &        - t2(alp,bet)*disq(i,alp,bet)/(rxi(i)**3)/3.
          enddo
        enddo
      enddo

      vexx = vexx*1.0d2 !* convert to hartree/100
      vddisp = vddisp*1.d2
      vqdisp = vqdisp*1.d2
```

```
      potout = vexx+vddisp+vqdisp

      return
      end

*=====================================================================
      double precision FUNCTION ERF(xin)
*=====================================================================
*
*  Error function.  Adapted from Numerical Recipies.
*
      implicit double precision (a-h,o-z)
      PARAMETER (ITMAX=500,EPS=5.d-14,GLN=0.57236494292470d0,
     &  fpmin = 1.d-30)

      X = xin**2
      A = 0.5d0

      IF(X.LT.1.5)THEN
        IF(xin.lt.0.)THEN
          GAMMP=0.
          goto 3
        ENDIF
        AP=A
        SUM=1.0d0/A
        DEL=SUM
        DO N=1,ITMAX
          AP=AP+1.
          DEL=DEL*X/AP
          SUM=SUM+DEL
          IF(ABS(DEL).LT.ABS(SUM)*EPS)GO TO 1
        enddo
        PAUSE 'A too large, ITMAX too small'
1       GAMMP=SUM*EXP(-X+A*LOG(X)-GLN)
      ELSE
        b=x+1.0-a
        c=1.0/fpmin
        d=1.0/b
        h=d
        do i=1,itmax
          an=-i*(i-a)
          b=b+2.0d0
          d=an*d+b
          if (abs(d).lt.fpmin) d=fpmin
          c=b+an/c
          if (abs(c).lt.fpmin) c=fpmin
          d=1.0/d
          del=d*c
          h=h*del
          if (abs(del-1.0).lt.eps) goto 2
        enddo
        pause 'a too large, ITMAX too small in gcf'
2       gammcf=exp(-x+a*log(x)-gln)*h
        GAMMP=1.0-GAMMCF
      ENDIF

3     IF(xin.LT.0.)THEN
```

```
        ERF=-GAMMP
      ELSE
        ERF=GAMMP
      ENDIF

      RETURN
      END

*=====================================================================
======
      subroutine vat(c9,qx,qy,qz,potout)
*=====================================================================
======
*
*  Triple-dipole (Axilrod-Teller) potential
*  Ref: Allen & Tildesley, Computer Simulation of Liquids, p. 334.
*
*  Input:       c9                  Triple-dipole constant (au)
*               qx,qy,qz(ncl+1)           Rg, Halide cart. coordinates
(a0/10)
*
*
*  Output:      potout           Potential (hartree/100)
*
*

      include 'param.file'
      double precision c9,potout
      double precision qx(ncl+1),qy(ncl+1),qz(ncl+1)

      nhal = ncluster+1
      potout = 0.
      do i = 1,ncluster-1
        xik = (qx(nhal)-qx(i))/10.
        yik = (qy(nhal)-qy(i))/10.
        zik = (qz(nhal)-qz(i))/10.
        do j = i+1,ncluster
          xij = (qx(j)-qx(i))/10.
          xjk = (qx(nhal)-qx(j))/10.
          yij = (qy(j)-qy(i))/10.
          yjk = (qy(nhal)-qy(j))/10.
          zij = (qz(j)-qz(i))/10.
          zjk = (qz(nhal)-qz(j))/10.

          dikdjk = xik*xjk + yik*yjk + zik*zjk
          dikdij = xik*xij + yik*yij + zik*zij
          dijdjk = xij*xjk + yij*yjk + zij*zjk

          rij2 = xij*xij+yij*yij+zij*zij
          rik2 = xik*xik+yik*yik+zik*zik
          rjk2 = xjk*xjk+yjk*yjk+zjk*zjk

          pd = rij2*rjk2*rik2

          potout = potout+(pd - 3.*dikdjk*dikdij*dijdjk)
     &            /(pd*pd*dsqrt(pd))
        enddo
```

```
      enddo

      potout = potout*c9*1.d2

      return
      end
```

C5.8. File "averages.f"

```
      subroutine averages(y0,nstep,h,potave,potsd,pmin,prgrg,
     &   prgx,pind,pexq,pex,pddis,pqdis,pextot,pindi,pcd,pcq,pinddq,
     &   prgdrgq,prgqq,prgqxd,piq,pexqd,prgxX,prgxI,prgxII,ekinave,
     &   energy,yave,rgx,rgrg,
     &   dip,qin,paxtel)
*
*  Subroutine to find average parameters during a constant E MD run
*
      include 'param.file'

      common /dipoles/dx(natmaxc),dy(natmaxc),dz(natmaxc),
     &  qind(natmaxc),dpol(natmaxc,3),qpol(natmaxc,3,3)

      DIMENSION Y(neqc),D1Y(neqc),D2Y(neqc),D3Y(neqc),D4Y(neqc),
     .ZY(neqc),Y0(neqc),Y1(neqc),Y2(neqc),Y3(neqc),YH(neqc),Y3P(neqc),
     .F0(neqc),F1(neqc),F2(neqc),F3(neqc),FH(neqc),ZF(neqc),
     .fdum(neqc),yave(neqc),ysd(neqc),rgx(2,ncl),rgrg(2,ncl,ncl),
     &dip(natmaxc),dx(natmaxc),dy(natmaxc),dz(natmaxc),qin(natmaxc),
     &qind(natmaxc)
C
C-------------------------------------------------------------------------
--------
C     DATA FOR INTEGRATOR ALGORITHMS
C-------------------------------------------------------------------------
--------
C     DATA FOR RUNGE KUTTA INTEGRATION
      B11= 1.0D0/3.0D0
      B21=-1.0D0/3.0D0
      B22= 1.0D0
      B31= 1.0D0
      B32=-1.0D0
      B33= 1.0D0
       W1= 1.0D0/8.0D0
       W2= 3.0D0/8.0D0
       W3= 3.0D0/8.0D0
       W4= 1.0D0/8.0D0
C
C     DATA FOR HYBRID GEAR ROUTINE
      GA02= 153.0D0/128.0D0
      GA01= 25.0D0/16.0D0
      GA00=-225.0D0/128.0D0
      GB02= 45.0D0/128.0D0
      GB01= 75.0D0/32.0D0
      GB00= 225.0D0/128.0D0
C
C     ALPHA1=-0.5 FOR STABILITY
```

```
C
      GA12= (15.0D0/16.0D0)-0.5D0*(29.0D0/32.0D0)
      GA11= (-1.0D0)-0.5D0*(1.0D0)
      GA10= (17.0D0/16.0D0)-0.5D0*(-61.0D0/32.0D0)
      GB12= (5.0D0/16.0D0)-0.5D0*(43.0D0/160.0D0)
      GB11= (11.0D0/12.0D0)-0.5D0*(41.0D0/24.0D0)
      GB10= (-11.0D0/16.0D0)-0.5D0*(31.0D0/32.0D0)
      GG10= (4.0D0/3.0D0)-0.5D0*(-2.0D0/15.0D0)
C
C     ALPHA2=1.0, BETA=9/56
C
      BETA= 9.0D0/56.0D0
      GA22= 1.5D0*(29.0D0/32.0D0)+BETA*(-45.0D0/4.0D0)
      GA21= 1.5D0*(1.0D0)+BETA*(0.0D0)
      GA20= 1.5D0*(-61.0D0/32.0D0)+BETA*(45.0D0/4.0D0)
      GB22= 1.5D0*(43.0D0/160.0D0)+BETA*(-71.0D0/20.0D0)
      GB21= 1.5D0*(41.0D0/24.0D0)+BETA*(-16.0D0)
      GB20= 1.5D0*(31.0D0/32.0D0)+BETA*(-3.0D0/4.0D0)
      GG20= 1.5D0*(-2.0D0/15.0D0)+BETA*(-16.0D0/5.0D0)
      GG21= BETA*1.0D0
c
      call deriv(f0,y0)
      call kinetic(y0,ekin)
      energy = ekin + poten
*
*  Zero averages
*
      do jj=1,neq
        yave(jj) = 0.
        ysd(jj) = 0.
      enddo
      do ii=1,ncluster
        rgx(1,ii) = 0.
        rgx(2,ii) = 0.
        do jj=1,ncluster
          rgrg(1,ii,jj) = 0.
          rgrg(2,ii,jj) = 0.
        enddo
      enddo


C
C-------------------------------------------------------------------------
--------
C     SIXTEEN STEP RUNGE KUTTA INTEGRATION TO START
C-------------------------------------------------------------------------
--------
C
      DH=H/8.0D0
      ICOUNT=0
      DO 99 I=1,NEQ
      Y(I)=Y0(I)
99    CONTINUE
C
      CALL DERIV(F0,Y)
C
      DO 1000 ISTEP=1,16
```

```
C
      ICOUNT=ICOUNT+1
C
      CALL DERIV(ZF,Y)
C
      DO 100 I=1,NEQ
      D1Y(I)=DH*ZF(I)
100   CONTINUE
C
      DO 200 I=1,NEQ
      ZY(I)=Y(I)+B11*D1Y(I)
200   CONTINUE
C
      CALL DERIV(ZF,ZY)
C
      DO 300 I=1,NEQ
      D2Y(I)=DH*ZF(I)
300   CONTINUE
C
      DO 400 I=1,NEQ
      ZY(I)=Y(I)+B21*D1Y(I)+B22*D2Y(I)
400   CONTINUE
C
      CALL DERIV(ZF,ZY)
C
      DO 500 I=1,NEQ
      D3Y(I)=DH*ZF(I)
500   CONTINUE
C
      DO 600 I=1,NEQ
      ZY(I)=Y(I)+B31*D1Y(I)+B32*D2Y(I)+B33*D3Y(I)
600   CONTINUE
C
      CALL DERIV(ZF,ZY)
C
      DO 700 I=1,NEQ
      D4Y(I)=DH*ZF(I)
700   CONTINUE
C
      DO 800 I=1,NEQ
      Y(I)=Y(I)+W1*D1Y(I)+W2*D2Y(I)+W3*D3Y(I)+W4*D4Y(I)
800   CONTINUE
C
      IF(ICOUNT.EQ.8)THEN
      CALL DERIV(F1,Y)
      DO 810 I=1,NEQ
      Y1(I)=Y(I)
810   CONTINUE
      ELSE IF(ICOUNT.EQ.16)THEN
      CALL DERIV(F2,Y)
      DO 820 I=1,NEQ
      Y2(I)=Y(I)
820   CONTINUE
      END IF
C
1000  CONTINUE
C
```

```
C
C-------------------------------------------------------------------------
--------
C     ENTER MAIN INTEGRATION LOOP
C-------------------------------------------------------------------------
--------
      ICOUNT=2
      itime = 2
      iave = 0
      ekinave = 0.0d0
      potave = 0.0d0
      potsd = 0.
c
      DO 2000 ISTEP=1,NSTEP
      ICOUNT=ICOUNT+1
      itime = itime + 1
      iave = iave + 1
C
      DO 1100 I=1,NEQ
      J=I
      YH(I)=GA02*Y0(I)+GA01*Y1(I)+GA00*Y2(I)+
     .H*(GB02*F0(I)+GB01*F1(I)+GB00*F2(I))
1100  CONTINUE
C
      CALL DERIV(FH,YH)
C
C     CALCULATE PREDICTED ARRAY Y3P
C
      DO 1300 I=1,NEQ
      Y3P(I)=GA12*Y0(I)+GA11*Y1(I)+GA10*Y2(I)+
     .H*(GB12*F0(I)+GB11*F1(I)+GB10*F2(I))+
     .H*GG10*FH(I)
1300  CONTINUE
C
      CALL DERIV(F3,Y3P)
C
C     CALCULATE CORRECTED ARRAY Y3
C     SET LOCAL TRUNCATION ERROR ERRLOC EQUAL TO ZERO
      ERRLOC=0.0D0
C
      DO 1500 I=1,NEQ
      Q=GA22*Y0(I)+GA21*Y1(I)+GA20*Y2(I)+
     .H*(GB22*F0(I)+GB21*F1(I)+GB20*F2(I))+
     .H*(GG20*FH(I)+GG21*F3(I))
c
      Y3(I)=Y3P(I)+Q
1500  CONTINUE
*
*  call deriv again at corrected configuration to get
*  correct potential and forces
*
      call deriv(F3,Y3)
      potave=potave+poten
      potsd=potsd+poten**2
*
*  Accumulate averages
*  Average positions: yave(1-3) positions, yave(4-6) squares
```

```
*
      do jj = 1,neq-5,6
        yave(jj) = yave(jj) + y3(jj)
        yave(jj+1) = yave(jj+1) + y3(jj+1)
        yave(jj+2) = yave(jj+2) + y3(jj+2)
        yave(jj+3) = yave(jj+3) + y3(jj)**2
        yave(jj+4) = yave(jj+4) + y3(jj+1)**2
        yave(jj+5) = yave(jj+5) + y3(jj+2)**2
      enddo
*
*  Average RG - X distances
*    rgx(1,j) = sum of distances
*    rgx(2,j) = sum of squares
*
*  Average RG-RG distances
*    rgrg(1,i,j) = sum of distances
*    rgrg(2,i,j) = sum of squares
*
      xhal = y3(neq-5)
      yhal = y3(neq-4)
      zhal = y3(neq-3)
      iic = 0
      do ii = 1,neq-11,6
        iic = iic+1
        rsq = (y3(ii)-xhal)**2 + (y3(ii+1)-yhal)**2
     &        + (y3(ii+2)-zhal)**2
        rgx(1,iic) = rgx(1,iic) + sqrt(rsq)
        rgx(2,iic) = rgx(2,iic) + rsq
        jjc = iic - 1
        do jj = ii,neq-11,6
          jjc = jjc+1
          rsq = (y3(ii)-y3(jj))**2 + (y3(ii+1)-y3(jj+1))**2
     &          +(y3(ii+2)-y3(jj+2))**2
          rgrg(1,iic,jjc) = rgrg(1,iic,jjc) + sqrt(rsq)
          rgrg(2,iic,jjc) = rgrg(2,iic,jjc) + rsq
        enddo
      enddo
C
C     RESET ARRAYS FOR NEXT STEP
C
      DO 1600 I=1,NEQ
      Y0(I)=Y1(I)
      Y1(I)=Y2(I)
      Y2(I)=Y3(I)
      F0(I)=F1(I)
      F1(I)=F2(I)
      F2(I)=F3(I)
1600  CONTINUE

      call kinetic(y2,ekin)
      ekinave = ekinave + ekin
C
2000  CONTINUE
*
*  Calculate Averages and standard deviations
*
      dstep = dble(nstep)
```

```
      ekinave = ekinave/dstep
      potave = potave/dstep
      potsd = dsqrt(dabs(potsd/dstep - potave**2))
      do jj = 1,neq-5,6
        do nn=0,2
          yave(jj+nn) = yave(jj+nn)/dstep
          ysd(jj+nn) = dsqrt(dabs(yave(jj+nn+3)/dstep
     &                          - yave(jj+nn)**2))
        enddo
      enddo
*
*  calculate potential at average configuration
*
      call deriv(fdum,yave)
      pmin = poten
      prgrg = rgrgpot
      prgx = rgxpot
      pind = cddpot
      pexq = exqpot
      pex = gexpot
      pddis = ddispot
      pqdis = qdispot
      pextot = gextot
      pindi = vindit
      pcd = cdpot
      pcq = cqpot
      pinddq = cdqpot
      prgdrgq = rgdrgq
      prgqq = rgqq
      prgqxd = rgqxd
      piq = viq
      pexqd = vexqd
      prgxX = rgxXpot
      prgxI = rgxIpot
      prgxII = rgxIIpot
      paxtel = atpot
*
*  Dipoles & Quadrupoles
*
      if (indqi.eq.0) then
        do iii = 1,ncluster+1
          dip(iii) = dsqrt(dx(iii)**2+dy(iii)**2+dz(iii)**2)
          qin(iii) = qind(iii)
        enddo
      else
        do ii = 1,ncluster+1
          dip(ii)=0.
          qin(ii)=0.
          do jj = 1,3
            dip(ii) = dip(ii) + dpol(ii,jj)**2
            do kk = 1,3
              qin(ii) = qin(ii) + qpol(ii,jj,kk)**2
            enddo
          enddo
          dip(ii) = dsqrt(dip(ii))
          qin(ii) = dsqrt(qin(ii))
        enddo
```

```
      endif

*
      jjc = 0
      do jj = 1,neq-11,6
        jjc = jjc+1
        rgx(1,jjc) = rgx(1,jjc)/dstep
        rgx(2,jjc) = sqrt(dabs(rgx(2,jjc)/dstep-rgx(1,jjc)**2))
      enddo

      iic = 0
      do ii = 1,neq-11,6
        iic = iic + 1
        jjc = iic -1
        do jj = ii,neq-11,6
          jjc = jjc + 1
          rgrg(1,iic,jjc) = rgrg(1,iic,jjc)/nstep
          rgrg(2,iic,jjc) = sqrt(dabs(rgrg(2,iic,jjc)/dstep
     &                          - rgrg(1,iic,jjc)**2))
        enddo
      enddo

C
C
      do i = 1, neq
      y0(I)=y2(i)
      end do
c
      return
      END
```

C5.9. File "ch.f"[17]

```
      subroutine ch(nm,n,ar,ai,w,matz,zr,zi,fv1,fv2,fm1,ierr)
c
      integer i,j,n,nm,ierr,matz
      double precision ar(nm,n),ai(nm,n),w(n),zr(nm,n),zi(nm,n),
     x       fv1(n),fv2(n),fm1(2,n)
c
c     this subroutine calls the recommended sequence of
c     subroutines from the eigensystem subroutine package (eispack)
c     to find the eigenvalues and eigenvectors (if desired)
c     of a complex hermitian matrix.
c
c     on input
c
c        nm  must be set to the row dimension of the two-dimensional
c        array parameters as declared in the calling program
c        dimension statement.
c
c        n  is the order of the matrix  a=(ar,ai).
c
c        ar  and  ai  contain the real and imaginary parts,
c        respectively, of the complex hermitian matrix.
c
```

```
c        matz  is an integer variable set equal to zero if
c        only eigenvalues are desired.  otherwise it is set to
c        any non-zero integer for both eigenvalues and eigenvectors.
c
c     on output
c
c        w  contains the eigenvalues in ascending order.
c
c        zr  and  zi  contain the real and imaginary parts,
c        respectively, of the eigenvectors if matz is not zero.
c
c        ierr  is an integer output variable set equal to an error
c           completion code described in the documentation for tqlrat
c           and tql2.  the normal completion code is zero.
c
c        fv1, fv2, and  fm1  are temporary storage arrays.
c
c     questions and comments should be directed to burton s. garbow,
c     mathematics and computer science div, argonne national laboratory
c
c     this version dated august 1983.
c
```

C6. Documentation of the "normal" program for finding zero-point energies

The "normal" program is used in conjunction with the simulated annealing program for finding the zero point energies of clusters. It accepts as input the cluster minimum energy configuration files produced by the simulated annealing program and calculates vibrational eigenvalues for each mode of the cluster as described in Section 4.4.3 (Chapter 4). The program also can be used for calculating higher vibrational eigenvalues and Franck-Condon factors.

C6.1. Example: Zero-point energy calculation for Ar_6I^-

C6.1.1. The input file

The input file for the "normal" program for the global minimum configuration of Ar_6I^- found in section C4.3 is shown below. The line numbers in boldface are for reference only and are not included in the actual input file. The name of the file is "normal_in1_ar6i."

```
1:  Comment line
2:  1    !* ndo (1 anion or neutral, 2 fcfs)
3:  1    !* nfm (1 to calculate force mat, 2 to read from file)
4:  6    !* nrg
5:  1    !* nhalpot (1 for anion, 2 for neutral)
6:  1    !* neigval (neutral 1 or 2 for X; 3,4 for I; 5,6 for II)
7:  0    !* indflag
8:  0    !* iexqflag
9:  0    !* iexqd (exchange dipoles)
10: 0     !* iexg (Gaussian exchange)
11: 0    !* indqi (iterated multipoles)
12: 0     !* iaxtel
13: average_file4_6
14: neutral_file
15: nc1_1
16: normal_out1_6
17: zpe1_6
18: wf1_6
19: dvr1_6
20: vsticks
21: fm1_write
22: fm1_read
23: 0.94268    !* I SO Const 0.94268 eV
24: 1.6419     !* polind (RG polarizability Ang^3)
25: 11.08      !* polrg
26: 52.7       !* polx
27: 27.11.     !* pqrg
28: 254.       !* pqx
29: 0.936      !* betaexq (Exchange quadrupole range parameter, Ang^-1)
30: 6.5        !* cutexq (exchange quad cutoff distance, Ang)
31: 2086.      !* theta6
32: 0.         !* c9anion
33: 0.         !* c9neut
34: 35         !* ndvr
35: 2          !* nanshow
36: 8          !* nnshow
37: 40.        !* vtemp
38: 0.0        !* origin (cm-1)
39: 0.001      !* 2nd deriv step size (Ang)
40: 126.9044         !* halogen mass (amu) Iodine
41: 39.96238         !* rare gas mass (amu) Argon
42: 0.0458, 4.07, 5.70, 4.45, 1.08, 1.62, 279.5, 3537.  !* Anion MMSV
43: 0.0188, 3.95, 7.15, 6.18, 1.01, 1.62, 5234., 38032. !* X1/2 MMSV
44: 0.0139, 4.18, 7.25, 6.30, 1.04, 1.62, 7079., 51439. !* I3/2 MMSV
45: 0.0159, 4.11, 6.90, 6.40, 1.04, 1.64, 6189., 44969. !* II1/2 MMSV
46: 0.0123422,3.7565,10.77874743,1.8122004,2.26210716e5,1.10785136,
      .56072459,.34602794,1.36            !* Ar-Ar HFD-B
```

```
47: 0          !* How many lambdas to enter by hand
```

Line **1** is a comment line which is copied directly to the output file. Line **2** determines the mode of running the program: when this line is set to 1, only the calculation for either the anion or neutral eigenvalues is performed; when it is 2, the eigenvalues for both anion and neutral are calculated as well as Franck-Condon factors and a vibrational stick spectrum. The setting of line **3** determines whether the force matrix is calculated from the atomic configuration and potential (line **3** = 1) or if a previously calculated force matrix is read in from in from a file (line **3** = 2).

Lines **5-12, 23-33** and **42-46** describe the nature of the cluster and its potential and are identical to the corresponding lines of the input file "in1" to the simulated annealing program (see Section C4.1, above).

Lines **13** and **14** are the anion and neutral atomic configuration files generated by the simulated annealing program used as input by "normal." If the FCFs calculation is not performed, only one of the files is read in, but the input file must nonetheless contain both lines. In this example we are only interested in the zero point energies of the Ar_6I^-, and so have placed the Ar_6I^- global minimum configuration file "average_file4_6" in line **13** and a "dummy" file name in line **14**. Line **15** is the name of a file to save the normal coordinates, which may be examined to ascertain the symmetry of each vibrational mode. Line **16** is the name of the program output file, which lists the individual mode vibrational frequencies and zero-point energies. If FCF mode is enabled, the FCFs are also listed in this file. Line **17** is the name of a file to save the total zero point energy. Line **18** is the name of a file to save the DVR eigenvalues and eigenvectors for each mode. As when running the Morse DVR program (Section A5.2), the eigenvectors

should be examined for convergence. The file named in line **19** lists the DVR points for each vibrational mode. The file named in line **20** is the vibrational stick spectrum (if FCF mode is enabled). Line **21** is the name of the file to save the calculated force matrix. Line **22** is the name of the file from which to read in the force matrix if it is not to be calculated.

The program uses a harmonic oscillator-based DVR method[18] for solving Hamiltonian along each normal coordinate. Line **34** is the number of harmonic oscillator basis functions to be used in the calculation. Lines **35** and **36** are the number of anion and neutral eigenvalues, respectively, one wishes to calculate. Line **37** is the vibrational temperature used to generate the vibrational stick spectrum. Line **38** is the origin, in wavenumbers, of the vibrational stick spectrum. Line **39** is the step size in Å used in calculating the second derivatives of the potential for the force matrix. Lines **40** and **41** are the halogen and rare gas masses in amu, respectively.

The frequency of the harmonic oscillator basis set for the DVR calculation is chosen to be the same for each mode as the harmonic frequency calculated numerically from the potential. In some cases, the DVR calculation does not converge well with this choice of basis frequency. In such cases one can specify the basis frequency for one or mode by setting line **47** of the input file to the number of basis frequencies to be set by hand. The user is then prompted for the numbers of the normal modes (as listed in the output file) and the basis frequencies.

C6.1.2. Running the program

The program is run as in the following example, with user input shown in boldface.

```
> normal < normal_in1_6

>
```

The output file "normal_out1_6" is then generated, and is excerpted here:

```
Anion potential at eq. (eV):    -0.4003685069
Anion
 Mode Lambda (s-2)  Harm.freq(cm-1)  ZPE DVR    v1-v0 DVR
      -0.66040E+20       0.00
      -0.63225E+20       0.00
      -0.34854E+20       0.00
       0.28304E+19       0.01
       0.27080E+20       0.03
       0.10132E+21       0.05
  1    0.12983E+26      19.13          9.75       19.83
  2    0.12983E+26      19.13          9.75       19.83
  3    0.16334E+26      21.46         10.78       21.65
  4    0.16335E+26      21.46         10.78       21.65
  5    0.17361E+26      22.12         11.32       23.11
  6    0.26132E+26      27.14         13.88       28.37
  7    0.26135E+26      27.14         13.89       28.37
  8    0.46533E+26      36.21         17.94       35.70
  9    0.54459E+26      39.18         19.48       39.05
 10    0.54461E+26      39.18         19.48       39.05
 11    0.55494E+26      39.55         19.20       38.26
 12    0.55496E+26      39.55         19.21       38.29
 13    0.88910E+26      50.06         24.58       49.13
 14    0.88912E+26      50.06         24.58       49.13
 15    0.93776E+26      51.41         24.82       48.44

Harmonic zero point energy (cm-1):  251.43
                           (eV):    0.0311728180

DVR zero point energy (cm-1):  249.43
                      (eV):    0.0309252277
```

The first column of this file is the mode number, the second column is the harmonic mode frequency in Hz, the third column is the harmonic mode frequency, the fourth column contains the zero point energies for each mode from the DVR calculation, and the fifth column contains the mode frequencies from the DVR calculation.

C6.2. Outline of the program

The subroutines and functions used by the program "normal" are listed in Table C2, sorted according the file in which they reside. Some of the functions and subroutines are the same as those previously described for the simulated annealing program and Morse DVR program, and these are noted in the "Description" column of the table.

Table C2. Subroutines and functions used by the program "normal"

File	Subroutine or Function	Description
param.file		See Table C1
normal.f	normal	Main program
	peval	Evaluates neutral potential of cluster
	apeval	Evaluate anion potential of cluster
	pind	Calculates three-body induction potential from Eq. (C10)
	vexq	Calculates three-body "exchange quadrupole" potential from Eq. (C12)
	hopts	Calculates DVR points using harmonic oscillator basis for a given vibrational mode
	hokin	Transforms kinetic energy matrix to DVR basis
	hoham	Calculated DVR Hamiltonian
	prinaxes	Transforms cluster coordinates to principal axis system (used in FCF mode)
	calcfcfs	Calculates multi-mode Frank-Condon factors and vibrational stick spectrum
	veqd	Calculated exchange-quadrupole potential using the distributed dipole model.
	vinddq	See Table C1
	exg	See Table C1
	erf	See Table C1
	vat	See Table C1
porb.f	porb	See Table C1
poten2.f		See Table A1
ch.f	ch	See Table C1
rs.f	rs	See Table A1
rsb.f	rsb	See Table A1

C6.3. Source code for the program "normal"

The source code for the "normal" program is listed below. The files "rsb.f", "rs.f", and "poten2.f" are identical to those listed or described in Sections A6.3, A6.4 and A6.5 and hence are omitted here, as is the file "ch.f" which is identical to the EISPACK subroutine described in Section C5.9. Some of the subroutines and functions in the files "normal.f" and the subroutine in the file "porb.f" are identical with those in the simulated annealing program listing, and only the headers of these are reproduced below.

6.3.1. File "makenormal"

The program is recompiled with the command "make -f makenormal". The executable file "normal" is saved in the subdirectory "norm." The makefile is listed below.

```
normal: normal.o porb.o poten2.o ch.o rs.o rsb.o
      f77 -C -o norm/normal normal.o porb.o poten2.o ch.o rs.o rsb.o

normal.o: param.file normal.f
porb.o: porb.f
poten2.o: poten2.f
ch.o: ch.f
rs.o: rs.f
rsb.o: rs.o
```

6.3.2. File "normal.f"

```
      program normal
*
*  Normal mode analysis
*
      include 'param.file'

      parameter(ndm=150,nmmax=18)
      character*30 infile,anfile,outfile,outfile2,outwf,outdvr,
     &  zfile,stkfile,fmout,fminfile
      character*80 comment
      double precision pa(10),px(10),p1(10),p2(10),q(10)
      integer iarr(3)
```

```
      integer nmode(natmaxc*3-6)
      double precision amass(natmaxc*3),fmat(natmaxc*3,natmaxc*3),
     &  co(natmaxc*3),co2(natmaxc*3),co3(natmaxc*3),
     &  lambda(natmaxc*3),fv1(natmaxc*3),
     &  fv2(natmaxc*3),evec(natmaxc*3,natmaxc*3),co1(natmaxc*3),
     &  cdis(natmaxc*3),qdis(natmaxc*3),zpedvr(natmaxc*3-6),
     &  frqdvr(natmaxc*3-6),alamhan(natmaxc*3-6)
      double precision rdvr(ndm+1),tmat(ndm+1,ndm+1),
     &  akin(ndm+1,ndm+1),hamdvr(ndm+1,ndm+1),evalm(ndm+1),
     &  evalm2(ndm+1),evecm(ndm+1,ndm+1),evecm2(ndm+1,ndm+1),
     &  fv1m(ndm+1),fv2m(ndm+1),vdvr(ndm+1),vdvr2(ndm+1)
      double precision wfs(nmmax,ndm+1,ndm+1),wfs2(nmmax,ndm+1,ndm+1),
     &  evs(nmmax,ndm+1),evs2(nmmax,ndm+1)
      double precision fcfs(nmmax,ndm+1,ndm+1),sticks(2,1000),
     &  totfcf(nmmax,ndm+1)
      integer nmd(2,nmmax,1000)

      common/anparams/polind,betaexq,cutexq,soconst,
     & pa,px,p1,p2,q,polrg,polx,pqrg,pqx,theta6,c9anion,
     & c9neut,nrg

      write(*,*) 'Enter comment line'
      read(*,80) comment
80    format(a80)
      write(*,*) 'Enter mode: 1 anion or neutral frequencies.'
      write(*,*) '            2 both, and fcfs (shifted normal coords)'
      write(*,*) '            3   "       "    (vertical normal
coords)'
      read(*,*) ndo
      write(*,*) 'Enter 1 to calculate force mat; 2 to read from file'
      read(*,*) nfm
      write(*,*) 'Enter number of rare gas atoms:'
      read (*,*) nrg
      if ((ndo.gt.1).and.(nrg.gt.7)) then
        pause 'cant do fcfs for > 18 modes'
      endif
      write(*,*) 'Enter 1 for anion, 2 for neutral (detrimines which'
      write(*,*) 'NCs used in fcf mode)'
      read (*,*) nhalpot
      write(*,*) 'Enter neutral eigenvalue (1 or 2 X; 3,4 I; 5,6 II)'
      read(*,*) neigval
      write(*,*) 'Enter 1 for 3-body induction (old), 0 for not'
      read(*,*) indflag
      write(*,*) 'Enter 1 for exchange quadrupole (old), 0 for not'
      read(*,*) iexqflag
      write(*,*) 'Enter 1 for exchange dipoles, 0 for not'
      read(*,*) iexqd
      if (iexqd.eq.1) then
        iexqflag = 0
      endif
      write(*,*) 'Enter 1 for Gaussian exchange, 0 not'
      read(*,*) iexg
      if (iexg.eq.1) then
        iexqflag = 0
        iexqd = 0
      endif
      write(*,*) 'Enter 1 for iterated multipoles (new), 0 for not'
```

```
      read(*,*) indqi
      if (indqi.eq.1) then
        indflag = 0
      endif
      write(*,*) 'Axilrod-Teller, 1=on, 0=off:'
      read(*,*) iaxtel
      write(*,*) 'Enter name of anion coordinate file (a0/10):'
      read(*,*) anfile
      write(*,*) 'Enter name of neutral coordinate file (a0/10):'
      read(*,*) infile
      write(*,*) 'Enter name of normal coordinate output file:'
      read(*,*) outfile
      write(*,*) 'Enter name of info output file:'
      read(*,*) outfile2
      write(*,*) 'Enter name of zero point energy file:'
      read(*,*) zfile
      write(*,*) 'Enter wavefunction outfile:'
      read(*,*) outwf
      write(*,*) 'Enter dvr point outfile:'
      read(*,*) outdvr
      write(*,*) 'Enter stick spectrum outfile:'
      read(*,*) stkfile
      write(*,*) 'Force matrix save file name :'
      read(*,*) fmout
      write(*,*) 'Force matrix read file name : '
      read(*,*) fminfile
      write(*,*) 'Enter atomic spin-orbit coupling constant (eV):'
      read(*,*) soconst
      write(*,*) 'Enter rare gas polarizability, for old model
(Ang^3):'
      read(*,*) polind
      write(*,*) 'Rg polarizability, for new model (a0^3)'
      read(*,*) polrg
      write(*,*) 'Halide polarizability (a0^3)'
      read(*,*) polx
      write(*,*) 'Rg quadrupole polarizability (a0^5)'
      read(*,*) pqrg
      write(*,*) 'Halide quadrupole polarizability (a0^5)'
      read(*,*) pqx
      write(*,*) 'Enter exchange quadrupole range parameter:'
      read(*,*) betaexq
      if ((iexqd.eq.1).or.(iexg.eq.1)) then
        betaexq = betaexq*a0/10.
      endif
      write(*,*) 'Enter exchange quadrupole cotoff distance (Ang):'
      read(*,*) cutexq
      write(*,*) 'quadrupole dispersion coefficient e*a0^8'
      read(*,*) theta6
      write(*,*) 'C9 (Axilrod-Teller) for anion (eV*Ang^9):'
      read(*,*) c9anion
      write(*,*) 'C9 for neutral:'
      read(*,*) c9neut
      c9anion = c9anion/harev/a0**9 !* convert to au
      c9neut = c9neut/harev/a0**9
      write(*,*) 'Enter nmax for dvr basis:'
      read(*,*) ndvr
      write(*,*) 'Enter number of anion eigenvalues to show :'
```

```
      read(*,*) nanshow
      write(*,*) 'Enter number of neutral eigenvalues to show :'
      read(*,*) nnshow
      if (nhalpot.eq.1) then
        nshow = nanshow
        nshow2 = nnshow
      elseif (nhalpot.eq.2) then
        nshow = nnshow
        nshow2 = nanshow
      endif
      write(*,*) 'Enter vibrational temperature (K):'
      read(*,*) vtemp
      write(*,*) 'Enter origin position (cm-1):'
      read(*,*) origin
      write(*,*) 'Enter step size for 2nd derivatives (Ang):'
      read(*,*) hs
      hs = hs*10.0d0/a0
      write(*,*) 'Enter halide mass (amu):'
      read(*,*) halmass
      write(*,*) 'Enter rare gas mass (amu):'
      read(*,*) rgmass
      write(*,*) 'Enter Anion potential parameters (MMSV):'
      read(*,*) (pa(i),i=1,8)
      write(*,*) 'Enter X1/2 state potential parameters (MMSV):'
      read(*,*) (px(i),i=1,8)
      write(*,*) 'Enter I3/2 state parameters:'
      read(*,*) (p1(i),i=1,8)
      write(*,*) 'Enter II1/2 state parameters:'
      read(*,*) (p2(i),i=1,8)
      write(*,*) 'Enter RG-RG HFD-B Parameteres:'
      read(*,*) (q(i),i=1,9)
      do i=1,nrg*3-6
        alamhan(i) = 0.d0
      enddo
      write(*,*) 'Enter how many HO lambdas for dvr by hand?'
      read(*,*) nlam
      if (nlam.ne.0) then
        do i=1,nlam
          write(*,*) 'enter lambda for which mode?'
          read(*,*) nmode(i)
          write(*,*) 'enter lambda in cm-1 for mode ',nmode(i)
          read(*,*) alamhan(nmode(i))
          alamhan(nmode(i)) = (2.d0*pi*alamhan(nmode(i))/hztocm)**2
        enddo
      endif
*
*  Read in coordinates of minimum
*
      if (nhalpot.eq.1) then
        write(*,*) 'Reading anion coordinates from ',anfile
        open(1,file=anfile)
        do n=1,nrg+1
          read(1,*) junk,co(n*3-2),co(n*3-1),co(n*3)
        enddo
        close(1)
      elseif (nhalpot.eq.2) then
        write(*,*) 'Reading neutral coordinates from ',infile
```

```
        open(1,file=infile)
        do n=1,nrg+1
          read(1,*) junk,co(n*3-2),co(n*3-1),co(n*3)
        enddo
        close(1)
      endif
      if (ndo.eq.2) then
        if (nhalpot.eq.1) then
          write(*,*) 'Reading neutral coordinates from ',infile
          open(1,file=infile)
          do n=1,nrg+1
            read(1,*) junk,co2(n*3-2),co2(n*3-1),co2(n*3)
          enddo
          close(1)
        elseif (nhalpot.eq.2) then
          write(*,*) 'Reading anion coordinates from ',anfile
          open(1,file=anfile)
          do n=1,nrg+1
            read(1,*) junk,co2(n*3-2),co2(n*3-1),co2(n*3)
          enddo
          close(1)
        endif
      endif

*
*  If in fcf (2) mode orient both anion and neutral in principal
*  axis frame
*
      if (ndo.eq.2) then
        call prinaxes(co,rgmass,halmass)
        call prinaxes(co2,rgmass,halmass)
      endif
*
*  Set up masses list, masses in kg
*
      do i=1,nrg*3
        amass(i) = rgmass*amutokg
      enddo
      do i=nrg*3+1,(nrg+1)*3
        amass(i) = halmass*amutokg
      enddo
*
*  Set up force matrix (mks units)
*  using central difference approximation to force constants;
*  Diagonals:
*   d^2f/dx^2 = [f(x+h)-2f(x)+f(x-h)]/ h^2
*  Off diagonals:
*   d^2f/dxdy = [f(x+h,y+h)-f(x-h,y+h)-f(x+h,y-h)+f(x-h,y-h)]/(2h)^2
*
      if (nfm.eq.1) then

        if (nhalpot.eq.1) then            !* Anion

          anp = apeval(co)
          if (ndo.eq.2) then
            anp2 = peval(co2)
          elseif (ndo.eq.3) then
```

```
          anp2 = peval(co)
        endif
        write(*,*) 'Setting up force matrix'
        nelem = (nrg+1)*3 + ((nrg+1)*3-1)**2/2
        write(*,*) nelem,' elements to calculate'
        neldone = 0
        do i = 1,(nrg+1)*3  !* Diagonals
          savei = co(i)
          fmat(i,i) = -2.0d0*apeval(co)
          co(i) = savei + hs
          fmat(i,i) = fmat(i,i) + apeval(co)
          co(i) = savei - hs
          fmat(i,i) = (fmat(i,i) + apeval(co))*evtoj
     &                 /(hs*a0/10.0d0*1.0d-10)**2/amass(i)
          co(i) = savei
          neldone = neldone+1
          if (mod(neldone,20).eq.0) then
            write(*,7593) neldone*100/nelem
          endif
        enddo

7593    format(i3,'% done')

        do i=2,(nrg+1)*3    !* Off diagonals (lower triangle only)
          do j=1,i-1
            savei = co(i)
            savej = co(j)
            co(i) = co(i) + hs
            co(j) = co(j) + hs
            fmat(i,j)=apeval(co)
            co(i) = savei - hs
            fmat(i,j)=fmat(i,j)-apeval(co)
            co(i) = savei + hs
            co(j) = savej - hs
            fmat(i,j)=fmat(i,j)-apeval(co)
            co(i) = savei - hs
            fmat(i,j)=(fmat(i,j)+apeval(co))*evtoj
     &                 /(2.0d0*hs*a0/10.0d0*1.0d-10)**2
     &                 /dsqrt(amass(i)*amass(j))
            co(i)=savei
            co(j)=savej
            neldone = neldone+1
            if (mod(neldone,20).eq.0) then
              write(*,7593) neldone*100/nelem
            endif
          enddo
        enddo

      elseif (nhalpot.eq.2) then  !* Neutral

        anp = peval(co)
        if (ndo.eq.2) then
          anp2 = apeval(co2)
        elseif (ndo.eq.3) then
          anp2 = apeval(co)
        endif
        write(*,*) 'Setting up force matrix'
```

```
      do i = 1,(nrg+1)*3  !* Diagonals
        savei = co(i)
        fmat(i,i) = -2.0d0*peval(co)
        co(i) = savei + hs
        fmat(i,i) = fmat(i,i) + peval(co)
        co(i) = savei - hs
        fmat(i,i) = (fmat(i,i) + peval(co))*evtoj
&                    /(hs*a0/10.0d0*1.0d-10)**2/amass(i)
        co(i) = savei
      enddo

      do i=2,(nrg+1)*3    !* Off diagonals (lower triangle only)
        do j=1,i-1
          savei = co(i)
          savej = co(j)
          co(i) = co(i) + hs
          co(j) = co(j) + hs
          fmat(i,j) = peval(co)
          co(i) = savei - hs
          fmat(i,j)=fmat(i,j)-peval(co)
          co(i) = savei + hs
          co(j) = savej - hs
          fmat(i,j)=fmat(i,j)-peval(co)
          co(i) = savei - hs
          fmat(i,j)=(fmat(i,j)+peval(co))*evtoj
&                   /(2.0d0*hs*a0/10.0d0*1.0d-10)**2
&                   /dsqrt(amass(i)*amass(j))
          co(i)=savei
          co(j)=savej
        enddo
      enddo

    endif

    open(1,file=fmout)          !* Save force matrix to file
    do i = 1,(nrg+1)*3
      do j = 1,i
        write(1,*) fmat(i,j)
      enddo
    enddo
    close(1)
    write(*,*) 'Force matrix saved to ',fmout

  elseif (nfm.eq.2) then

    if (nhalpot.eq.1) then            !* Anion
      anp = apeval(co)
      if (ndo.eq.2) then
        anp2 = peval(co2)
      elseif (ndo.eq.3) then
        anp2 = peval(co)
      endif
    elseif (nhalpot.eq.2) then  !* Neutral
      anp = peval(co)
      if (ndo.eq.2) then
        anp2 = apeval(co2)
      elseif (ndo.eq.3) then
```

```
            anp2 = apeval(co)
          endif
        endif

        open(1,file=fminfile)        !* Read force matrix from file
        do i=1,(nrg+1)*3
          do j = 1,i
            read(1,*) fmat(i,j)
          enddo
        enddo
        close(1)
        write(*,*) 'Force matrix read from ',fminfile

      endif

*
*  Diagonalize force matrix
*
      write(*,*) 'Diagonalizing force matrix...'
      call rs(natmaxc*3,(nrg+1)*3,fmat,lambda,1,evec,fv1,fv2,ierr)
      write(*,*) 'Normal frequencies : '
      do i=1,(nrg+1)*3
        write(*,*) lambda(i)
      enddo
*
*  If in fcf mode, determine the displacements of anion or neutral
*  from the one whose normal coordinates are being used, first in
*  cartesian displacedments and then transform to normal coordinates
*
      if (ndo.eq.2) then

        do i=1,(nrg+1)*3             !*  Loop over cartesian coordinates
         cdis(i) = co2(i) - co(i)
        enddo

        do m=1,6
          qdis(m) = 0.0d0
        enddo
        do m=7,(nrg+1)*3             !*  Transform to normal coordinates
          qdis(m) = 0.0d0
          do i=1,(nrg+1)*3
            qdis(m) = qdis(m) + evec(i,m)*cdis(i)*dsqrt(amass(i))
*                       !*  Units sqrt(kg)*a0/10
          enddo
        enddo

      endif
*
*  Set up dvr points and kinetic matrix, which are the same
*  for all the normal modes neglecting the frequencies
*
      call hopts(ndvr,rdvr,tmat)
      call hokin(ndvr,tmat,akin)
*
*  Evaluate potential at dvr points for each normal mode, and
*  diagonalize dvr hamiltonian
*
```

```
      open(1,file=outwf)
      open(2,file=outdvr)

      do 2500 m = 7,(nrg+1)*3 !* Loop over normal modes

        write(*,*) 'Mode ',m-6
        write(1,*) 'Mode ',m-6
        write(1,*)
        write(2,*) 'Mode ',m-6

        if (ndo.eq.1) then
          write(2,*) 'Q [sqrt(amu)*Ang]    V(Q) [eV]'
        elseif (ndo.eq.2) then
          write(2,9310)
9310      format('Q1',15x,'V1(Q1)',11x,'Q2',15x,'V2(Q2)')
        elseif (ndo.eq.3) then
          write(2,9320)
9320      format('Q1',15x,'V1(Q1)',15x,'V2(Q1)')
        endif

        if (alamhan(m-6).eq.0.d0) then
          alambda = lambda(m)
        else
          alambda = alamhan(m-6)
        endif

        do i = 1,ndvr+1       !* Loop over dvr points
          qm = rdvr(i)*dsqrt(hbar/(dsqrt(alambda)*2.0d0))
     &         *10.0d10/a0    !* Normal coord units converted from
*                     !* sqrt(kg)*m to sqrt(kg)*a0/10

          if (ndo.eq.2) then
            qm2 = qm - qdis(m)
          endif

          do j=1,(nrg+1)*3    !* Loop over cartesian coordinates
            co1(j) = co(j) + evec(j,m)*qm/dsqrt(amass(j))   !* other
q's=0
            if (ndo.eq.2) then
              co3(j) = co2(j) + evec(j,m)*qm2/dsqrt(amass(j))
            endif
          enddo

          if (nhalpot.eq.1) then
            vdvr(i) = apeval(co1)
          elseif (nhalpot.eq.2) then
            vdvr(i) = peval(co1)
          endif
          if (ndo.eq.2) then        !* Displaced
            if (nhalpot.eq.1) then
              vdvr2(i) = peval(co3)
            elseif (nhalpot.eq.2) then
              vdvr2(i) = apeval(co3)
            endif
          elseif (ndo.eq.3) then
            if (nhalpot.eq.1) then  !* Vertical
              vdvr2(i) = peval(co1)
```

```
          elseif (nhalpot.eq.2) then
            vdvr2(i) = apeval(co1)
          endif
        endif

        if (ndo.eq.1) then
          write(2,578) qm*a0/10./sqrt(amutokg),vdvr(i)
        elseif (ndo.eq.2) then
          write(2,579) qm*a0/10./sqrt(amutokg),vdvr(i),
     &      qm2*a0/10./sqrt(amutokg),vdvr2(i)
        elseif (ndo.eq.3) then
          write(2,579) qm*a0/10./sqrt(amutokg),vdvr(i),vdvr2(i)
        endif

      enddo           !* End loop over dvr points

      write(2,*)
      call hoham(akin,vdvr,ndvr,alambda,hamdvr)
      if (nhalpot.eq.1) then
        write(*,*) 'Diagonalizing Anion Hamiltonian'
      elseif (nhalpot.eq.2) then
        write(*,*) 'Diagonalizing Neutral Hamiltonian'
      endif
      call rs(ndm+1,ndvr+1,hamdvr,evalm,1,evecm,fv1m,fv2m,ierr)
      do kk = 1,nshow
        do ll = 1,ndvr+1
          wfs(m-6,ll,kk) = evecm(ll,kk)
        enddo
        evs(m-6,kk) = evalm(kk) - anp
      enddo

      if (ndo.ge.2) then
        call hoham(akin,vdvr2,ndvr,alambda,hamdvr)
        call rs(ndm+1,ndvr+1,hamdvr,evalm2,1,evecm2,fv1m,fv2m,ierr)
        do kk = 1,nshow2
          do ll = 1,ndvr+1
            wfs2(m-6,ll,kk) = evecm2(ll,kk)
          enddo
          evs2(m-6,kk) = evalm2(kk) - anp2
        enddo
      endif

      if (nhalpot.eq.1) then
        write(1,*) '*********Anion**********'
      else
        write(1,*) '*********Neutral**********'
      endif
      write(1,*) 'Eigenvalues'
      do i = 1,nshow
        write(1,*) (evalm(i)-anp)*evtocm
      enddo
      write(1,*)
      zpedvr(m-6) = evalm(1) - anp
      frqdvr(m-6) = evalm(2) - evalm(1)
      write(1,*) 'Wavefunctions'
      call showarr4(1,ndm+1,evecm,ndvr+1,nshow)
      write(1,*)
```

```
        if (ndo.ge.2) then
          if (nhalpot.eq.1) then
            write(1,*) '*********Neutral**********'
          else
            write(1,*) '*********Anion**********'
          endif
          write(1,*) 'Eigenvalues'
          do i = 1,nshow2
            write(1,*) (evalm2(i)-anp2)*evtocm
          enddo
          write(1,*)
          write(1,*) 'Wavefunctions'
          call showarr4(1,ndm+1,evecm2,ndvr+1,nshow2)
          write(1,*)
        endif

2500  continue          !* End loop over normal modes

578   format(g16.8,1x,g16.8)
579   format(g16.8,3(1x,g16.8))
      close(1)
      close(2)

*
*  Calculate fcfs
*
      if (ndo.ge.2) then
        if (nhalpot.eq.1) then
          call calcfcfs(ndvr,nanshow,nnshow,vtemp,origin,evs,wfs,
     &                evs2,wfs2,fcfs,totfcf,nsticks,nmd,sticks)
        elseif (nhalpot.eq.2) then
          call calcfcfs(ndvr,nanshow,nnshow,vtemp,origin,evs2,wfs2,
     &                evs,wfs,fcfs,totfcf,nsticks,nmd,sticks)
        endif
      endif
*
*
*  Output to files
*
      open(1,file=outfile2)
      open(2,file=zfile)
      write(1,*) 'Normal Coordinate Program'
      call idate(iarr)
      write(1,300) iarr(2),iarr(1),iarr(3)
      call itime(iarr)
      write(1,301) iarr(1),iarr(2),iarr(3)
300   format('Date: ',i2,'/',i2,1x,i4)
301   format('Time: ',i2,':',i2,':',i2)
      write(1,*)
      write(1,*) comment
      write(1,*)
      write(1,*) 'Number of RGs: ',nrg
      write(1,7356) anfile,infile
7356  format('Eq pos files (an, neut): ',
     & a30,a30)
      write(1,*) 'Normal coordinate file: ',outfile
```

```
      write(1,*) 'Output file: ',outfile2
      write(1,*) 'Zero point energy file: ',zfile
      write(1,*) 'Wavefunction save file: ',outwf
      write(1,*) 'DVR point save file: ',outdvr
      if (nfm.eq.1) then
        write(1,*) 'Force matrix save file: ',fmout
      elseif (nfm.eq.2) then
        write(1,*) 'Force matrix imput file: ',fminfile
      endif
      if (ndo.ge.2) then
        if (nhalpot.eq.1) then
          write(1,3023)
        else
          write(1,3024)
        endif
        if (ndo.eq.2) then
          write(1,*) ' -- displaced coordinates for other'
        elseif (ndo.eq.3) then
          write(1,*) ' -- vertical coordinates for other'
        endif
        write(1,*) 'vibrational stick file: ',stkfile
        write (1,3030) vtemp,origin
      endif
3023  format(/,'FCF mode -- Using ANION normal coordinates',$)
3024  format(/,'FCF mode -- Using NEUTRAL normal coordinates',$)
3030  format('vtemp (K) : ',f10.3,5x,'origin(cm-1) : ',f15.5)
      write(1,*)
      if ((nhalpot.eq.2).or.(ndo.ge.2)) then
        if ((neigval.eq.1).or.(neigval.eq.2)) then
          write(1,*) 'X State'
        elseif ((neigval.eq.3).or.(neigval.eq.4)) then
          write(1,*) 'I State'
        elseif ((neigval.eq.5).or.(neigval.eq.6)) then
          write(1,*) 'II State'
        endif
        write(1,95) soconst
95      format('SO constant (eV) :',f12.8)
      endif
      if ((nhalpot.eq.1).or.(ndo.ge.2)) then
        if (indflag.eq.1) then
          write(1,*) 'Three body induction ON'
          write(1,305) polind
305       format('Rare gas polarizability: ',f10.6,' A^3')
        endif
        if (iexqflag.eq.1) then
          write(1,*) 'Exchange quadrupole ON'
          write(1,307) betaexq,cutexq
307       format('Exchange quadrupole beta (A^-1):',f10.6,3x,
     &           'Cutoff (A):',f10.6)
        endif
        if (indqi.eq.1) then
          write(1,*) 'Iterated multipoles ON'
          write(1,565) polrg,polx,pqrg,pqx
565       format('Dipole polarizabilities, a0^3: Rg = ',f10.6,3x,
     &      'Hal = ',f10.6,/,'Quad. polarizabilities, a0^5: Rg = ',
     &      f10.6,3x,'Hal = ',f10.6)
        endif
```

```
        if (iexqd.eq.1) then
          write(1,573) betaexq*10./a0,theta6
573      format('Exchange dipoles ON',3x,'betaexq =',f10.6,3x,
     &    'theta6 =',f10.3)
        endif
        if (iexg.eq.1) then
          write(1,689) betaexq,theta6
689       format('Gaussian exchange ON,',3x,'betaexq =',f10.6,
     &    3x,'theta6 =',f10.3)
        endif
       endif
       if (iaxtel.eq.1) then
         write(1,695) c9anion*harev*a0**9,c9neut*harev*a0**9
695      format('Axilrod-Teller ON, ',3x,'c9anion =',f10.3,
     &  3x,'c9neut = ',f10.3)
       endif
       write(1,400) ndvr,nanshow,nnshow,hs*a0/10.
400    format('ndvr: ',i3,3x,'nanshow: ',i2,'nnshow:',i3,
     &  3x,'Step size (Ang.): ',g10.4)
       write(1,405) halmass,rgmass
405    format('Halogen mass (amu): ',f15.10,3x,'Rare gas mass: ',f15.10)
       write(1,1099) a0,harev,evtocm
       write(1,1100) evtoj,hztocm,amutokg
1099   format('a0 = ',g15.10,' harev = ',g15.10,'evtocm = ',g15.10)
1100   format('evtoj = ',g16.10,' hztocm = ',g16.10,' amutokg =
',g16.10)
       if ((nhalpot.eq.1).or.(ndo.ge.2)) then
         write(1,*) 'Anion rg-x MMSV parameters:'
         write(1,100) (pa(i),i=1,8)
       endif
       if ((nhalpot.eq.2).or.(ndo.ge.2)) then
       write(1,*) 'Neutral rg-x MMSV Parameters (X,I,II diatom states):'
         write(1,100) (px(i),i=1,8)
         write(1,100) (p1(i),i=1,8)
         write(1,100) (p2(i),i=1,8)
       endif
       write(1,*) 'RG-RG HFD-B Parameters:'
       write(1,100) (q(i),i=1,9)
100    format(2x,5g14.8,/,2x,5g14.8)
       write(1,*)
       write(1,200)
200    format(80('*'),/)
       if (nhalpot.eq.1) then
         write(1,340) anp
       else
         write(1,341) anp
       endif
       if (ndo.ge.2) then
         if (nhalpot.eq.2) then
           write(1,340) anp2
         else
           write(1,341) anp2
         endif
       endif
*
*  Write zero point energies and vib frequencies to file in mode 1
*
```

```
      if (ndo.eq.1) then
        if (nhalpot.eq.1) then
          write(1,*) 'Anion'
        else
          write(1,*) 'Neutral'
        endif
        write(1,330)
        zpe = 0.0d0
        do i=1,(nrg+1)*3
          if (lambda(i).lt.0) then
            freq = 0.
          else
            freq = dsqrt(lambda(i))/(2.0d0*pi)
            zpe = zpe + freq/2.0d0
          endif
          if (i.le.6) then
            write(1,320)lambda(i),freq*hztocm
          elseif (alamhan(i-6).eq.0) then
            write(1,322) i-6,lambda(i),freq*hztocm,zpedvr(i-6)*evtocm,
     &        frqdvr(i-6)*evtocm
          else
            write(1,323) i-6,lambda(i),freq*hztocm,zpedvr(i-6)*evtocm,
     &        frqdvr(i-6)*evtocm,dsqrt(alamhan(i-6))/(2.*pi)*hztocm
          endif
        enddo
        write(1,*)
        write(1,325) zpe*hztocm
        write(1,326) zpe*hztocm/evtocm
        write(1,*)
        zped = 0.0d0
        do i=1,(nrg+1)*3-6
          zped = zped + zpedvr(i)
        enddo
        write(1,327) zped*evtocm
        write(1,328) zped
        write(2,329) nrg,zped
        write(1,*)

340   format('Anion potential at eq. (eV): ',f15.10)
341   format('Neutral potential at eq. (eV): ',f15.10)
330   format(' Mode',1x,'Lambda (s-2)',2x,'Harm.freq(cm-1)',2x,
     & 'ZPE DVR',4x,'v1-v0 DVR ')
320   format(5x,g13.5,3x,f7.2)
322   format(i3,2x,e13.5,3x,f7.2,7x,f7.2,5x,f7.2)
323   format('*',i2,2x,e13.5,3x,f7.2,7x,f7.2,5x,f7.2,1x,
     & '(basis freq = ',f6.2,')')
325   format('Harmonic zero point energy (cm-1): ',f7.2)
326   format('                              (eV): ',f15.10)
327   format('DVR zero point energy (cm-1): ',f7.2)
328   format('                          (eV): ',f15.10)
329   format(i3,1x,g18.10)
*
*  Write vibrational eigenvalues and single mode fcfs to file in fcf
mode
*
      elseif (ndo.ge.2) then
        do m=1,(nrg+1)*3-6    !* Loop over modes
```

```
      write(1,605) m
      if (ndo.eq.2) then
        write(1,640) qdis(m+6)*a0/10./sqrt(amutokg)
      endif
      write(1,607)
      if (nhalpot.eq.1) then
        write(1,510)
      else
        write(1,515)
      endif
      write(1,610)
      nshowm = max(nshow,nshow2)
      do i=1,nshowm              !* Loop over v
        if (i.gt.1) then
          elast = evs(m,i-1)
          elast2 = evs2(m,i-1)
        else
          elast = 0.d0
          elast2 = 0.d0
        endif
        if ((i.le.nshow).and.(i.le.nshow2)) then
          write(1,615) i-1,evs(m,i)*evtocm,(evs(m,i)-elast)
&           *evtocm,i-1,evs2(m,i)*evtocm,
&           (evs2(m,i)-elast2)*evtocm
        elseif (i.le.nshow) then
          write(1,620) i-1,evs(m,i)*evtocm,(evs(m,i)-elast)*evtocm
        elseif (i.le.nshow2) then
          write(1,625) i-1,evs2(m,i)*evtocm,(evs2(m,i)-elast2)
&           *evtocm
        endif
      enddo
      write(1,630)
      write(1,145)
      write(1,150) (k, k = 0,nanshow-1)
      write(1,*)
      do j=1,nnshow
        write(1,155) j-1,(fcfs(m,k,j), k = 1,nanshow)
      enddo
      write(1,*)
      write(1,160) (totfcf(m,k), k=1,nanshow)
      write(1,*)
    enddo
    nmde = 3*(nrg+1)-6
    write(1,5000)          !* Write vsticks
    write(1,5005)
    write(*,*) 'nsticks',nsticks
    do i=1,nsticks
      do m=1,nmde
        write(1,5010) nmd(1,m,i)
      enddo
      if (nmde.lt.15) then
        do n=nmde+1,15
          write(1,5015)
        enddo
      endif
      do m=1,nmde
        write(1,5010) nmd(2,m,i)
```

```
          enddo
          if (nmde.lt.15) then
            do n=nmde+1,15
              write(1,5015)
            enddo
          endif
          write(1,5020) sticks(1,i),sticks(2,i)
        enddo

      endif
      close(1)
      close(2)

605   format(/,35('*'),' Mode ',i2,1x,36('*'),/)
607   format('Vibrational Eigenvalues :',/)
610   format(13x, 'Total', 6x, 'Spacing',20x,'Total', 6x, 'Spacing')
615   format(1x,'v = ',i2,2x,f9.2,4x,f8.2,9x,'v = ',i2,2x,
     &  f9.2,4x,f8.2)
620   format(1x,'v = ',i2,2x,f9.2,4x,f8.2)
625   format(39x,'v = ',i2,2x,f9.2,4x,f8.2)
630   format(/,'Single mode FCFs :',/)
640   format('Displacement [sqrt(amu)*Ang] : ',g12.5)
510   format(15x,'Anion',31x,'Neutral',/)
515   format(15x,'Neutral',31x,'Anion',/)
145   format(19x,'Anion v = ')
150   format(17x,20(i3,4x),/)
155   format('Neutral v = ',i3,2x,20(f6.4,1x),/)
160   format('    Total = ',5x,20(f6.4,1x),/)

5000  format(/,'Vibrational sticks')
5005  format('Anion mode',21x,'Neutral mode',17x,'Pos(cm-1)',
     & 2x,'Inten')
5010  format(i2,$)
5015  format('  ',$)
5020  format(f9.3,2x,f8.6)
*
*  Write out normal frequencies (lambdas) and coordinates
*
      open(1,file=outfile)
      do i=7,(nrg+1)*3-6
        write(1,*) lambda(i)
      enddo
      do i=1,(nrg+1)*3
        write(1,*) (evec(i,j),j=7,(nrg+1)*3)
      enddo
      close(1)
*
*  Write vibrational stick file if in mode 2 or 3
*
      if (ndo.ge.2) then
        open(1,file=stkfile)
        do i=1,nsticks
          write(1,4000) sticks(1,i),sticks(2,i)
        enddo
        close(1)
      endif
4000  format(f10.3,f10.6)
```

```
      end


*=====================================================================
      double precision function peval(coord)
*=====================================================================
*
*   Evaluate neutral potential -- coords in a0/10, poten in eV
*
      include 'param.file'
      double precision hfd_b
      double precision rx(natmaxc),ry(natmaxc),rz(natmaxc),
     &  pa(10),px(10),p1(10),p2(10),q(10),potnrgx(6)
      double precision coord(natmaxc*3)

      common/anparams/polind,betaexq,cutexq,soconst,
     & pa,px,p1,p2,q,polrg,polx,pqrg,pqx,theta6,c9anion,
     & c9neut,nrg
*
      ncluster = nrg

      do i = 1,nrg+1
        rx(i) = coord(i*3-2)
        ry(i) = coord(i*3-1)
        rz(i) = coord(i*3)
      enddo

      potat = 0.
      if (iaxtel.eq.1) then
        call vat(c9neut,rx,ry,rz,potat)
        potat = potat*harev/100.
      endif

      do i = 1,nrg+1
        rx(i) = coord(i*3-2)*a0/10.0d0          !* Convert to Ang
        ry(i) = coord(i*3-1)*a0/10.0d0
        rz(i) = coord(i*3)*a0/10.0d0
      enddo

      call porb(nrg,rx,ry,rz,px,p1,p2,soconst,potnrgx)

      rgrg = 0.
      do i=2,nrg
        do j=1,i-1
          rt = dsqrt((rx(i)-rx(j))**2 + (ry(i)-ry(j))**2
     &               + (rz(i)-rz(j))**2)
          rgrg = rgrg + hfd_b(q(1),q(2),q(3),q(4),
     &                  q(5),q(6),q(7),q(8),q(9),rt)
        enddo
      enddo

      peval = potnrgx(neigval) + rgrg + potat

      return
      end
```

```
*=====================================================================
      double precision function apeval(coord)
*=====================================================================
*
*   Evaluate anion potential, coord in a0/10, poten returned in eV
*
      include 'param.file'
      double precision anmmsv,hfd_b
      double precision rxin(natmaxc),ryin(natmaxc),rzin(natmaxc),
     &  rx(natmaxc),ry(natmaxc),rz(natmaxc),coord(natmaxc*3),
     &  pa(10),px(10),p1(10),p2(10),q(10),dp(natmaxc,3),
     &  dhalpair(ncl),drgpair(ncl),qhalpair(ncl),qrgpair(ncl),
     &  qp(natmaxc,3,3)

      common/anparams/polind,betaexq,cutexq,soconst,
     & pa,px,p1,p2,q,polrg,polx,pqrg,pqx,theta6,c9anion,
     & c9neut,nrg

      ncluster = nrg
*
      do i = 1,nrg+1
        rxin(i) = coord(i*3-2)
        ryin(i) = coord(i*3-1)
        rzin(i) = coord(i*3)
      enddo
*
*  Calculate three body terms
*
      anpotind = 0.d0
      anpotexq = 0.d0
      anpotat = 0.d0
      if ((indflag.eq.1).or.(iexqflag.eq.1)) then
        do i=2,nrg
          do j=1,i-1
            if (indflag.eq.1) then
             call pind(polind,rxin(nrg+1),ryin(nrg+1),rzin(nrg+1),
     &        rxin(i),ryin(i),rzin(i),rxin(j),ryin(j),rzin(j),
     &        dpind)
             anpotind = anpotind + dpind
            endif
            if (iexqflag.eq.1) then
             call vexq(betaexq,cutexq,rxin(nrg+1),ryin(nrg+1),
     &        rzin(nrg+1),rxin(i),ryin(i),rzin(i),rxin(j),
     &        ryin(j),rzin(j),dpexq)
             anpotexq = anpotexq + dpexq
            endif
          enddo
        enddo
       endif
       anpotind = anpotind*harev/1.d2
       anpotexq = anpotexq*harev/1.d2
*
*  Calculate new induction & exchange dipole potentials
*
       if (indqi.eq.1) then
         call vinddq(polrg,polx,pqrg,pqx,rxin,ryin,rzin,dp,dhalpair,
```

```
     &    drgpair,qhalpair,qrgpair,qp,vcd,vcq,anpotind)
          anpotind = anpotind*harev/1.d2
        endif
        if (iexqd.eq.1) then
          call veqd(betaexq,theta6,rxin,ryin,rzin,anpotexq)
          anpotexq = anpotexq*harev/1.d2
        elseif (iexg.eq.1) then
          call exg(betaexq,theta6,rxin,ryin,rzin,anexx,anddisp,
     &  anqdisp,anpotexq)
          anpotexq = anpotexq*harev/1.d2
        endif
        if (iaxtel.eq.1) then
          call vat(c9anion,rxin,ryin,rzin,anpotat)
          anpotat = anpotat*harev/1.d2
        endif
*
*  Convert from a0/10 to angstroms
*
        do n=1,nrg+1
          rx(n) = rxin(n) * a0/10.
          ry(n) = ryin(n) * a0/10.
          rz(n) = rzin(n) * a0/10.
        enddo
*
*  Calculate anion potential
*
        xhal = rx(nrg+1)
        yhal = ry(nrg+1)
        zhal = rz(nrg+1)

        anpotrgx = 0.
        do i=1,nrg
          rt = dsqrt((rx(i)-xhal)**2 + (ry(i)-yhal)**2
     &             + (rz(i)-zhal)**2)
          anpotrgx = anpotrgx + anmmsv(pa(1),pa(2),pa(3),pa(4),
     &                         pa(5),pa(6),pa(7),pa(8),rt)
        enddo

        anpotrgrg = 0.
        do i=2,nrg
          do j=1,i-1
            rt = dsqrt((rx(i)-rx(j))**2 + (ry(i)-ry(j))**2
     &                + (rz(i)-rz(j))**2)
            anpotrgrg = anpotrgrg + hfd_b(q(1),q(2),q(3),q(4),
     &                   q(5),q(6),q(7),q(8),q(9),rt)
          enddo
        enddo

        apeval = anpotrgx+anpotrgrg+anpotind+anpotexq+anpotat

        return
        end

*=============================================================
        subroutine pind(pol,x0,y0,z0,x1,y1,z1,x2,y2,z2,potind)
*=============================================================
*
```

```
*  Charge-Ind Dipole-Ind Dipole Three body potential
*
*  Input:   pol         Rare gas polarizibility (A^3)
*           x0,y0,z0    Halide cartesian coordinates (a0/10)
*           x1,y1,z1, x2,y2,z2      Rare gas coordinates (a0/10)
*
*  Output:  potind            Potential (hartree/100)
*
      implicit double precision(a-h,o-z)
      parameter(pi=3.14159265359d0,a0=0.529177249d0,
     &  harev=27.2113961d0,evtocm=8065.5410d0)

      const = (pol/(a0**3)*1.d3)**2


      r10sqr = (x1-x0)**2+(y1-y0)**2+(z1-z0)**2
      r20sqr = (x2-x0)**2+(y2-y0)**2+(z2-z0)**2
      r12sqr = (x1-x2)**2+(y1-y2)**2+(z1-z2)**2

      dot1020 = (x1-x0)*(x2-x0)+(y1-y0)*(y2-y0)+(z1-z0)*(z2-z0)
      dot1012 = (x1-x0)*(x1-x2)+(y1-y0)*(y1-y2)+(z1-z0)*(z1-z2)
      dot1220 = (x1-x2)*(x2-x0)+(y1-y2)*(y2-y0)+(z1-z2)*(z2-z0)

      t1 = dsqrt(r10sqr)
      t2 = r10sqr**2
      t6 = dsqrt(r12sqr)
      t7 = r12sqr**2
      t10 = dsqrt(r20sqr)
      t11 = r20sqr**2
      potind = const*t1/t2*t6/t7*t10/t11*(dot1020-
     & 3*dot1012*dot1220/r12sqr)
      potind = potind*1.d3

      return
      end

*=====================================================================
      subroutine vexq(betain,cutoff,x0in,y0in,z0in,
     &  x1in,y1in,z1in,x2in,y2in,z2in,pexq)
*=====================================================================
*
*  Charge-exchange quadrupole interaction.
*  Gaussian 1-electron model
*
*  Input:   betain            Gaussian range parameter (Ang^-1)
*           cutoff            RG-RG distance cutoff (Ang)
*           x0in,y0in,z0      Halide cartesian coordinates (a0/10)
*           x1,y1,z1    Rare gas 1 cart. coords. (a0/10)
*           x2,y2,z2    Rare gas 2 cart. coords. (a0/10)
*
*  Returns: pexq        potential in hartree/100
*

      implicit double precision(a-h,o-z)
      parameter(pi=3.14159265359d0,a0=0.529177249d0,
     &  harev=27.2113961d0,evtocm=8065.5410d0)
```

```
      x0 = x0in/10.0d0  !* Convert to usual atomic units
      y0 = y0in/10.0d0
      z0 = z0in/10.0d0
      x1 = x1in/10.0d0
      y1 = y1in/10.0d0
      z1 = z1in/10.0d0
      x2 = x2in/10.0d0
      y2 = y2in/10.0d0
      z2 = z2in/10.0d0

      beta = betain*a0

      r12sqr = (x1-x2)**2+(y1-y2)**2+(z1-z2)**2
      if (r12sqr.gt.(cutoff/a0)**2) then
        pexq = 0.
        return
      endif
      rcsqr = (x1/2.-x0+x2/2.)**2+(y1/2.-y0+y2/2.)**2
     &        +(z1/2.-z0+z2/2.)**2
      dotc12 = (x1/2-x0+x2/2)*(x1-x2)+(y1/2-y0+y2/2)*(y1-y2)
     &         +(z1/2-z0+z2/2)*(z1-z2)
      ex = exp(-r12sqr*beta**2/2.d0)
      quad = -r12sqr*ex/(1.d0-ex)/2
      rcthir = rcsqr**1.5d0
      pexq = -(quad/rcthir)*(3.d0*dotc12**2/rcsqr/r12sqr-1.d0)/2.d0
*
* Result is now in atomic units, convert hartree/100
*
      pexq = pexq*100.0d0
      return
      end

*
*================================================
      subroutine hopts(nmax,r,tl)
*================================================
*
*  Calculate dvr points for harmonic oscillator eigenfunction
*  basis
*
*  Input:   nmax
*
*  Output:  r()   dvr points (without hbar/2*sqrt(lambda) factor)
*           tl()  transformation matrix
*
*
      include 'param.file'

      parameter(ndm=150)
      integer nmax
      double precision r(ndm+1),tl(ndm+1,ndm+1)
      double precision u(ndm+1,ndm+1),fv1(ndm+1),
     &  fv2(ndm+1)

      write(*,*) 'hopts'
*
*  Zero diagonal elements
```

```
*
      do 100 i=0,nmax
        u(i+1,2)=0.
100   continue
*
*  Off diagonal elements
*
      u(1,1)=0.0d0
      do 200 i=1,nmax
        u(i+1,1)=sqrt(dble(i))
200   continue

      call rsb(ndm+1,nmax+1,2,u,r,1,tl,fv1,fv2,ierr)
      write(*,*) 'ierr=',ierr

      return
      end

*=======================================================
      subroutine hokin(nmax,tl,ak)
*=======================================================
*  Find kinetic matrix in HO basis:
*  <n'|(-hbar^2/2)d^2/dQ^2|n>, which has diagonal elements:
*  2n+1 (n=0..nmax) and diagonal-2 elements sqrt[n(n-1)]
*  (n=2..nmax), then transform to dvr
*
*  Input:   nmax  largest n in basis
*           tl    transformation matrix from hopts
*
*  Output:  ak    kinetic matrix in dvr basis, without
*                 omega*hbar/4 factor (lower triangle)
*                 [omega = sqrt(lambda)]
*
      include 'param.file'

      parameter(ndm=150)
      integer nmax
      double precision tl(ndm+1,ndm+1),ak(ndm+1,ndm+1),
     &  adiag(ndm+1),adiag2(ndm-1),tlt(ndm+1,ndm+1),
     &  cmat(ndm+1,ndm+1)

      write(*,*) 'hokin'

      do i=0,nmax
        di = dble(i)
        adiag(i+1) = 2*di+1
        if (i.ge.2) then
          adiag2(i+1) = -dsqrt(di*(di-1))
        else
        endif
      enddo
*
*  Zero K matrix
*
      do i=1,nmax+1
        do j=1,nmax+1
          ak(i,j) = 0.0d0
```

```
        enddo
      enddo
*
*  Transform to dvr:
*  K(dvr) = T(trans)K(fbr)T
*
      do i=1,nmax+1
        ak(i,i)=adiag(i)
        if (i.gt.2) then
          ak(i,i-2)=adiag2(i)
          ak(i-2,i)=adiag2(i)
        endif
      enddo

      call transpos(ndm+1,tl,nmax+1,nmax+1,tlt)
      call mms(ndm+1,nmax+1,tlt,ak,cmat)
      call mms(ndm+1,nmax+1,cmat,tl,ak)

      return
      end

*============================================================
      subroutine hoham(ak,vr,nmax,alambda,ham)
*============================================================
*
*  Set up hamiltonian in HO/dvr basis
*
*  Input:   ak()  kinetic energy matrix (without hbar*omega/4)
*           vr()  potential evaluated at dvr points (eV)
*           nmax
*           alambda  = (2*pi*nu)^2 in s^-2
*
*  Output:  ham() hamiltonian in dvr basis (lower triangle)
*
      include 'param.file'

      parameter(ndm=150)
      integer nmax
      double precision ak(ndm+1,ndm+1),vr(ndm+1),alambda,
     &  ham(ndm+1,ndm+1)

      do i=1,nmax+1            !* Zero Ham matrix
        do j=1,nmax+1
          ham(i,j) = 0.0d0
        enddo
      enddo

      factor=hbarev*dsqrt(alambda)/4.0d0

      do 200 n=1,nmax+1
        do 100 m=1,n           !* Lower triangle only needed by rs
          ham(n,m)=factor*ak(n,m)
100     continue
200   continue

      do 300 i=1,nmax+1
        ham(i,i)=ham(i,i)+vr(i)
```

```
300   continue

      return
      end

*=================================================================
      subroutine prinaxes(coordin,rgmass,halmass)
*=================================================================
*
*  Transform input coordinates to principal axis system
*
      include 'param.file'
      parameter (x=1,y=2,z=3,tiny=1.0d-2)
      double precision coordin(natmaxc*3)
      double precision rgmass,halmass
      double precision rx(natmaxc),ry(natmaxc),rz(natmaxc),
     &  rh(natmaxc-1),th(natmaxc-1),ph(natmaxc-1),ti(3,3),
     &  prmoms(3),prcos(3,3),fv1(3),fv2(3)
      double precision pa(10),px(10),p1(10),p2(10),q(10)

      common/anparams/polind,betaexq,cutexq,soconst,
     & pa,px,p1,p2,q,polrg,polx,pqrg,pqx,theta6,c9anion,
     & c9neut,nrg

*
      write(*,*) 'prinaxes'
*
      do i = 1,nrg+1
        rx(i) = coordin(i*3-2)
        ry(i) = coordin(i*3-1)
        rz(i) = coordin(i*3)
      enddo
*
*  Find CM coordiantes, reset origin to CM
*
      cmx=0.0d0
      cmy=0.0d0
      cmz=0.0d0
      do i=1,nrg
        cmx = cmx+rgmass*rx(i)
        cmy = cmy+rgmass*ry(i)
        cmz = cmz+rgmass*rz(i)
      enddo
      tmass = nrg*rgmass + halmass
      cmx = (cmx + halmass*rx(nrg+1))/tmass
      cmy = (cmy + halmass*ry(nrg+1))/tmass
      cmz = (cmz + halmass*rz(nrg+1))/tmass

      do i=1,nrg+1              !* Set origin to CM
        rx(i) = rx(i) - cmx
        ry(i) = ry(i) - cmy
        rz(i) = rz(i) - cmz
      enddo
*
*  Set up inertia tensor
*
      do j=1,3
```

```
        do k=1,3
          ti(j,k) = 0.0d0
        enddo
      enddo

      do i=1,nrg                    !* Rare gases (take rgmass = 1)
        ti(x,x) = ti(x,x) + ry(i)**2 + rz(i)**2 !* Diagonal elements
        ti(y,y) = ti(y,y) + rx(i)**2 + rz(i)**2
        ti(z,z) = ti(z,z) + rx(i)**2 + ry(i)**2
        ti(y,x) = ti(y,x) - rx(i)*ry(i)         !* Below diagonals
        ti(z,x) = ti(z,x) - rx(i)*rz(i)
        ti(z,y) = ti(z,y) - ry(i)*rz(i)
      enddo

      i = nrg+1                             !* Halogen, mass weighted
      rat = halmass/rgmass
      ti(x,x) = ti(x,x) + rat*(ry(i)**2 + rz(i)**2)   !* Diagonal
elements
      ti(y,y) = ti(y,y) + rat*(rx(i)**2 + rz(i)**2)
      ti(z,z) = ti(z,z) + rat*(rx(i)**2 + ry(i)**2)
      ti(y,x) = ti(y,x) - rat*rx(i)*ry(i)       !* Below diagonals
      ti(z,x) = ti(z,x) - rat*rx(i)*rz(i)
      ti(z,y) = ti(z,y) - rat*ry(i)*rz(i)

      call rs(3,3,ti,prmoms,1,prcos,fv1,fv2,ierr)
*
*  Transform to principal axis coordinates, using direction cosine
*  matrix.
*
      do i=1,nrg+1
        xnew = prcos(x,x)*rx(i) + prcos(y,x)*ry(i) +
     &         prcos(z,x)*rz(i)
        ynew = prcos(x,y)*rx(i) + prcos(y,y)*ry(i) +
     &         prcos(z,y)*rz(i)
        znew = prcos(x,z)*rx(i) + prcos(y,z)*ry(i) +
     &         prcos(z,z)*rz(i)
        rx(i) = xnew
        ry(i) = ynew
        rz(i) = znew
      enddo

      write(*,*)
      write(*,*) 'coords wrt principal axes'
      do i=1,nrg+1
        write(*,*) rx(i),ry(i),rz(i)
      enddo
*
*  Reflect in each plane, so that sum of coordinates is positive
*
      xsum=0.0d0
      do i=1,nrg+1
        xsum = xsum + rx(i)
        ysum = ysum + ry(i)
        zsum = zsum + rz(i)
      enddo
      if (xsum.lt.0) then
        do i = 1,nrg+1
```

```
          rx(i) = -rx(i)
        enddo
      endif
      if (ysum.lt.0) then
        do i = 1,nrg+1
          ry(i) = -ry(i)
        enddo
      endif
      if (zsum.lt.0) then
        do i = 1,nrg+1
          rz(i) = -rz(i)
        enddo
      endif
*
*
*  Calculate r,theta and phi of rgs wrt halide
*
      j=nrg+1
      do i=1,nrg
        rh(i) = dsqrt((rx(i)-rx(j))**2+(ry(i)-ry(j))**2
     &                +(rz(i)-rz(j))**2)
        ph(i) = atan(dabs(ry(i)/rx(i)))
        if (((rx(i)-rx(j)).lt.0).and.((ry(i)-ry(j)).ge.0)) then
          ph(i) = pi - ph(i)
        elseif (((rx(i)-rx(j)).lt.0).and.((ry(i)-ry(j)).lt.0)) then
          ph(i) = pi + ph(i)
        elseif (((rx(i)-rx(j)).ge.0).and.((ry(i)-ry(j)).lt.0)) then
          ph(i) = 2*pi - ph(i)
          if (dabs(ph(i)-2*pi).lt.tiny) then          !* wraparound
            ph(i) = 0.
          endif
        endif
        if (dabs(ph(i)-2*pi).lt.tiny) then
          ph(i) = 0.
        endif
        th(i) = acos(rz(i)/rh(i))

      enddo
*
*  Sort rgs by theta and then phi
*
1000  nswitch = 0
      do i=1,nrg-1
        do j=i,nrg
          if (((th(i)-th(j)).gt.tiny).or.((dabs(th(i)-th(j)).le.
     &       tiny).and.((ph(i)-ph(j)).gt.tiny))) then
            t1 = rx(i)
            t2 = ry(i)
            t3 = rz(i)
            t4 = rh(i)
            t5 = th(i)
            t6 = ph(i)
            rx(i) = rx(j)
            ry(i) = ry(j)
            rz(i) = rz(j)
            rh(i) = rh(j)
            th(i) = th(j)
```

```
            ph(i) = ph(j)
            rx(j) = t1
            ry(j) = t2
            rz(j) = t3
            rh(j) = t4
            th(j) = t5
            ph(j) = t6
            nswitch = 1
          endif
        enddo
      enddo
      if (switch.ne.0) goto 1000
*
*
*  If two axes are degenerate, rotate around the nondegenerate axis so
that
*  the first rg atom not on the nondegenerate axis lies in the
direction of
*  one of the degenerate axes
*
*
      if (dabs(prmoms(x)-prmoms(y)).lt.tiny) then

        write(*,*) 'x and y are degenerate (oblate symmetric top)'

        j = 0
        do i=1,nrg
          if ((j.eq.0).and.((dabs(rx(i)).gt.tiny).or.
     &       (dabs(ry(i)).gt.tiny))) then
            j = i
          endif
        enddo

        angle = atan(dabs(ry(j)/rx(j)))
        if ((rx(j).lt.0).and.(ry(j).ge.0)) then
          angle = pi - angle
        elseif ((rx(j).lt.0).and.(ry(j).lt.0)) then
          angle = pi + angle
        elseif ((rx(j).ge.0).and.(ry(j).lt.0)) then
          angle = 2*pi - angle
        endif

        do i = 1,nrg+1
          xnew = cos(angle)*rx(i) + sin(angle)*ry(i)
          ynew = -sin(angle)*rx(i) + cos(angle)*ry(i)
          rx(i) = xnew
          ry(i) = ynew
        enddo

      elseif  (dabs(prmoms(z)-prmoms(y)).lt.tiny) then

        write(*,*) 'y and z are degenerate (prolate symmetric top)'

        j = 0
        do i=1,nrg
          if ((j.eq.0).and.((dabs(rz(i)).gt.tiny).or.
     &       (dabs(ry(i)).gt.tiny))) then
```

```
            j = i
          endif
        enddo

        angle = atan(dabs(ry(j)/rz(j)))
        if ((rz(j).lt.0).and.(ry(j).ge.0)) then
          angle = pi - angle
        elseif ((rz(j).lt.0).and.(ry(j).lt.0)) then
          angle = pi + angle
        elseif ((rz(j).ge.0).and.(ry(j).lt.0)) then
          angle = 2*pi - angle
        endif

        do i = 1,nrg+1
          znew = cos(angle)*rz(i) + sin(angle)*ry(i)
          ynew = -sin(angle)*rz(i) + cos(angle)*ry(i)
          rz(i) = znew
          ry(i) = ynew
        enddo

      endif
*
*  Calculate r,theta and phi of rgs wrt halide (AGAIN)
*
      j=nrg+1
      do i=1,nrg
        rh(i) = dsqrt((rx(i)-rx(j))**2+(ry(i)-ry(j))**2
     &                +(rz(i)-rz(j))**2)
        ph(i) = atan(dabs(ry(i)/rx(i)))
        if (((rx(i)-rx(j)).lt.0).and.((ry(i)-ry(j)).ge.0)) then
          ph(i) = pi - ph(i)
        elseif (((rx(i)-rx(j)).lt.0).and.((ry(i)-ry(j)).lt.0)) then
          ph(i) = pi + ph(i)
        elseif (((rx(i)-rx(j)).ge.0).and.((ry(i)-ry(j)).lt.0)) then
          ph(i) = 2*pi - ph(i)
          if (dabs(ph(i)-2*pi).lt.tiny) then          !* wraparound
            ph(i) = 0.
          endif
        endif
        if (dabs(ph(i)-2*pi).lt.tiny) then
          ph(i) = 0.
        endif
        th(i) = acos(rz(i)/rh(i))

      enddo
*
*  Sort rgs by theta and then phi (AGAIN)
*
1010  nswitch = 0
      do i=1,nrg-1
        do j=i,nrg
          if (((th(i)-th(j)).gt.tiny).or.((dabs(th(i)-th(j)).le.
     &       tiny).and.((ph(i)-ph(j)).gt.tiny))) then
            t1 = rx(i)
            t2 = ry(i)
            t3 = rz(i)
            t4 = rh(i)
```

```
            t5 = th(i)
            t6 = ph(i)
            rx(i) = rx(j)
            ry(i) = ry(j)
            rz(i) = rz(j)
            rh(i) = rh(j)
            th(i) = th(j)
            ph(i) = ph(j)
            rx(j) = t1
            ry(j) = t2
            rz(j) = t3
            rh(j) = t4
            th(j) = t5
            ph(j) = t6
            nswitch = 1
          endif
        enddo
      enddo
      if (switch.ne.0) goto 1010


      write(*,*)
      write(*,*) 'coords wrt principal axes x,y,z,r,theta,phi'
      do i=1,nrg+1
        write(*,150) rx(i),ry(i),rz(i),rh(i),th(i)*180./pi,
     &             ph(i)*180/pi
      enddo
150   format(6f10.4)

      do i = 1,nrg+1
        coordin(i*3-2) = rx(i)
        coordin(i*3-1) = ry(i)
        coordin(i*3) = rz(i)
      enddo

      return
      end

*=====================================================================
      subroutine calcfcfs(nmax,nanshow,nnshow,vtemp,origin,evs,wfs,
     &   evs2,wfs2,fcfs,totfcf,nsticks,nmode,sticks)
*=====================================================================
=
*
*  Input    nmax        maximum n in basis (# dvr points = nmax+1)
*           nanshow           number of anion eigenvalues to use
*                      (same for all modes)
*           nnshow            number of neutral eigenvalues
*           vtemp       vibrational temperature
*                      (in K, same for all modes)
*           origin            position of 0-0 line (in cm-1)
*           evs(m,i)    anion eigenvalues, m=mode
*                      (in eV, from well bottom)
*           wfs(m,i,j)  anion dvr wavefunctions of modes m in columns j
*           evs2(m,i)   neutral eigenvals of second state
*                      (eV, from well bottom)
*           wfs2(m,i,j) neutral wavefunctions of second state
```

```
*
*  Output    fcfs(m,i,j)        single mode fcfs:  m=mode,
*                                i=anion, j=neutral #quanta in mode
*            totfcf(m,i)        Total fcfs in mode m from anion v=i
*            nsticks                  number of vib sticks generated
*            nmode([1,2],m,i)   number of quanta in mode m of stick i
*                                1=anion, 2=neutral
*            sticks([1 or 2],i)       stick spectrum 1->position (cm-1)
*                                2->intensity (not normalized), i=index
*
      include 'param.file'
      parameter(ndm=150,nmmax=18,evtokelv=11604.45,cut=0.01)

      double precision wfs(nmmax,ndm+1,ndm+1),wfs2(nmmax,ndm+1,ndm+1),
     &  evs(nmmax,ndm+1),evs2(nmmax,ndm+1),fcfs(nmmax,ndm+1,ndm+1),
     &  totfcf(nmmax,ndm+1),sticks(2,1000)
      double precision pa(10),px(10),p1(10),p2(10),q(10)
      integer nmode(2,nmmax,1000)   !* 1=lower, 2=upper, nmmax=mode
index
*                               !* 1000 stick index
      integer ma(nmmax),mn(nmmax)
*                       !* anion, neutral mode indices (#quanta
*                       !* in mode)
      logical carry,switch

      common/anparams/polind,betaexq,cutexq,soconst,
     & pa,px,p1,p2,q,polrg,polx,pqrg,pqx,theta6,c9anion,
     & c9neut,nrg
*
      write(*,*) 'calcfcfs'
      orgev = origin/evtocm
*
*  Calculate single mode fcfs
*
      nmde = (nrg+1)*3-6
      do m = 1,nmde                 !* loop over modes
        do i = 1,nanshow              !* loop over anion state
          totfcf(m,i) = 0.0d0
          do j = 1,nnshow               !* loop over neutral state
            fct = 0.0d0
            do k = 1,nmax+1                 !* loop over dvr points
              fct = fct + wfs(m,k,i)*wfs2(m,k,j)
            enddo
            fcfs(m,i,j) = fct*fct
            totfcf(m,i) = totfcf(m,i)+fcfs(m,i,j)
          enddo
        enddo
      enddo
*
*  Calculate vibrational sticks, Including combination bands
*
      do m=1,nmde
        ma(m) = 0
        mn(m) = 0
      enddo
      nsticks = 0
c
```

```
c
Calc stick
      write(*,*) 'calculating vsticks'
100   if (.true.) then
        ncount = ncount+1
      endif

      aterm = 0.0d0      !* anion energy above v=0
      bterm = 0.0d0      !* neutral
      do m = 1,nmde
        aterm = aterm + evs(m,ma(m)+1) - evs(m,1)
        bterm = bterm + evs2(m,mn(m)+1) - evs2(m,1)
      enddo
      stick1 = orgev - aterm + bterm
      boltz = exp(-aterm*evtokelv/vtemp)
      stick2 = boltz
      do m=1,nmde
        stick2 = stick2 * fcfs(m,ma(m)+1,mn(m)+1)
      enddo
      if (stick2.ge.cut) then
        nsticks = nsticks + 1
        sticks(1,nsticks) = stick1*evtocm
        sticks(2,nsticks) = stick2
        do m = 1,nmde
          nmode(1,m,nsticks) = ma(m)
          nmode(2,m,nsticks) = mn(m)
        enddo
      endif

      carry = .true.                !* increment neutral mode indices
      do m=1,nmde
        if (carry) then
          if (mn(m).lt.nnshow-1) then
            mn(m) = mn(m) + 1
            carry = .false.
          else
            mn(m) = 0
          endif
        endif
      enddo
      if (carry) then          !* if neutral modes all zeroed, then inc
anion
        do m=1,nmde
          if (carry) then
            if (ma(m).lt.nanshow-1) then
              ma(m) = ma(m) + 1
              carry = .false.
            else
              ma(m) = 0
            endif
          endif
        enddo
      endif
      if (carry) then
        goto 200
      endif
      goto 100
```

```
200   write(*,*) ncount
*
*  Sort sticks
*
      write(*,*) 'sorting sticks'
500   switch = .false.
      do i=1,nsticks
        do j=i+1,nsticks
          if (sticks(1,i).gt.sticks(1,j)) then
            stemp1 = sticks(1,i)
            stemp2 = sticks(2,i)
            sticks(1,i) = sticks(1,j)
            sticks(2,i) = sticks(2,j)
            sticks(1,j) = stemp1
            sticks(2,j) = stemp2
            do m=1,nmde
              mtemp1 = nmode(1,m,i)
              mtemp2 = nmode(2,m,i)
              nmode(1,m,i) = nmode(1,m,j)
              nmode(2,m,i) = nmode(2,m,j)
              nmode(1,m,j) = mtemp1
              nmode(2,m,j) = mtemp2
            enddo
            switch = .true.
          endif
        enddo
      enddo
      if (switch) goto 500

      return
      end

*=====================================================================
      subroutine veqd(betain,theta6,qx,qy,qz,potout)
*=====================================================================
*
*  Exchange quadrupole-dipole & dispersion quadrupole
*  Ref: Ernesti & Hutson, Phys. Rev. A v.51,p.239
*
*  Input:   betain            Exchange quadrupole range parameter
(a0/10)^-1
*           theta6            quadrupole dispersion coefficient
(e*a0^8)
*           qx,qy,qz(ncl+1) Rg, halide coordinates (a0/10)
*
*  Output:  potout            potential (hartree/100)
*
      include 'param.file'
      double precision betain,theta6
      double precision qx(ncl+1),qy(ncl+1),qz(ncl+1),
     & dx(ncl+1),dy(ncl+1),dz(ncl+1),
     & rijx(ncl+1,ncl+1),rijy(ncl+1,ncl+1),rijz(ncl+1,ncl+1),
     & rij2(ncl+1,ncl+1),rij3(ncl+1,ncl+1),rij1(ncl+1,ncl+1)
*
*
      beta = betain*10.0d0
```

```
      nhal = ncluster+1
*
      do i = 1,ncluster       !* Relative Rg-Rg and Rg-X vectors
        do j = i+1,ncluster+1 !* (rij is vector from Rg j to Rg i)
          rijx(i,j) = (qx(i) - qx(j))/10.0d0    !* Convert to a0
          rijy(i,j) = (qy(i) - qy(j))/10.0d0
          rijz(i,j) = (qz(i) - qz(j))/10.0d0
          rijx(j,i) = -rijx(i,j)
          rijy(j,i) = -rijy(i,j)
          rijz(j,i) = -rijz(i,j)
          rij2(i,j) = rijx(i,j)**2+rijy(i,j)**2+rijz(i,j)**2
          rij1(i,j) = dsqrt(rij2(i,j))
          rij3(i,j) = rij1(i,j)**3
          rij1(j,i) = rij1(i,j)
          rij2(j,i) = rij2(i,j)
          rij3(j,i) = rij3(i,j)
        enddo
      enddo
*
* Zero out dipoles
*
      do i = 1,ncluster
        dx(i) = 0.0d0
        dy(i) = 0.0d0
        dz(i) = 0.0d0
      enddo
*
*  Add contribution to Rg dipoles from exchange quadrupole and
*  dispersion quadrupole between each pair of Rgs
*
      do i = 1,ncluster-1
        do j = i+1,ncluster
          ex = exp(-(beta**2)*rij2(i,j)/2.0d0)
          exquad = -rij2(i,j)*ex/(1.0d0-ex)/2
     &             +theta6/(rij3(i,j)**2)
          exdip = exquad/rij2(i,j)/2.0d0
          dx(i) = dx(i)+exdip*rijx(i,j)
          dy(i) = dy(i)+exdip*rijy(i,j)
          dz(i) = dz(i)+exdip*rijz(i,j)
          dx(j) = dx(j)+exdip*rijx(j,i)
          dy(j) = dy(j)+exdip*rijy(j,i)
          dz(j) = dz(j)+exdip*rijz(j,i)
        enddo
      enddo
*
*
*
*  Compute charge-dipole interactions
*
      vcd = 0.0d0
      do i = 1,ncluster
        dotprod = dx(i)*rijx(nhal,i)+dy(i)*rijy(nhal,i)
     &            +dz(i)*rijz(nhal,i)
        vcd = vcd - dotprod/rij3(nhal,i)
      enddo

      potout = vcd*1.0d2      !* Convert to hartree/100
```

```
      return

      end

*=====================================================================
      subroutine vinddq(polrg,polx,pqrg,pqx,qx,qy,qz,dp,dhalpair,
     &  drgpair,qhalpair,qrgpair,qp,vcd,vcq,potout)
*=====================================================================
```

See Section C5.7 (file "pforce.f") for a complete listing of this subroutine

```
*=====================================================================
      subroutine exg(betain,th6,qx,qy,qz,vexx,vddisp,vqdisp,potout)
*=====================================================================
```

See Section C5.7 (file "pforce.f") for a complete listing of this subroutine

```
*=====================================================================
      double precision FUNCTION ERF(xin)
*=====================================================================
```

See Section C5.7 (file "pforce.f") for a complete listing of this subroutine

```
*=====================================================================
      subroutine vat(c9,qx,qy,qz,potout)
*=====================================================================
```

See Section C5.7 (file "pforce.f") for a complete listing of this subroutine

C7. References for Appendix C

[1] F. H. Stillinger and D. K. Stillinger, J. Chem. Phys. **93**, 6106 (1990).

[2] F. H. Stillinger and T. A. Weber, Phys. Rev. A **28**, 2408 (1983).

[3] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes*, 2nd ed. (Cambridge University Press, Cambridge, 1992).

[4] P. R. Bevington, *Data Reduction and Error Analysis for the Physical Sciences* (McGraw-Hill, New York, 1969).

[5] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, Science **220**, 671 (1983).

[6] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, J. Chem. Phys. **21**, 1087 (1953).

[7] D. Vanderbilt and S. G. Louie, J. Comput. Phys. **56**, 259 (1984).

[8] I. M. Navon, F. B. Brown, and D. H. Robertson, Computers Chem. **14**, 305 (1990).

[9] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids* (Clarendon Press, Oxford, 1987).

[10] Z. Li, A. Borrmann, and C. C. Martens, J. Chem. Phys. **97** (1992).

[11] R. A. Aziz and M. J. Slaman, Mol. Phys. **58**, 679 (1986).

[12] E. L. Pollock and B. J. Alder, Phys. Rev. Lett. **41**, 903 (1978).

[13] P. Jedlovszky and G. Palinkas, Mol. Phys. **84**, 217 (1995).

[14] R. L. Asher, D. A. Micha, and P. J. Brucat, J. Chem. Phys. **96**, 9683 (1992).

[15] A. D. Buckingham, Adv. Chem. Phys. **12**, 107 (1967).

[16] A. Ernesti and J. M. Hutson, Phys. Rev. A **51**, 239 (1995).

[17] The EISPACK source code may be obtained by email from netlib@research.att.com or by FTP from the server "research.att.com" using the login name "netlib."

[18] R. B. Metz, Ph.D., University of California, 1991.