KINETICS OF BIMOLECULAR EXCHANGE REACTIONS: A COMPUTATIONAL APPROACH

By

Chris Carruthers

A thesis submitted to
the School of Graduate Studies and Research
in partial fulfillment of the requirements
for the degree of Master of Science

Department of Chemistry

University of Ottawa
Ottawa, Ontario

ISBN 0-315-85809-5

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

Abstract

In order to better understand the kinetics of gas phase bimolecular exchange reactions a computer program has been written which allows easy exploration of the time dependent and vibrational level dependent details of this class of reactions. BIMSIM (for BIMolecular exchange reaction SIMulation) is intended as a very flexible "virtual laboratory" which can easily be configured and reconfigured for a wide range of different experiments (e.g., laser pulse or shock tube), different initial conditions (e.g., of temperature, reactant concentration, and molecular environment), for different reactions in this class, and for different levels of approximation. In order to test the validity and demonstrate the use of the program a reaction system was found for which appropriate input data is available and for which suitably detailed analytical calculations have been done. Agreement was found to be excellent. Using BIMSIM, results of chemical interest were obtained for the reaction $Br+HCl \rightarrow HBr+Cl$. It was found that non-equilibrium depletion of the vibrational levels of HCl are as much as a factor of 10 and that they depend on the relative amounts of Br, HCl and inert diluent He, as well as on the temperature, becoming more pronounced around 500 K. Interesting details of the time dependence of the fractional level populations are discussed.

Acknowledgement

I would like to thank my supervisor Prof. Heshel Teitelbaum, especially for his inexhaustible patience. I would also like to thank my wife Karen McCarthy and my good friend and colleague Dr. Victoria Barclay. I gratefully acknowledge an Ontario Graduate Scholarship from the Government of Ontario. This research was also supported by the Natural Sciences and Engineering Research Council of Canada. Finally, the generous computing resources made available by the University of Ottawa which allowed the completion of this work are acknowledged.

Contents

List of Tables

List of Figures

Chapter 1. Introduction

1.1 Scope

The gas-phase exchange reaction is one of the most common, important, relatively simple, and widely studied chemical reactions (see, for example, ref. [1]). In order to better understand the kinetics of such reactions of the form

$$A + BC \rightarrow AB + C, \tag{1.1}$$

a computer program has been written by the author which allows easy exploration of the time dependent details of this class of reactions. BIMSIM (for BIMolecular exchange reaction SIMulation) is intended as a very flexible "virtual laboratory" which can easily be configured and reconfigured for a wide range of different experiments (e.g., laser pulse or shock tube), different initial conditions (e.g., of temperature, reactant concentration, and molecular environment), for different reactions in this class, and for different levels of approximation. See Appendix A for a detailed description of BIMSIM and Appendix C for the source code.

The need to study the kinetics of bimolecular reactions with their vibrational detail stems from two important observations: a) endothermic bimolecular reaction rates are enhanced by vibrational energy of the reagents; b) exothermic bimolecular reactions often produce species which are vibrationally and rotationally excited[2]. Until now kineticists have simulated real reaction mechanisms (consisting of several bimolecular reactions) by implictly assuming that the vibrational and rotational levels of all multiatomic species are distributed according to Boltzmann statistics. This is an approximation which is not justified particularly for fast chemical reactions or for slowly relaxing species. Thus, it is generally true that the population of the high vibrational levels of reagents are depleted, whereas the high vibrational levels of products are overpopulated. Since the products of one reaction are sometimes

the reagents for subsequent reactions, the whole process is rather complicated. Depending on the reaction conditions different scenarios can be envisaged. The 'rate coefficient' becomes distorted from the kineticist's impression of it. Indeed the whole concept of a rate constant comes into question. In order to see how serious such effects can become, it is important to investigate the kinetics at its microscopic detail.

The approach, admitting to non-Boltzmann distributions, can be termed, by definition, "non-equilibrium kinetics". Although this term encompasses all kinds of energy levels, including electronic and translational, BIMSIM is restricted to the cases of chemical interest and does not consider plasma temperatures where electronic and translational effects become important[3]. What is known currently about vibrational/rotational non-equilibrium kinetics is restricted in the literature to the field of unimolecular reactions. BIMSIM generalizes the techniques to include bimolecular reactions.

1.2 Non-equilibrium Kinetics of Unimolecular Reactions

Figure 1.1 is a schematic diagram showing the energy levels of molecule BC, and the barrier to rupture the weakest bond, E_0. For most molecules of chemical interest $E_0 \gg k_B T$ (where k_B is the Boltzmann constant and T is the absolute temperature). Because of the high barrier, reaction occurs by stepwise climbing of the vibrational ladder. It is only near the barrier that excited molecules actually can dissociate collisionally and only above the barrier that excited molecules fall apart spontaneously. The whole process is a competition between vibrational excitation and reaction. At the low pressure limit of a unimolecular reaction, the processes which determine the rate of reaction are energy transfer steps. At this limit, energy transfer (vibrational excitation or replenishment of the population of those levels near the dissociation limit depleted by reaction) is inefficient. The balance is struck leaving a steady non-Boltzmann distribution. The reaction can be modeled by solving the very large

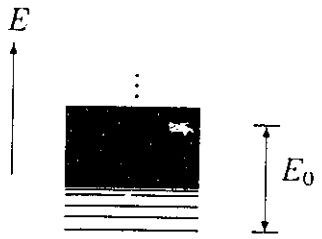

Figure 1.1: Schematic Energy Levels of a Molecule

set of differential rate equations, commonly called the master equation. For $E < E_0$

$$\frac{d[\mathrm{BC}(E,t)]}{dt} = \sum_{E'} k(E' \rightarrow E)[\mathrm{M}][\mathrm{BC}(E',t)] - \sum_{E'} k(E \rightarrow E')[\mathrm{M}][\mathrm{BC}(E,t)]. \quad (1.2)$$

For $E \geq E_0$

$$\begin{aligned} \frac{d[\mathrm{BC}(E,t)]}{dt} &= \sum_{E'} k(E' \rightarrow E)[\mathrm{M}][\mathrm{BC}(E',t)] \\ &\quad - \sum_{E'} k(E \rightarrow E')[\mathrm{M}][\mathrm{BC}(E,t)] - k(E)[\mathrm{BC}(E,t)]. \end{aligned} \quad (1.3)$$

Here $[\mathrm{BC}(E,t)]$ is the concentration of the molecule with internal energy E at time t, [M] is the concentration of an inert bath gas, $k(E)$ is the specific rate constant for spontaneous dissociation of the molecule, $k(E \rightarrow E')$ is the rate constant for collisional energy transfer from energy level E to energy level E', and $k(E' \rightarrow E)$ is that for the reverse process. $k(E \rightarrow E')$ and $k(E' \rightarrow E)$ are related by microscopic reversibility. This relationship guarantees that, in the absence of reaction, a Boltzmann distribution results at $t = \infty$.

Equation 1.3 can be solved analytically[4] under the assumptions that BC is big, energy transfer is poor, $k(E' \rightarrow E)$ decreases exponentially with energy gap, BC is highly dilute in the bath gas M, BC is translationally equilibrated, M is thermally equilibrated, and the reverse recombination of B and C is unimportant. Because of the large number of levels involved one often replaces the summation by an integration. For the special case of the steady state limit (when the distribution

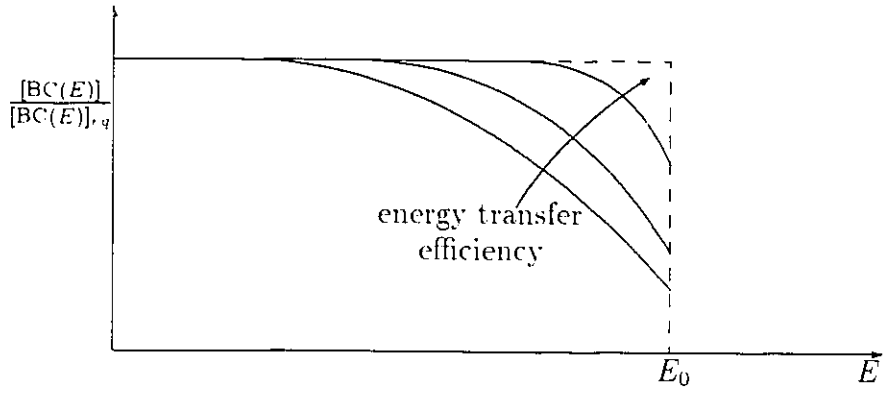

Figure 1.2: The Depletion of BC at high E

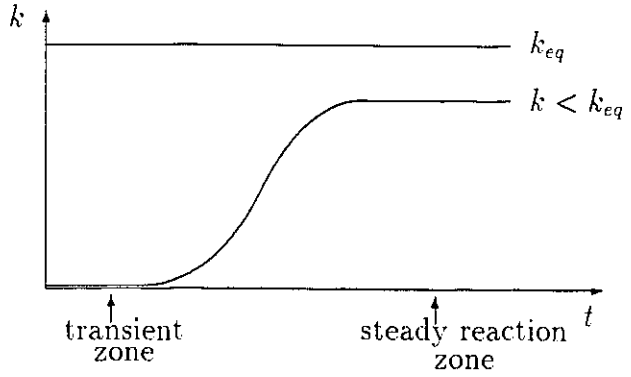

Figure 1.3: k vs. t

becomes invariant in time) the following non-Boltzmann distribution

$$\frac{\mathrm{BC}(E)]}{\mathrm{BC}(E)]_{eq}} \approx 1 - \frac{k_B T}{\langle \Delta E \rangle + k_B T} \exp[-\frac{(E_0 - E)}{k_B T}] \tag{1.4}$$

and the non-equilibrium rate coefficient k (compared to k_{eq}, the one where a Boltzmann distribution prevails):

$$\frac{k}{k_{eq}} \approx \frac{\langle \Delta E \rangle}{k_B T} < 1 \tag{1.5}$$

where $\langle \Delta E \rangle$ is the average energy transferred per collision ($\ll k_B T$). These results are shown schematically in Figures 1.2 and 1.3. Figure 1.3 additionally shows the time-evolution of the non-equilibrium rate-coefficient, k. This is a phenomenological quantity defined as $k = -Rate/([\mathrm{BC}][\mathrm{M}])$. Equation 1.5 shows that k is depressed

from k_{eq}. It turns out that this depression becomes more and more prominent as the temperature increases, resulting in convex Arrhenius plots. It is only after the transient process is finished that k becomes essentially constant.

As important as unimolecular reactions are, they form, after all, only a small fraction of chemical reactions. For every molecule that self-destructs there is an almost countless number of possible chemical partners with which it can react bimolecularly. So what is known about non-equilibrium kinetics is a small fraction of what should be known. The same guiding principles successfully used for unimolecular reactions (competition between energy transfer and reactivity) apply in general to all reactions, and suggest that manifestations of this should be looked for in the case of bimolecular reactions. It is the programme of our research group to start to fill in this gap in knowledge.

1.3 Comparison Between Unimolecular and Bimolecular Reactions of BC

The following table illustrates the expected similarities and differences between uni- and bimolecular reactions.

	Unimolecular Reaction $BC + M \rightarrow B + C + M$	Bimolecular Reaction $A + BC \rightarrow AB + C$
Competition between energy transfer and reactivity	Yes	Yes
Distortion of population distribution	Yes	Yes
k	$< k_{eq}$	$< k_{eq}$
[collider]	[M] = constant	[A](t)
E_0	$\gg k_B T$	no restriction
Important levels which react	only those close to the dissociation limit	any level can react with A
Scientific activity	well-studied	hardly studied at all

1.4 Non-Equilibrium Kinetics of Bimolecular Reactions

1.4.1 What are the Failings of Classical Chemical Kinetics?

Classical chemical kinetics is based upon two or three fundamental equations which are used (almost) universally without question. For the reaction $A + BC \rightarrow AB + C$

$$Rate = k[A][BC]. \tag{1.6}$$

If the reverse reaction is important

$$Rate = k_f[A][BC] - k_r[AB][C]. \tag{1.7}$$

Here k_f and k_r are the rate coefficients for the forward and reverse processes respectively. The kinetic mass-action law relates k_f to k_r by:

$$\frac{k_f}{k_r} = K_c \tag{1.8}$$

where K_c is the equilibrium constant.

These fundamental equations are predicated upon the implicit assumption that the internal energy states of BC are in a Boltzmann distribution during the reaction. From the above considerations this is not strictly correct. $k < k_{eq}$, and this distortion is dependent upon the chemical environment which determines how efficient vibrational (say) excitation is. So, k is really dependent upon the concentrations of all species present, some of which may be varying with time. In other words, $k = k([A](t), [BC](t), \ldots)$ and is not a true constant, preventing direct integration of (1.6) for use in analysing experimental data. Even if the form of equation 1.6 were correct, it is not obvious that under conditions of non-equilibrium distributions, forward and reverse rates are additive. Furthermore, the distortion to k_f may not be of the same magnitude as to k_r, so that (1.8) is not strictly correct either.

1.4.2 Manifestations of the Failings of Classical Kinetics

It has already been mentioned above that k differs from k_{eq} by varying amounts depending on the temperature. This could result in non-Arrhenius behaviour. Theories of chemical reaction rely on the equilibrium assumption and fine-tune themselves by trying to reproduce experimental results. Curvature of such plots might be falsely attributed to special features of the transition state or to quantum mechanical tunneling rather than to simple non-equilibrium effects. Consequently theoretical approaches might be led astray.

Consequences for experimental kinetics are no less severe. It may legitimately be asked if the rate coefficients in the literature, for the elementary reactions, are correct. Are they k_{eq}, or somewhat less than k_{eq}? Were they determined under conditions where non-equilibrium effects were absent? May they be used to predict the behaviour of complex systems such as combustion or atmospheric processes? In other words, even if k_{eq} for an elementary process were properly extracted, is it appropriate to use it for simulating a complex process in which non-equilibrium effects are likely to show up? Certainly the chemical environment of such systems is highly varied. Furthermore the products of elementary reactions may be produced in strongly non-Boltzmann distributions, especially if exothermic. Consequently, secondary reactions may not even proceed initially from a Boltzmann distribution. Each subsequent step's effect is compounded.

This thesis is designed to produce a program which allows our research group to explore conditions and give guidelines to chemists for deciding when non-equilibrium effects are important in bimolecular reactions.

1.4.3 Implications for Kinetic Modeling

If the vibrational/rotational substructure is not ignored, then a simple mechanism is transformed into a nightmare. It becomes necessary to measure or calculate many state-selected rate constants instead of a single thermal rate coefficent for each elementary reaction. It becomes necessary to solve a far more complicated set of simultaneous differential rate equations than before. Rather complicated (less

transparent) dependencies of species concentration on chemical environment will result.

1.4.4 Example

The first explosion limit for the H_2/O_2 reaction is considered to consist of the following steps (here only the vibrational structure is explicitly mentioned):

$$\begin{aligned} \textit{Reactants} &\rightarrow \mathrm{OH}(v) \\ \mathrm{OH}(v) + \mathrm{H_2}(v') &\rightarrow \mathrm{H_2O} + \mathrm{H} \\ \mathrm{H} + \mathrm{O_2}(v'') &\rightarrow \mathrm{OH}(v) + \mathrm{O} \\ \mathrm{O} + \mathrm{H_2}(v') &\rightarrow \mathrm{OH}(v) + \mathrm{H} \\ \mathrm{H} &\rightarrow \textit{wall}. \end{aligned}$$

Even if only 3 levels for each of H_2, O_2 and OH are considered as playing a role then there are 31 different rate constants to determine, and 12 differential equations to solve, some of which have a large number of terms (e.g. 28 for H). This is indeed a more difficult problem to solve than the classical problem.

At this point a choice can be made regarding tactics. One can try to revert to the 5-equation approach, accounting for non-equilibrium effects by using an improved analytical rate law for each elementary step as does Teitelbaum[5]; or else one can rigourously solve the 12 simultaneous differential equations. This study takes the rigorous approach. It makes one less assumption than the analytical approach does—namely it does not assume the existence of a steady state. It follows the entire time history of the evolution of population distributions, and so it can be used more generally. In this study it is specifically used to compare with the analytical expression of ref. [5], and then to learn some details of the reaction $\mathrm{Br} + \mathrm{HCl} \rightarrow \mathrm{HBr} + \mathrm{Cl}$.

1.5 Previous Numerical Studies

There has been very little work done which solves the master equation for this class of reactions numerically.

The first study of this type was done by Porter *et al.*[6] who considered the reaction

$$Cl + HCl \rightarrow Cl_2 + H \tag{1.9}$$

at 400 K. They solved a vibrational level master equation numerically, and used quasiclassical trajectory calculations to obtain the rates for the reactive process. They found that reaction proceeds predominately from $v = 5$ initially but that the population in this level is rapidly depleted. Thus actually about 56% of reaction proceeds from $v = 4$ but the authors did not state what effect this depletion has on the phenomenological rate constant, k.

Later Finkelman *et al.*[7, 8] addressed the validity of the phenomenological rate law for the general reaction

$$X + YZ \rightarrow XY + Z. \tag{1.10}$$

A vibrational ladder model was also employed in this study (in which reactive and non-reactive processes were considered). The system was composed of structureless atoms (X), diatomic molecules (YZ), products (XY, Z), and an inert gas (M). The reactant diatomic was treated as a truncated harmonic oscillator with 4 or 7 vibrational levels. Landau-Teller scaling was used to obtain the energy transfer rate constants which were also assumed to be proportional to [M]. The master equation for the system was integrated using a numerical integrator based on Gear's method[9] (as does the present study). They found that after a transient period the system obeys a second-order phenomenological rate law and that for some conditions the phenomenological rate coefficient averaged over the course of reaction represented the observed rate constant, whereas for extreme conditions (for example, high temperature) it did not. In the most extreme case they found that the final value of the phenomenological rate coefficient was almost 1000 times smaller than the maximum value at $t = 0$. They also found that the phenomenological rate coefficient depended on the initial conditions of the experiment simulated.

At about the same time, Gutkowicz-Krusin[10] attempted to understand how the phenomenological rate equation arises from the underlying microscopic description by studying a simple 3-component multistate model for the general reaction 1.10. In

this model the reactant has only two states and the product has one. Only collisions of the type X–Y[7] and XY–Z were considered. Gutkowicz-Krusin found that after a transient period the phenomenological rate equation always holds but that the transient period can be a macroscopically long time and that the duration of the transient period, as well as the magnitude of the rate constant, depended on the conserved quantities, such as the sum [X] + [XY].

More recently, Lim and Truhlar[11, 12] reported a more realistic calculation for the system

$$\mathrm{Cl} + \mathrm{HBr} \rightarrow \mathrm{HCl} + \mathrm{Br} \tag{1.11}$$

at 300 Kin an excess of Ar. Their differential rate equation accounted for state-to-state reactive processes and for simultaneous intramolecular vibration-rotation energy transfer as well as intermolecular vibration-vibration energy transfer processes and analogous processes for the reverse reaction. The phenomenological rate coefficient for this system was calculated by numerical integration of the rate equations using a method derived from the work of Gear[9], as well as by iteration to a quasisteady state and by eigenvalue analysis of a linearized master equation. They found multiple quasisteady states and, therefore, that the phenomenological rate law did not hold with a single pair of rate constants (one for forward reaction and one for reverse) throughout the reaction. They claim that the rate coefficient was distorted by no more than 1%, though, under typical experimental conditions. However, with the myriad of input rate constants involved, it is extremely difficult to "see the forest for the trees". Moreover, they chose a temperature of 300 Kand they concentrated on the exothermic direction. Both of these choices were self-defeating and led to negligible non-equilibrium effects. Notwithstanding these shortcomings this reaction system is a fascinating one because of the potentially large effects. Further study is warranted if one can concentrate on the endothermic direction, $\mathrm{Br} + \mathrm{HCl} \rightarrow \mathrm{HBr} + \mathrm{Cl}$, and on higher temperatures, as well as on the detailed time history of the individual vibrational levels of HCl. Spectacular effects can be expected.

Teitelbaum[13] solved the master equation for bimolecular exchange reactions

by treating the molecule as a harmonic oscillator truncated at the level from which reaction is primarily occurring. He obtained closed-form expressions for calculating the non-equilibrium correction factor to the rate coefficient over a wide range of temperatures and reagent concentrations—but only if the reaction can be isolated from secondary or back-reactions. Later work[5, 14] allowed reaction to occur from any level.

Other theoretical but less closely related work using techniques such as eigensystem analysis has been done by Present and Morris[15], Widom[16, 17], Snider and Ross[18], Shizgal[19] and Poulsen[20].

Chapter 2. Methodology

2.1 The Master Equation Approach

The master equation (1.3) describes the rate of change of concentration of molecule BC in a particular internal energy state when BC undergoes unimolecular reaction and energy transfer. The same formalism can be used for bimolecular reactions if $k(E)[\mathrm{BC}(E,t)]$ is replaced by $k_v[\mathrm{A}][\mathrm{BC}(v,t)]$. This study adopts the philosophy of addressing the basic physics of the problem and leaves refinements to future work. BIMSIM can be easily extended to solve more general problems but is at the moment restricted to accomodate only those processes for which rate constants can realisticly be expected to be found in the literature. Thus, for now, only single quantum energy transfer processes are considered. For purely vibrational processes this means that reaction 1.1 can be decomposed into the following elementary processes:

$$\mathrm{A} + \mathrm{BC}(v) \rightarrow \mathrm{AB} + \mathrm{C} \qquad v = 0, 1, 2 \ldots n \tag{2.1}$$

$$\mathrm{R} + \mathrm{BC}(v) \rightarrow \mathrm{R} + \mathrm{BC}(v-1) \qquad v = 1, 2, 3 \ldots n \tag{2.2}$$

$$\mathrm{R} + \mathrm{BC}(v) \rightarrow \mathrm{R} + \mathrm{BC}(v+1) \qquad v = 0, 1, 2 \ldots n-1 \tag{2.3}$$

where $\mathrm{BC}(v)$ is diatomic species BC in vibrational level v, A is an atom or radical and R represents all relaxer species present (possibly including A, BC, inert diluent M, etc.). Then the differential equation describing the rate of change of the population, N_v, of BC in level v is given by

$$\begin{aligned} -\frac{dN_v}{dt} &= k_{v,v-1} N_v N_{\mathrm{R}} + k_{v,v+1} N_v N_{\mathrm{R}} \\ &\quad - k_{v-1,v} N_{v-1} N_{\mathrm{R}} - k_{v+1,v} N_{v+1} N_{\mathrm{R}} + k_v N_{\mathrm{A}} N_v. \end{aligned} \tag{2.4}$$

In equation 2.4 each term containing N_{R} is actually a sum of terms such that

$$k_{i,j} N_R \equiv k_{i,j}^{\mathrm{A}} N_{\mathrm{A}} + k_{i,j}^{\mathrm{BC}} N_{\mathrm{BC}} + k_{i,j}^{\mathrm{M}} N_{\mathrm{M}} + \cdots \tag{2.5}$$

(that is, one term per relaxer species present); the rate constants for processes 2.1, 2.2 and 2.3 are k_v, $k_{v,v-1}$ and $k_{v,v+1}$, respectively; n is the uppermost vibrational level for which data are available; and terms of the form N_X represent the concentration of species X. The set of such equations for $v = 0, 1, 2 \ldots n$ is called the master equation for the reaction. This master equation can be solved numerically (that is, N_v for $v = 0, 1, 2 \ldots n$ as a function of t can be generated). An application of this approach is found in Chapter 3.

Rather than allowing only VT (that is, vibrational–translational) relaxation processes (equations 2.2 and 2.3) one can also allow VRT processes (in this context, R for rotational) such as

$$\mathrm{R} + \mathrm{BC}(v, J) \rightarrow \mathrm{R} + \mathrm{BC}(v + 1, J \pm 2), \tag{2.6}$$

or

$$\mathrm{R} + \mathrm{BC}(v, J) \rightarrow \mathrm{R} + \mathrm{BC}(v + 1, J \pm 1), \tag{2.7}$$

for homonuclear and heteronuclear BC respectively, and VV processes such as

$$2\mathrm{BC}(v) \rightarrow \mathrm{BC}(v + 1) + \mathrm{BC}(v - 1), \tag{2.8}$$

as well as all of the analogous processes for the reverse reaction. As more and more terms are introduced into the master equation to account for these other processes, the master equation becomes less and less of an approximation to reality. Currently, BIMSIM allows for processes 2.1, 2.2, 2.3, 2.6 but not 2.7 nor for VV processes and not for any of the processes for the reverse of reaction 2.1. On the surface, this seems to restrict BIMSIM to reactions with a slow reverse process, but in actual fact it is not so restrictive. It really means that the reverse reaction is slow enough not to interfere with the establishment of a steady forward process. In the example described in this study, Br + HCl, the reverse process is very fast, but like most other bimolecular reactions, it does not interfere with the details of the forward reaction[13]. Notwithstanding this, in keeping with the policy stated above, it is important to study the forward reaction in isolation in order to establish a baseline for comparision with the more complex case.

2.2 Gear's Method

Some systems of ordinary differential equations (o.d.e.'s) are called "stiff". This term has a formal mathematical definition (see, for example [21]), but of more practical concern is that o.d.e.'s (often describing physical systems) that locally behave exponentially and that have components decaying at very different rates are known to be stiff. The master equation approach described in the previous section is such a system. Conventional numerical methods for solving o.d.e.'s are very inefficient and have a tendency to be unstable when used to solve stiff o.d.e.'s. The method chosen by the author which is a commercial implementation of the work of Byrne and Hindmarsh[22] (which in turn is based on the work of Gear[9, 23]) was developed to handle stiff o.d.e.'s. Gear's method is an extremely efficient variation of the Adams predictor-corrector formulas. The algorithm adjusts the order and step size to allow the integration to proceed as quickly as possible while still keeping the estimated error within user-requested limits. It represents the state of the art in the numerical solution of systems of stiff o.d.e.'s[24].

2.3 Testing of the Numerical Method

BIMSIM was tested for accuracy in several ways. The initial step size and global error could be controlled by the user. In this study both the initial step size and tolerance parameter space were explored to find sufficiently stringent (small) values such that the end result was invariant. Values of these parameters were found which allowed the end result to be invariant to better than 1 part in 10^6.

Additionally one other numerical test was attempted. It was possible to check for the *chemical* accuracy by comparing the results of $\sum_v N_v$ (i.e., the sum of the integral of equation 2.4) with the integral of

$$\frac{dN_A}{dt} = \sum_v k_v N_A N_v. \tag{2.9}$$

The two results were identical within 1 part in 10^{10} by the end of the integration, indicating that round-off errors did not accumulate to serious levels, and that the

reagents were being converted into products as chemically required.

Chapter 3. Results and Discussion

3.1 The Br + HCl Reaction

In order to test the validity and demonstrate the use of the program a reaction system must be found for which appropriate input data is available and for which suitably detailed calculations have been done. There are several such systems available[5, 14]. The system

$$\mathrm{Br} + \mathrm{HCl} \rightarrow \mathrm{HBr} + \mathrm{Cl} \tag{3.1}$$

was studied by Teitelbaum *et al.*[5] who recently solved the steady-state master equation describing bimolecular exchange reactions and used the analytical expression thus obtained for the phenomenological rate coefficient to calculate it under various conditions. This system is also used in the present study for the purpose of testing BIMSIM against the analytical result and to extend Lim and Truhlar's study to the more interesting endothermic direction, and to more interesting temperatures. BIMSIM is sufficiently flexible that it can calculate [HCl] as a function of time using the same approximations and initial conditions as those of the above study. This is done in order to a) see if the same results expected from the steady-state calculations are obtained, at sufficiently long times, and to b) see if the steady-state approximation of ref. [5] is valid and if so under what conditions. Other approximations are as follows: HCl is treated as a Simple Harmonic Oscillator truncated at $v = 4$; only VRT energy transfer processes are considered; Landau-Teller scaling of energy transfer rate constants is employed and the species are in thermal equilibrium initially. Similarly, the initial conditions are: a range of five temperatures from 300–3000 K; a choice of species, R, responsible for vibrational relaxation (either He or HCl); and a choice of three mole fractions (that is [Br]/[R]) from 0.01–1.0. The analytical solution[5] further assumed that either He or HCl was the collider

but not both. In the case of the numerical solution, a finite amount of HCl must be present even if R = He, but the amount of HCl present must be such that it does not contribute significantly to relaxation in order that the present study can be validly compared to the analytical study. Therefore, whatever value is chosen for [HCl], the ratio [HCl]/[He] must be sufficiently small—at least as small as k_{10}^{He}/k_{10}^{HCl} or 2.8×10^{-3}. Through a process of trial and error it was determined that a value of [HCl]/[He] = 10^{-6} is sufficiently small.

Values for reaction rate constants and relaxation rate constants used as input to the program are given in Tables 3.1 and 3.2. These input parameters are the same as those used to obtain k analytically[5, 25].

Table 3.1: Relaxation Data[5]

T/K	k_{10}^{HCl}/dm^{3}mol^{-1}s^{-1}	k_{10}^{He}/dm^{3}mol^{-1}s^{-1}
300	1.44×10^{7}	4.10×10^{4}
500	2.04×10^{7}	5.31×10^{5}
1000	1.05×10^{8}	1.75×10^{7}
2000	8.91×10^{8}	2.66×10^{8}
3000	3.0×10^{9}	9.52×10^{8}

Table 3.2: Microscopic Reaction Rate Constants at all Temperatures

v	k_v/dm^{3}mol^{-1}s^{-1}
0	1×10^{-2}
1	2.0×10^{4}
2	1.0×10^{9}
3	7.0×10^{9}
4	1×10^{10}

3.2 Data Reduction and Results

The phenomenological rate law for reaction 3.1 is

$$rate = k[\mathrm{Br}][\mathrm{HCl}] \tag{3.2}$$

where k is the phenomenological rate coefficient for the reaction. Therefore, at any given time, $k = rate/([\mathrm{Br}][\mathrm{HCl}])$ and since the rate is defined as $-d[\mathrm{HCl}]/dt$,

$$k = -d[\mathrm{HCl}]/dt/([\mathrm{Br}][\mathrm{HCl}]). \tag{3.3}$$

Although not obvious, it can be shown[5] that k is made up of contributions from each vibrational level of HCl. Thus

$$k = \sum_{v=0}^{n} y_v k_v. \tag{3.4}$$

Here, $n = 4$, y_v is the fractional population in level v, and k_v is the level-selected rate constant for reaction 2.1 from level v. This is a quantity that can be readily calculated by BIMSIM. It turns out that y_v and hence k are dependent on time and initial conditions. To compare results with those of the analytical results[5] a single value of k must be chosen for each set of initial conditions. Figures 3.1 and 3.2 are plots of k normalized by its initial value $k(0)$ for the six sets of initial conditions at $T = 300\,\mathrm{K}$. Note that the abscissa is actually $\log \xi$ where ξ is the extent of reaction. Extent of reaction is here defined as $([\mathrm{Br}]_0 - [\mathrm{Br}])/[\mathrm{Br}]_0$ if Br is the limiting reagent, or $([\mathrm{HCl}]_0 - [\mathrm{HCl}])/[\mathrm{HCl}]_0$ if HCl is the limiting reagent, with $[\mathrm{Br}]_0$ and $[\mathrm{HCl}]_0$ being the initial reagent concentrations. This choice of abscissa allows for all the significant events that occur over the course of the reaction to be displayed on a single graph and also makes it easy to compare the results obtained from one set of initial conditions with those obtained from another. As can be seen from Figures 3.1 and 3.2, $k/k(0)$ is a relatively simple function; it starts out at unity and decreases monotonically to a smaller constant value; at later time the value of $k/k(0)$ stays the same (Figure 3.1) or increases to its equilibrium value $k_{eq}/k(0)$ (Figure 3.2), depending on the initial conditions. (k_{eq} corresponds to a Boltzmann

distribution. So $k_{eq} = k(0)$.) The increase back to the equilibrium value is discussed below. Teitelbaum's steady-state distribution of y_v is defined when $\dot{y}_v \approx 0$. Because of the linear relation between y_v and k (equation 3.4) steady state corresponds to times after which $\dot{k} = 0$. Hence, the obvious value to choose for comparison is the minimum value of $k/k(0)$ obtained for each set of initial conditions. The results are presented in Table 3.3. The graphs of $k/k(0)$ vs. $\log \xi$ for all initial conditions are given in Appendix B.

3.3 Comparison with Other Work

Since the analytical results of Teitelbaum *et al.* are mathematically exact, at least within the framework of the approximations given above, and since the same approximations were used in the present study it is expected at first glance that values obtained for k in the two studies should be identical. This is, in fact, not the case; while the agreement obtained might be considered excellent (within 1 part in 10^5 in most cases) it is not exact. There are two reasons why this is not surprising. (1) The analytical results are based on the steady-state assumption, which, as the results in Appendix B show, is not valid at all times. A steady reaction begins only after a finite time. (2) Furthermore, the analytical results shown in Table 3.3 were obtained using initial reagent concentrations and do not account for the disappearance of Br (or HCl). Consequently there is only a limited time interval where the analytical results are applicable. This time interval can be so short (see especially Figure B.19) that the value of $k/k(0)$ can be distorted, in principal, from the steady-state value. Alternatively it is possible that under some conditions the reaction is so fast that a steady distribution is not established in time, i.e., the reaction could proceed during the transient phase. Therefore we are not searching for exact agreement but rather for reasonable agreement. The results agreed for the most part to better than 1 part in 10^5 and where they did not the analytical results were always smaller than the numerical results. Furthermore, the cases where they did not agree corresponded to the cases (such as the one in Figure B.19) where the $k/k(0)$ curve decreased, and, rather than forming a plateau, reached a minimum and immediately increased

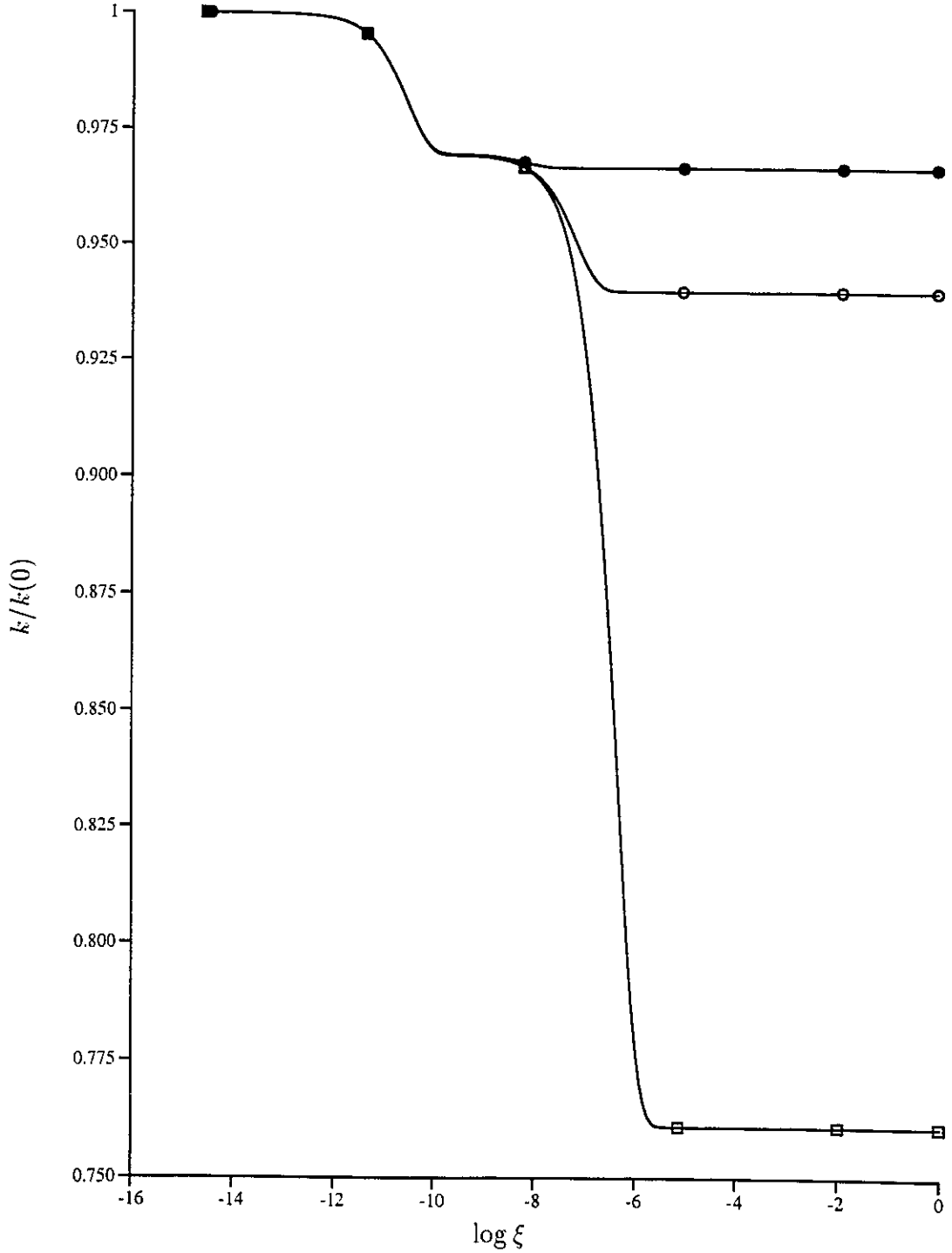

Figure 3.1: The ratio $k/k(0)$ vs. $\log \xi$ for R = He, $T = 300\,\mathrm{K}$ and various values of [Br]/[R]: (●) 0.01, (○) 0.1, (□) 1.0. ξ is the extent of reaction as defined on page 18. Representative points are shown merely to distinguish among curves.

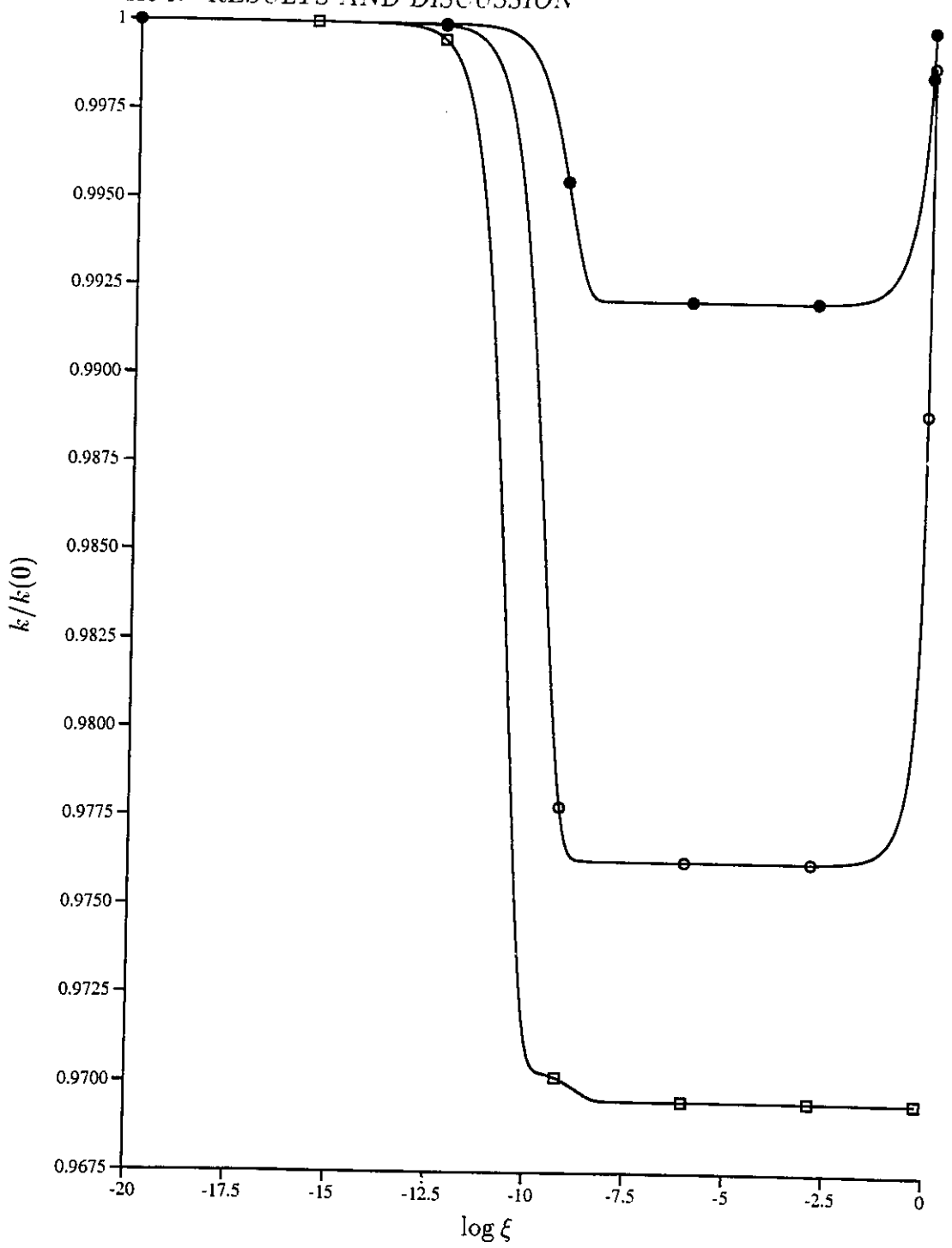

Figure 3.2: The ratio $k/k(0)$ vs. $\log \xi$ for R = HCl, $T = 300\,\mathrm{K}$ and various values of [Br]/[R]: (●) 0.01, (○) 0.1, (□) 1.0. ξ is the extent of reaction as defined on page 18. Representative points are shown merely to distinguish among curves.

Table 3.3: Comparison of Numerical and Analytical Results

R	T/K	[Br]/[R]	$k/k(0)$	HT Results[25]
He	300	0.01	0.96672	0.96672
		0.1	0.939979	0.939975
		1	0.76083	0.76083
	500	0.01	0.16409	0.16409
		0.1	0.084968	0.084968
		1	0.073802	0.073802
	1000	0.01	0.72021	0.72021
		0.1	0.22937	0.22937
		1	0.030605	0.030605
	2000	0.01	0.90912	0.90912
		0.1	0.55283	0.55283
		1	0.15738	0.15738
	3000	0.01	0.9547148	0.9547150
		0.1	0.690782	0.690786
		1	0.22999	0.22999
HCl	300	0.01	0.99215	0.99215
		0.1	0.97628	0.97628
		1	0.96953	0.96953
	500	0.01	0.81716	0.81716
		0.1	0.34289	0.34289
		1	0.11164	0.11164
	1000	0.01	0.93196	0.93134
		0.1	0.61890	0.61747
		1	0.153344	0.153346
	2000	0.01	0.97308	0.97032
		0.1	0.79318	0.78081
		1	0.33730	0.33731
	3000	0.01	0.98748	0.98510
		0.1	0.88672	0.87085
		1	0.43926	0.43926

again; i.e., where significant reaction took place during the transient phase.

3.4 Internal Validation

It is possible to adjust the input parameters to BIMSIM in various ways to check that the behavior obtained agrees with the behavior expected. Three such ways were employed.

First, the absolute concentration of the species was changed (keeping the relative concentrations the same) over 12 orders of magnitude. The expectation (at least computationally) is that there will be no change in the value obtained for k, or in the shape of the curve for $k(t)$. (In fact, initially the results were not concentration-independent and this led to the discovery of a rather subtle bug in the program.) Second, the value of k_{10}, the energy transfer rate constant for the transition from $v = 1$ to $v = 0$, can be made larger and larger. As the relaxation rate gets larger and larger, there is less and less depletion of the high vibrational levels caused by reaction from those levels and, therefore, the value of k obtained should approach k_{eq} and the $k(t)$ curve should approach the trivial straight line $k = k_{eq}$ if the program is working correctly. Third, the ratio [He]/[HCl] can be varied from 0 (where HCl is the species principally responsible for relaxation) to 10^6 (where He is the principal relaxer). In this case, it is expected that the value of k obtained and the shape of the $k(t)$ curve will make a smooth transition from the R = HCl behavior to the R = He behavior. In all three cases, the results obtained agree with the expected results.

The temperature dependence of $k/k(0)$ also turned out as expected, with the value becoming smallest at temperatures $\sim$ 500 K[5]. It is at around 500 Kthat the disparity between k_{10} and k_{eq} becomes largest. The disparity is smallest at 300 K, thus explaining why Lim and Truhlar found only 1% effects for their room-temperature study[12].

3.5 Time Dependence of k

The time dependence of k can be explained completely (at least qualitatively) in such a way that no features of a plot of k vs. t remain unexplained. This explanation also serves to further validate the program.

All of the features of such curves can be explained by looking at plots of $y_v/y_v(0)$ vs. $\log\xi$, where y_v is the amount of HCl in level v relative to the total amount of HCl present, and $y_v(0)$ is the value of y_v at $t = 0$. In order to best visualize the features of such plots some function of $y_v/y_v(0)$ is chosen as the ordinate rather than $y_v/y_v(0)$ itself. Figures 3.3 and 3.4 show how the fractional populations of each of the five vibrational levels (corresponding to $v = 0$–4) varies as a function of time (actually as a function of $\log\xi$). As with Figures 3.1 and 3.2, the figures are for the six sets of initial conditions for which $T = 300\,\mathrm{K}$. The features of the plot shown in Figure 3.1 for R = He and of the plot shown in Figure 3.2 for R = HCl are significantly different and will be examined separately in the next two sections.

3.5.1 R = He

For R = He (Figure 3.1) there are three features that require explanation: the dip at $\log\xi = -11$; the second dip at $\log\xi = -7$; and the fact that the curves for the three mole fractions remain the same until $\log\xi = -8$ and then split showing three final values of k/k_{eq}.

The first feature is easily seen from Figure 3.3 as the effect of the depletion of y_2, y_3 and y_4 at $\log\xi = -11$, and the second feature also is easily seen as the effect of the depletion of y_1 at $\log\xi = -7$. The third feature is harder to explain. However, close examination of the y_2 curve and especially the y_3 and y_4 curves in Figure 3.3 reveals that these levels are seriously depleted (by as much as a factor of 10^{15} in the case of y_4) and so all these values decrease at $\log\xi = -11$ to the point where none of them makes any contribution to the phenomenological rate coefficient, k (and therefore the contribution made for all three mole fractions is the same). Note that for the conditions existing when R = He, i.e., [HCl] $\ll$ [He] such that [HCl] is also

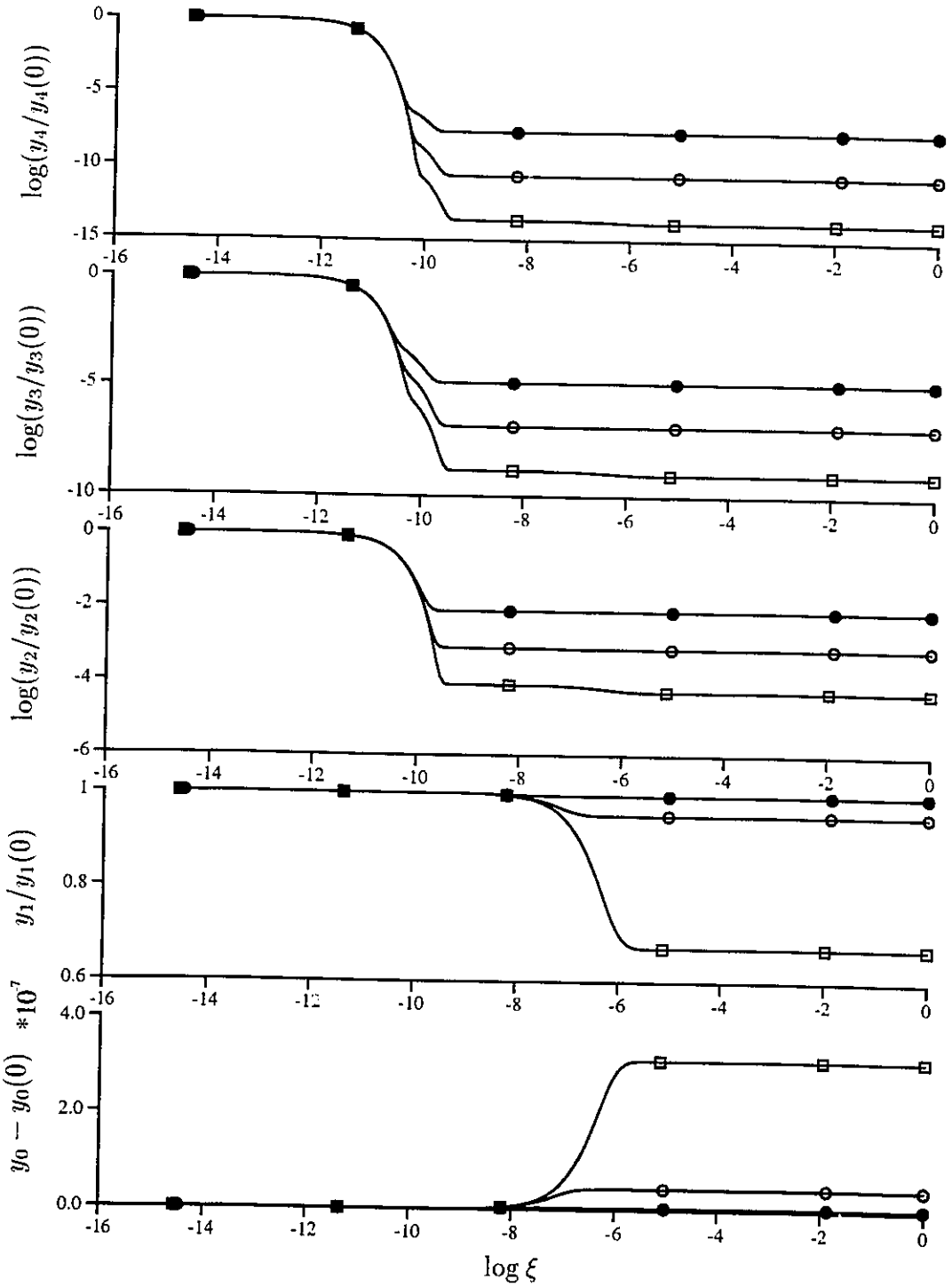

Figure 3.3: Changes in y_v vs. $\log \xi$ for v = 1–4, R = He, T = 300 K and various values of [Br]/[R]: (●) 0.01, (○) 0.1, (□) 1.0

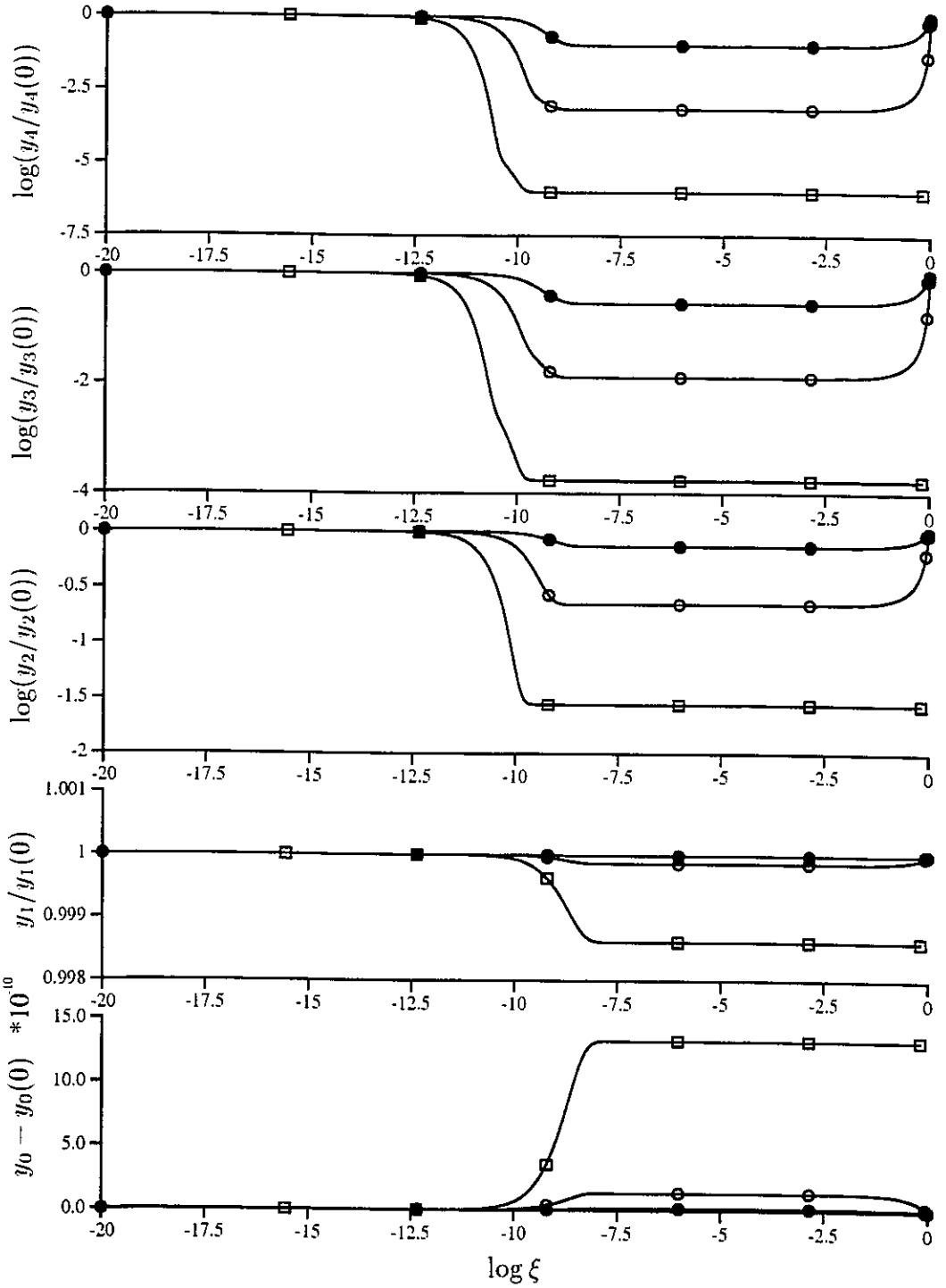

Figure 3.4: Changes in y_v vs. $\log \xi$ for v = 1–4, R = HCl, T = 300 K and various values of [Br]/[R]: (●) 0.01, (○) 0.1, (□) 1.0

$\ll$ [Br] (see p.17) then [Br] hardly changes as the limiting reagent HCl disappears. Consequently the relative rate of reaction and relaxation remains constant at all times, and the steady state values of k and of y_v show an exceptional constancy that is not in general a necessary feature of steady states (see below).

3.5.2 R = HCl

For R = HCl (Figure 3.2) there are again three specific features that require explanation: the shoulder on the curve for [Br]/[R] = 1 at $\log \xi = -9$; the difference in behavior between this curve and the other two at $\log \xi = -1$; and why these curves separate much earlier ($\log \xi = -13$) than the corresponding R = He curves.

The y_2, y_3 and y_4 curves of Fig. 3.4 clearly cause the bulk of the change in k at $\log \xi = -10$ (the first feature). Examination of the y_1 curve shows that (relatively speaking) the depletion for the [Br]/[R] = 1 curve is much larger than the other two curves and this depletion means that y_2 makes a slightly smaller contribution to k after $\log \xi = -9$ than before. The third feature is again a direct reflection of the behavior of the y_1, y_2, y_3 and y_4 curves of Fig. 3.4. For the [Br]/[R] = 1 curve, [Br] = [HCl] throughout the course of the reaction but for the other curves, HCl is in excess. This means that as soon as a finite amount of reaction has taken place ($\log \xi = -1$), the amount of Br begins to decrease but the amount of HCl does not decrease significantly. However, reaction rates depend on [Br] ($rate \propto$ [HCl(v)][Br]) and the relaxation rates do not ($rate \propto$ [HCl(v)][HCl]). Therefore, the overall reaction rate decreases in proportion to the overall relaxation rate as time increases. This, in turn, increasingly allows relaxation to "catch up" to reaction. In other words, the reaction behavior becomes less and less non-equilibrium (that is, less and less depletion of levels $v = 1, 2, 3, 4$) so that $k \rightarrow k_{eq}$ as $\log \xi \rightarrow 0$. Note that, even though $k \rightarrow k(0)$ and $y_v \rightarrow y_v(0)$ at late times, this is still the steady-state regime, since once a steady distribution is established then a steady state prevails thereafter even if reagents disappear significantly. A steady state k does not necessarily mean a constant k. For example Teitelbaum's steady state expression still applies, as long as the appropriate reagent concentrations are input.

Chapter 4. Conclusions and Future Work

4.1 Conclusions

BIMSIM has been validated at the same time showing that the analytical expression of [5] is valid. BIMSIM was used for extending the calculations of Teitelbaum to include time dependent details and for extending the calculations of Lim and Truhlar to the more interesting case of the higher temperature endothermic reaction $Br + HCl \rightarrow HBr + Cl$. It was found that non-equilibrium effects can be impressive. The magnitude depends on temperature, [Br], [HCl], and [He]. The bulk of the reaction occurs in the steady-state regime, but interesting effects show up as the reagents begin to disappear.

It is very difficult to produce a computer program which is correct—that is, one which calculates what you intend it to calculate. In other words, one which does not produce output which is in error. Even when independently calculated or measured results are available, if they differ from the results of the program in question one cannot say with certainty which result is correct and which is not (assuming that not both results are incorrect!). One has a renewed respect and healthy scepticism at the same time for other existing computational work—especially more complicated work such as *ab initio* calculations.

4.2 Contribution to Knowledge

BIMSIM is a computational tool which can be easily used to learn the details of non-equilibrium processes in bimolecular reactions. It is clear that the non-equilibrium effect is very sensitive to the chemical environment. Experimentalists now have a tool to guide them to extract steady-state rate constants and to allow them to intelligently compare results with other workers for whom conditions may be significantly

different. Theoreticians are also guided to proper comparisons with experiments.

4.3 Future Work—Chemical Aspects

Work yet to be done can be broken down into three categories: further analysis of already obtained data; using BIMSIM's existing capabilities to explore other initial conditions or levels of approximation; and extending BIMSIM, adding features and removing currently existing approximations.

It would be useful to analyse the data already obtained to see if one can use the relative populations to calculate a single "vibrational" temperature (by assuming a Boltzmann distribution, obtaining a best fit to the data and looking at the statistical error associated with this best fit) as a function of time. This might, for example, lead to an analytical expression for T(t) and, therefore, to an improved explicit expression for the correction factor obtained by Teitelbaum[13]. It would also be useful to examine the contribution each level makes to $k(t)$ more closely in order to answer questions such as can a single level, m^*, from which reaction is primarily occurring always be identified (this is an assumption of ref. [13] but not of ref. [5]).

It may be that if a particular vibrational level is selectively excited (through a laser pulse, for example) that a steady state is never established. Therefore, this is a particularly interesting set of initial conditions that needs to be explored. It can be done using BIMSIM's current capabilities.

Several desirable enhancements to BIMSIM can be itemized:

- Include back reaction, especially to allow for closer examination of conditions where k is still changing when ξ becomes large enough to allow significant back reaction.

- Build a sensitivity analysis capability into the program.

- Explore the possibility of using Groebner bases[26] as a way of merging existing analytical (e.g., the work of Teitelbaum *et al.*[13, 5, 14]) and theoretical treatments (e.g., this work).

- Include VV processes.
- Allow products AB and C to be relaxers.
- Calculate k_f/k_r to see if the kinetic mass action law is valid during non-equilibrium chemical reactions.
- Make a direct comparison, if possible, with Lim and Truhlar's method, using their input rate constants in BIMSIM.

4.4 Future Work—Computational Aspects

The author would like to make various computational improvements to BIMSIM as follows:

- Modify BIMSIM so that it can generate a separate version of itself for each reaction to be explored. In this way, only the input parameters that will change within a series of experiments performed need be provided, all the others having been determined once and isolated in a separate input file.
- The input mechanism needs to be more robust to allow more natural input, more errors in input detected, and error recovery where possible.
- The code which prompts the user when in interactive mode needs to be rewritten to allow the program to be smaller and more easily extended.
- Convert the code from Fortran to FWEB (an extension of Fortran which allows the system documentation for the program to be automatically generated from the source code).

Appendix A. BIMSIM Description

The important features of BIMSIM will be outlined in this appendix.

A.1 Program Flow

BIMSIM is a conventional program in the sense that it begins by obtaining input parameters (routines which obtain such parameters mostly have names starting with the letters "PIK" in BIMSIM) to indicate what reaction system, what temperature, etc., then it proceeds to calculate what is required (in this case what is required is the integration of the master equation—the name of the routine here is SOLVER) and writes results as they are available to various files (routine DUMP).

A.2 Flexible Input

BIMSIM has two ways of obtaining input parameters: the user can be prompted for input from the keyboard (interactive mode), or input can be from a file (batch mode). Or, in fact, input can be a combination of the two; if the user selects batch mode and end-of-file is reached before all parameters are specified then the program switches to interactive mode.

In any case, the program always echoes input to another file, formatted in such a way that the user can use this file as input on the next run. When using the program interactively, this allows the program to be interrupted and later restarted bringing the user back to the point where he left off. It also allows the use of interactive mode to build an input file through prompts from the program, making the program easy to use but not necessarily forcing the user to respond to a long series of prompts each time the program is run.

In fact, each prompt has a default value associated with it (obtained by hitting

<ENTER>) and so a valid input file can be built simply by selecting the default at each prompt. This mechanism has the added benefit that, if desired, a user of BIMSIM can begin using the program right away without having to read any documentation or having to create an input file (or even understanding what the program does!).

In addition, when the input expected by BIMSIM is a physical quantity that has units associated with it, the user must supply the units, but many different units are allowed by the program and converted to the equivalent SI units (the program always uses SI internally). For example, pressure can be specified in Torr, kiloPascals, atmospheres, p.s.i., and bars.

Another innovative feature of the program is that it has so-called novice and expert modes. If novice mode is selected, more information is given to the user at each prompt to help him make his selection—in particular the units which are allowed at a given prompt are displayed when in novice mode. In expert mode, this information is suppressed—presumably once a user has been using BIMSIM long enough he will not need this information cluttering up his screen and can request that it not appear.

BIMSIM is designed to handle a very wide range of initial conditions, experiments, reactions, and levels of approximation. There are no less than 115 separate prompts in the program; even if each prompt allows for only 2 branches this means there are 2^{115} different situations that BIMSIM can handle. For example, when determining the energy level of each vibrotational level, the user has six choices: SHO, 3-parameter Morse function, partial spectroscopic fit (vibrational constants only), file of v, $E(v)$ pairs, full spectroscopic fit, or a file of v, J, $E(v)$ triples. In some of these cases the energies of the rotational sub-levels are not yet determined and the user is further prompted to choose from a rigid rotor approximation or a partial spectroscopic fit to determine these energies.

A.3 Awareness of Experimental Conditions

The program easily allows the simulation of typical experimental conditions such as those of laser pulse experiments where one level is selectively excited by a laser from the ground vibrational level to some higher level. It also allows input data to be sparse and does various interpolations to provide any missing values, if requested. It recognizes that data may not be available in some cases, for example, even if BIMSIM calculates that 11 vibrational levels exist below the dissociation energy, it will allow for a model which has fewer than 11 levels (according to the data available).

A.4 Portability

BIMSIM conforms to the FORTRAN77 standard. The code contains no system-dependent sections. A port to a new system requires the equivalent of a CMS EXEC be provided to link FORTRAN I/O units to external files (and to the terminal), and to execute the program (assuming it has been successfully compiled). BIMSIM also requires any system to which it is to be ported have available the IMSL subroutine library. The IMSL subroutine library is a commercial product available for many different systems from IMSL, Inc., 2500 CityWest Blvd., Houston, TX.

A.5 Design Aspects

BIMSIM is designed to be easily modifed by those who use it (and not just by the author). There are extensive comments throughout the program. It consistently uses modular structured programming techniques. In many cases, routines which take user-defined action are already built into the program—the user has only to modify the existing shell, inserting the desired code.

The program is also designed to be robust and correct rather than fast. For example, a routine called EXIST is used in many places to greatly simplify the coding: if a value is to be calculated for each existing transition the program does

something like (in pseudo-code)

$$\text{if exist}(v, J, v', J') \text{ then calculate value}$$

where v, J are the quantum numbers for the initial level and v', J' are the quantum numbers for the final level of a particular transition. Input is extensively validated and there are many checks in the program to make sure input data is reasonable and that no data are accidentally missing from the input.

Appendix B. Graphs

In this appendix graphs of $k/k(0)$ vs. $\log \xi$ and graphs of various functions of y_v vs. $\log \xi$ are presented for all the initial conditions specified in section 3.1. In each graph ξ is the extent of reaction as defined on p.18, and representative points are shown merely to distinguish among curves.

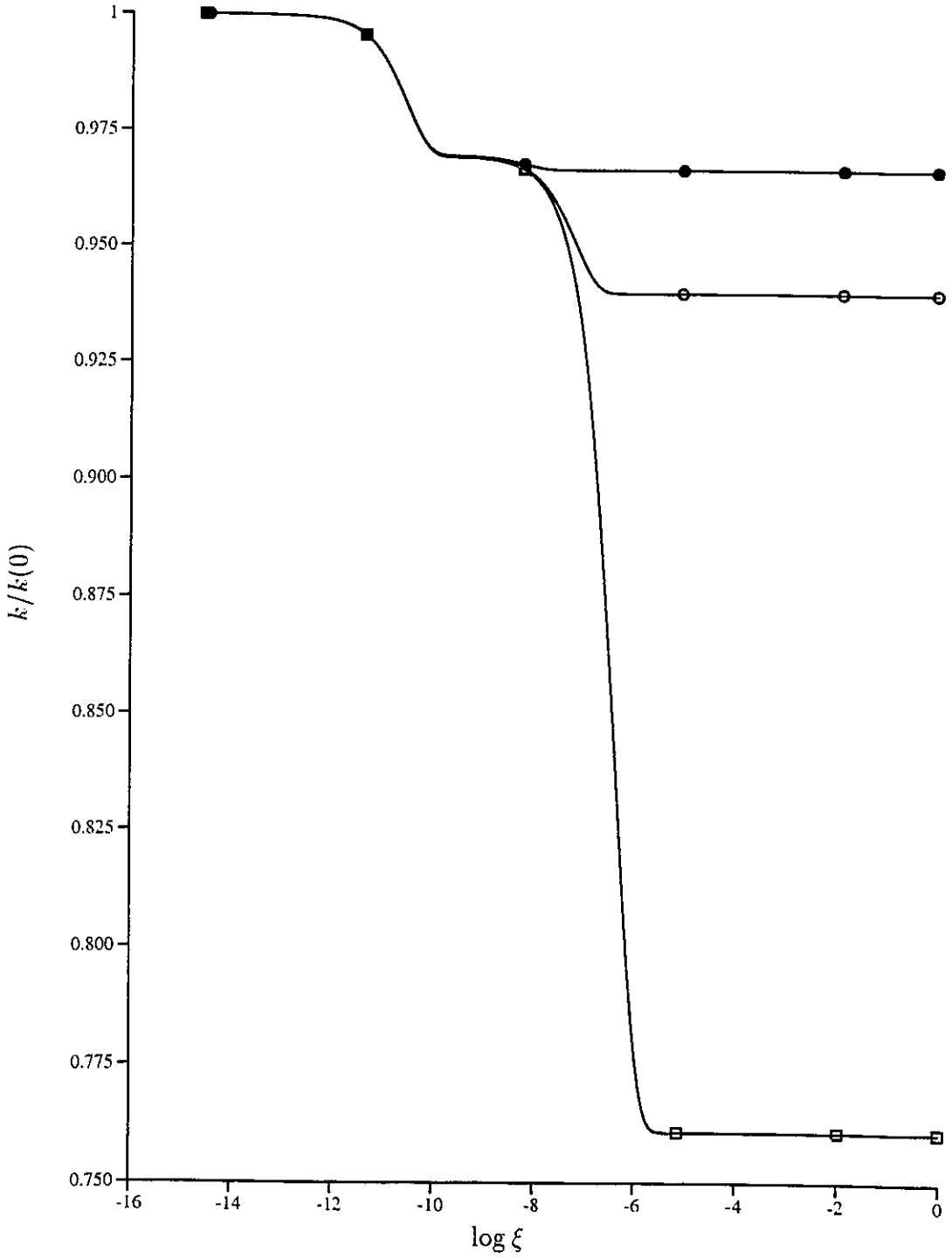

Figure B.1: The ratio $k/k(0)$ vs. $\log \xi$ for R = He, $T = 300\,\mathrm{K}$ and various values of [Br]/[R]: (●) 0.01, (○) 0.1, (□) 1.0

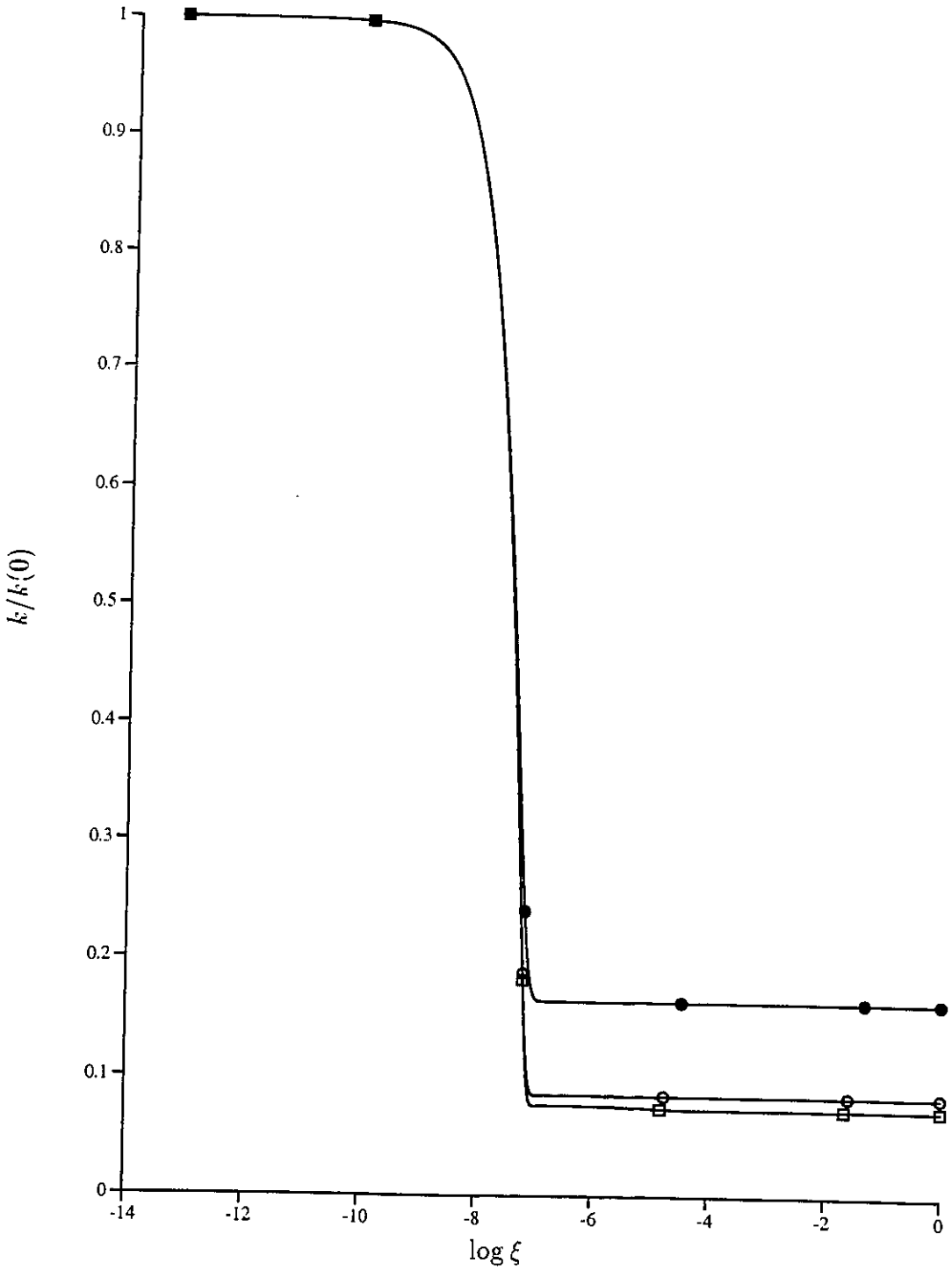

Figure B.2: The ratio $k/k(0)$ vs. $\log \xi$ for R = He, $T = 500\,\mathrm{K}$ and various values of [Br]/[R]: (●) 0.01, (○) 0.1, (□) 1.0

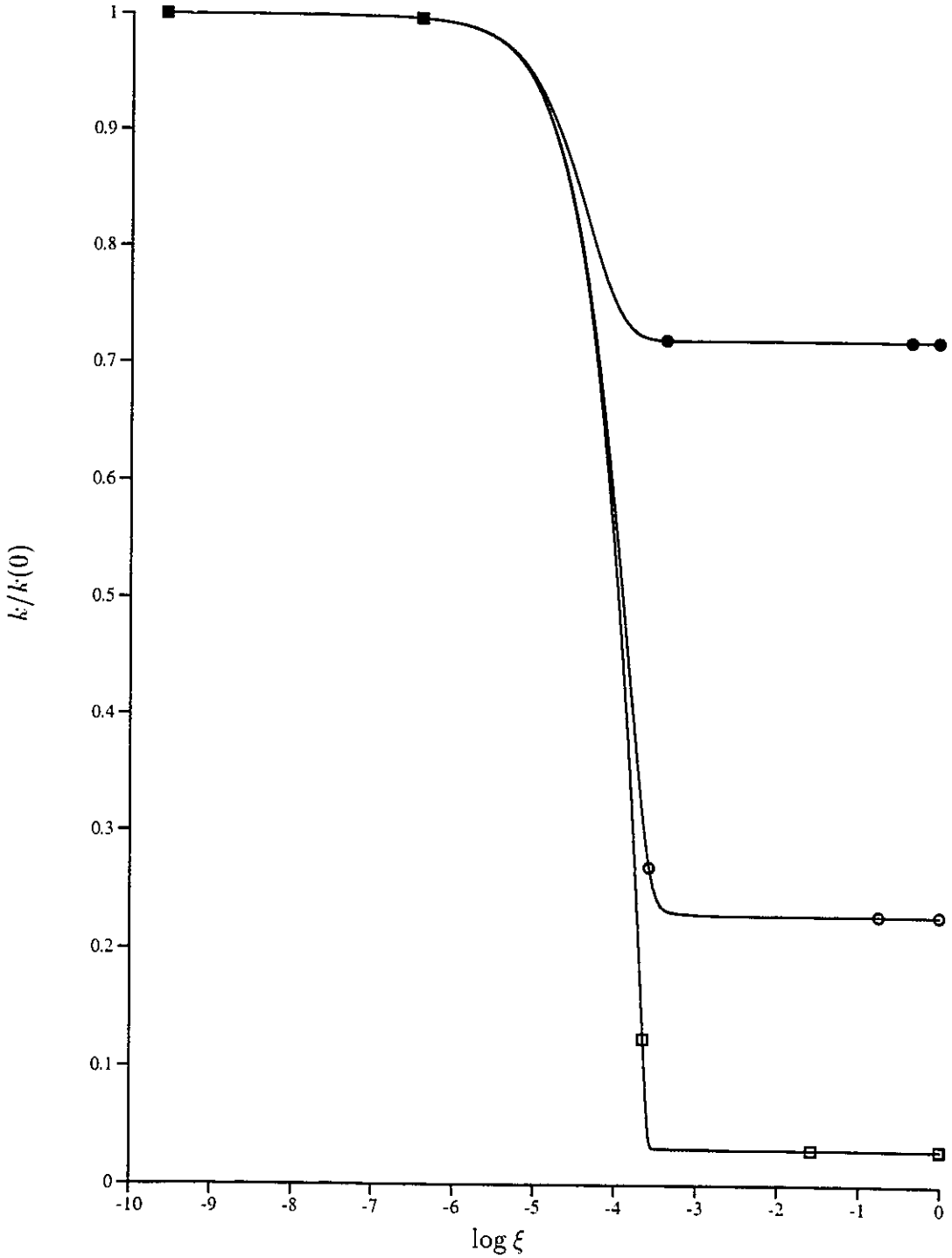

Figure B.3: The ratio $k/k(0)$ vs. $\log \xi$ for R = He, $T = 1000\,\mathrm{K}$ and various values of [Br]/[R]: (●) 0.01, (○) 0.1, (□) 1.0

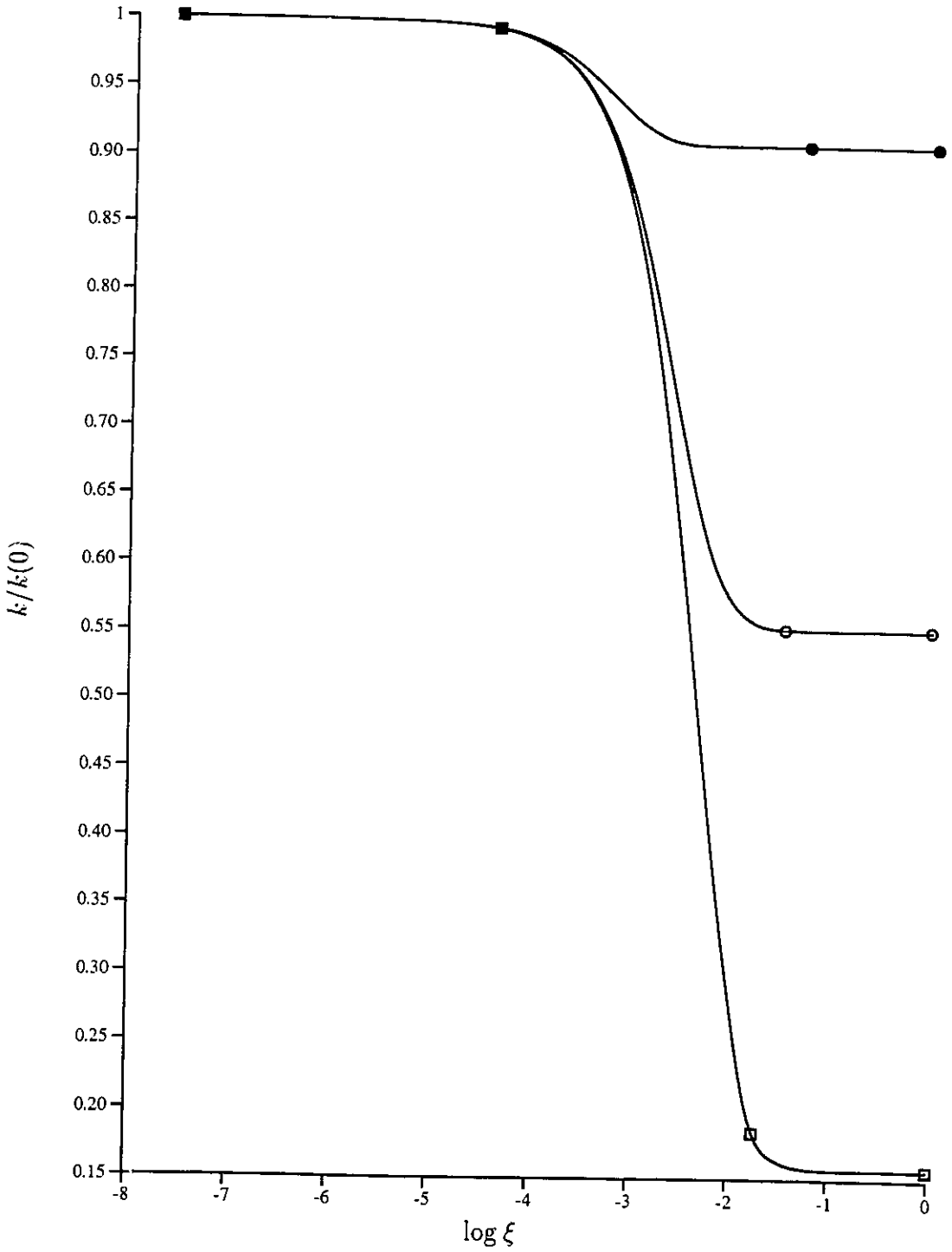

Figure B.4: The ratio $k/k(0)$ vs. $\log \xi$ for R = He, $T = 2000\,\mathrm{K}$ and various values of [Er]/[R]: (●) 0.01, (○) 0.1, (□) 1.0

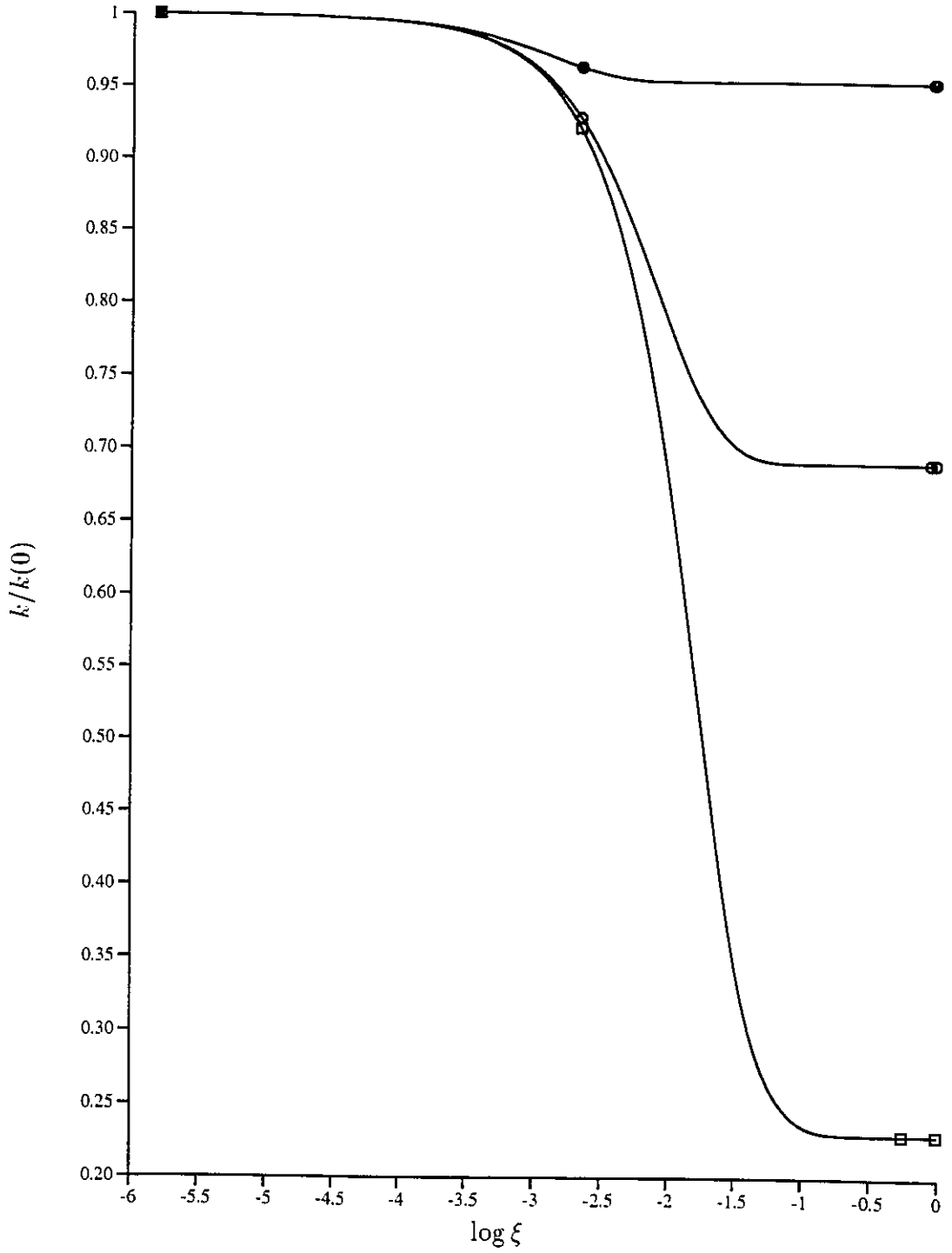

Figure B.5: The ratio $k/k(0)$ vs. $\log \xi$ for R = He, $T = 3000\,\mathrm{K}$ and various values of [Br]/[R]: (●) 0.01, (○) 0.1, (□) 1.0

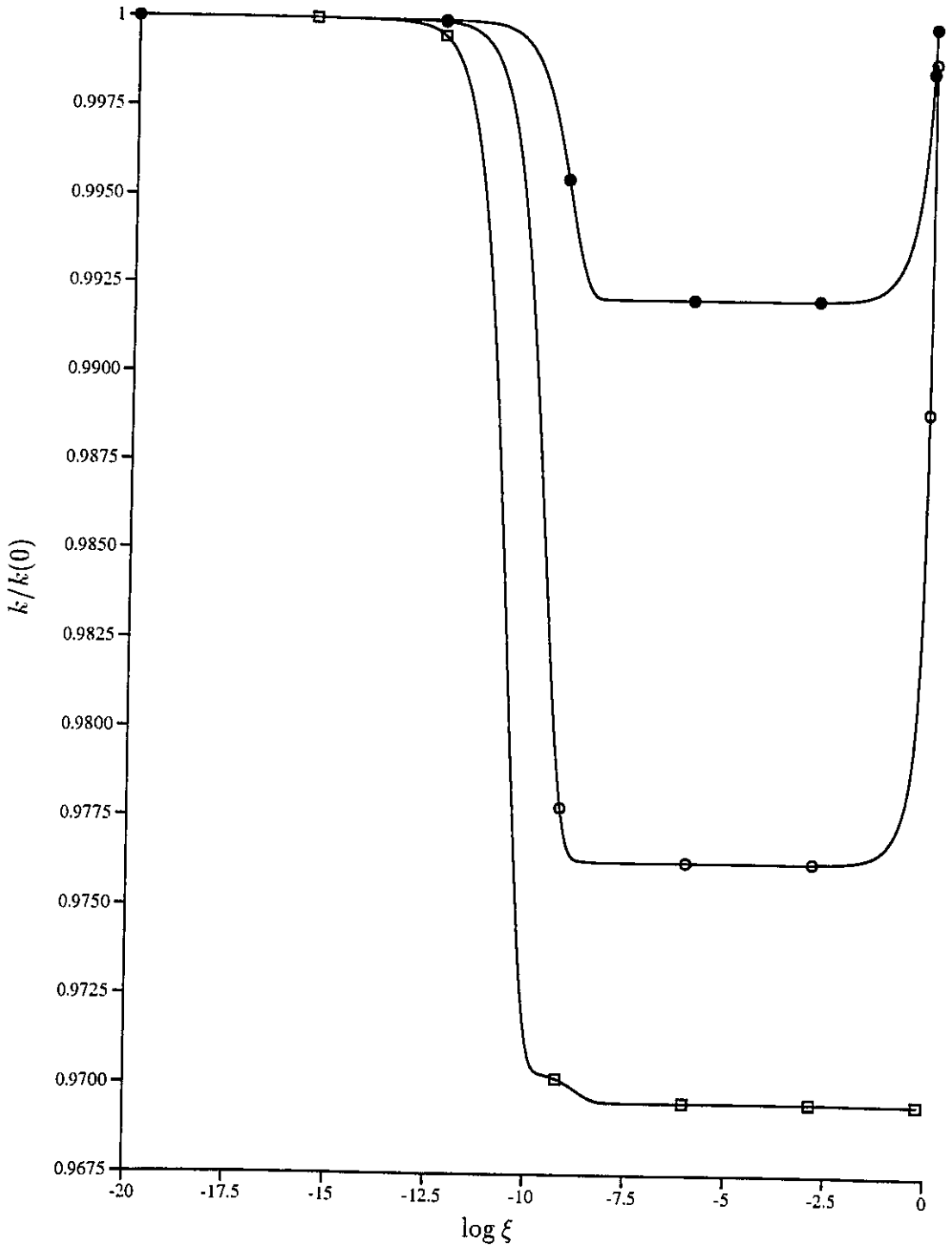

Figure B.6: The ratio $k/k(0)$ vs. $\log \xi$ for R = HCl, $T = 300\,\mathrm{K}$ and various values of [Br]/[R]: (●) 0.01, (○) 0.1, (□) 1.0

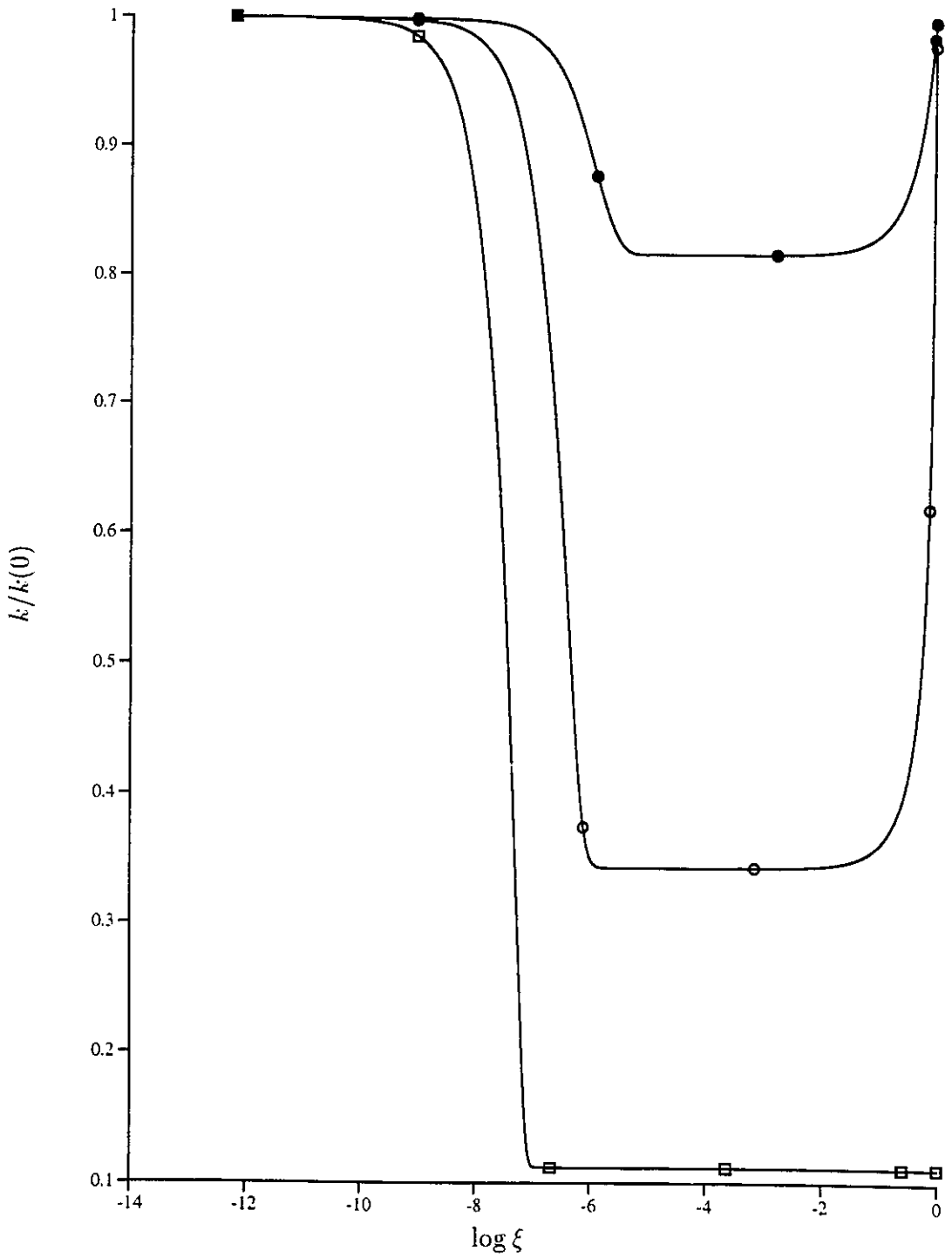

Figure B.7: The ratio $k/k(0)$ vs. $\log \xi$ for R = HCl, $T = 500\,\mathrm{K}$ and various values of [Br]/[R]: (●) 0.01, (○) 0.1, (□) 1.0

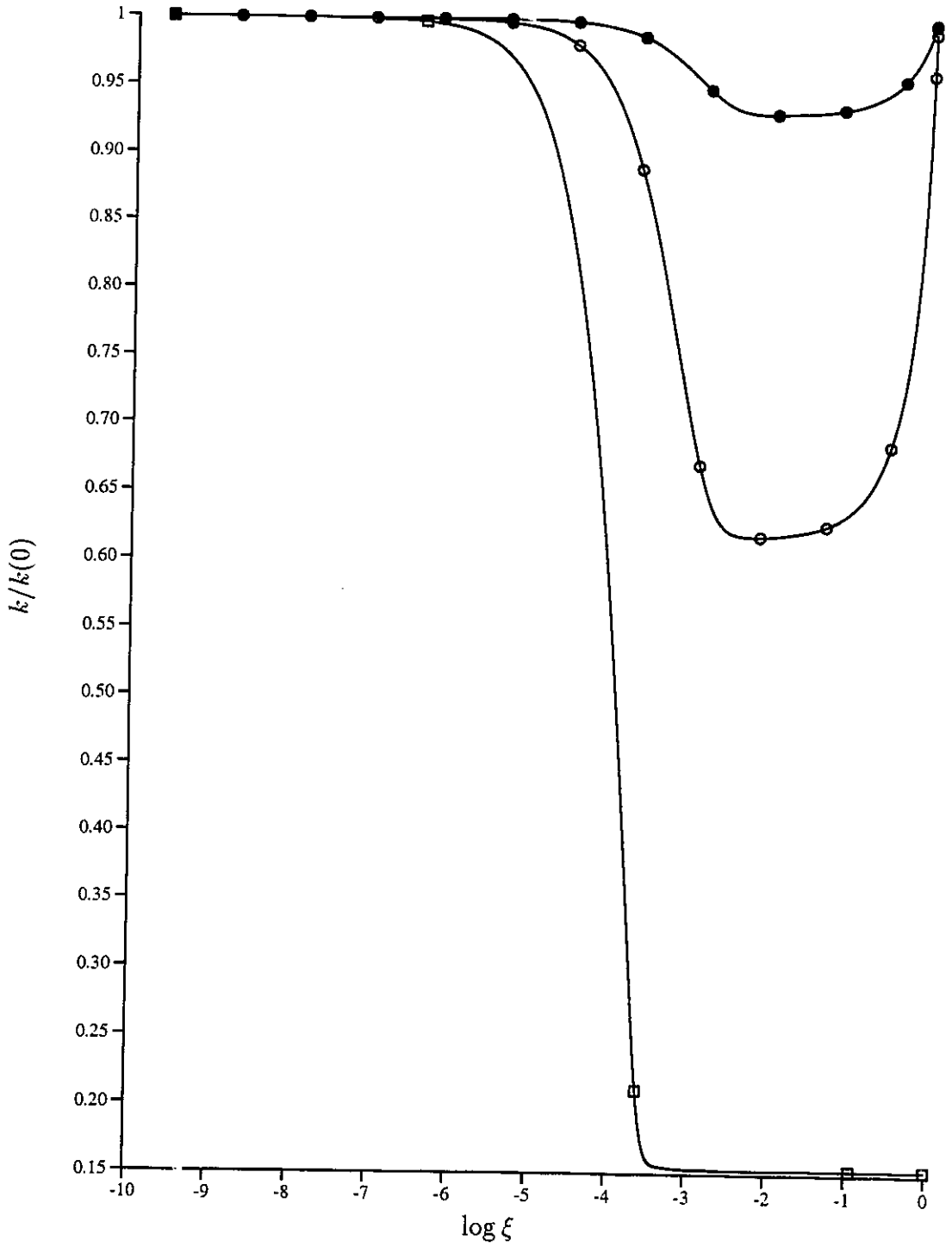

Figure B.8: The ratio $k/k(0)$ vs. $\log \xi$ for R = HCl, $T = 1000\,\mathrm{K}$ and various values of [Br]/[R]: (●) 0.01, (○) 0.1, (□) 1.0

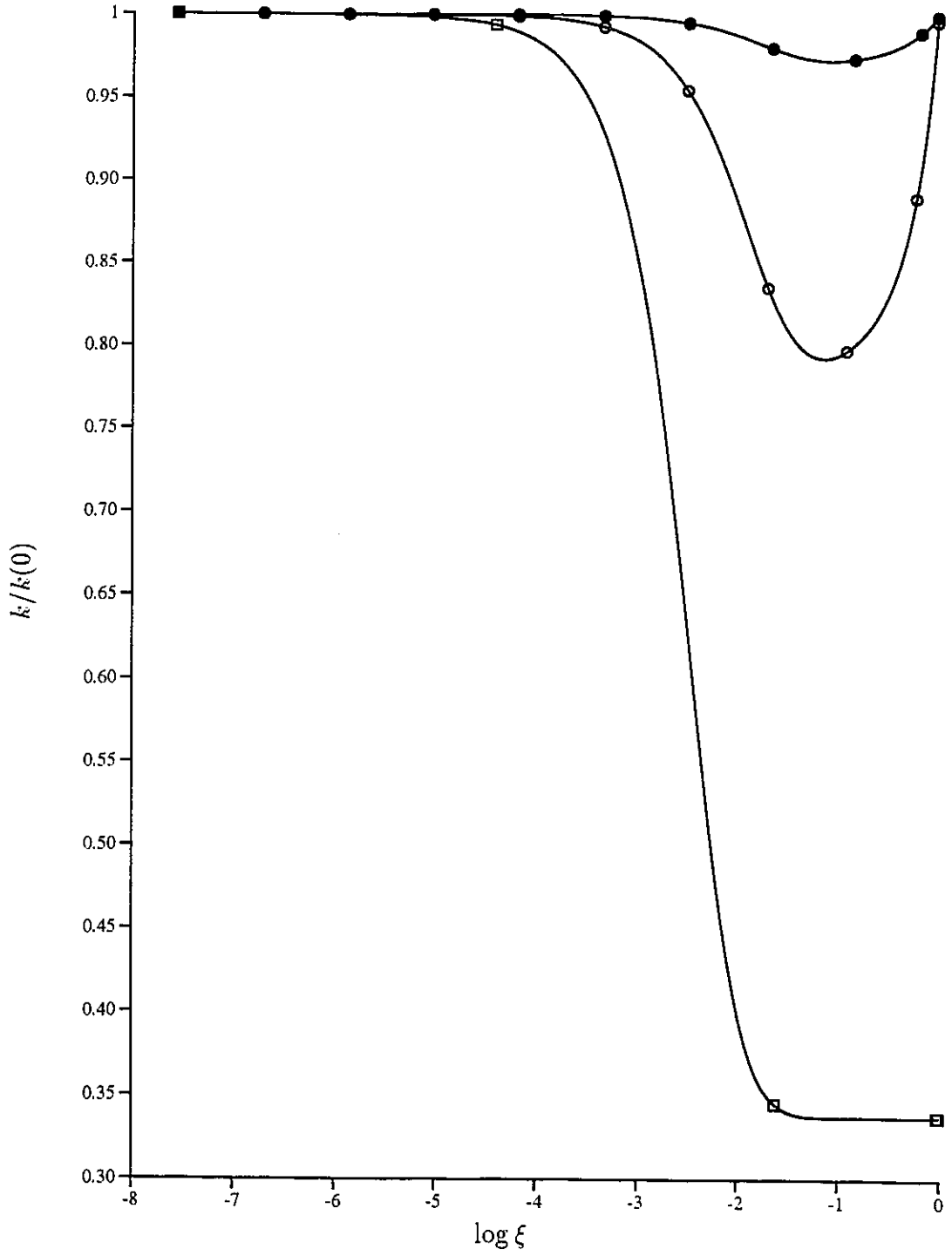

Figure B.9: The ratio $k/k(0)$ vs. $\log \xi$ for R = HCl, $T = 2000\,\mathrm{K}$ and various values of [Br]/[R]: (●) 0.01, (○) 0.1, (□) 1.0

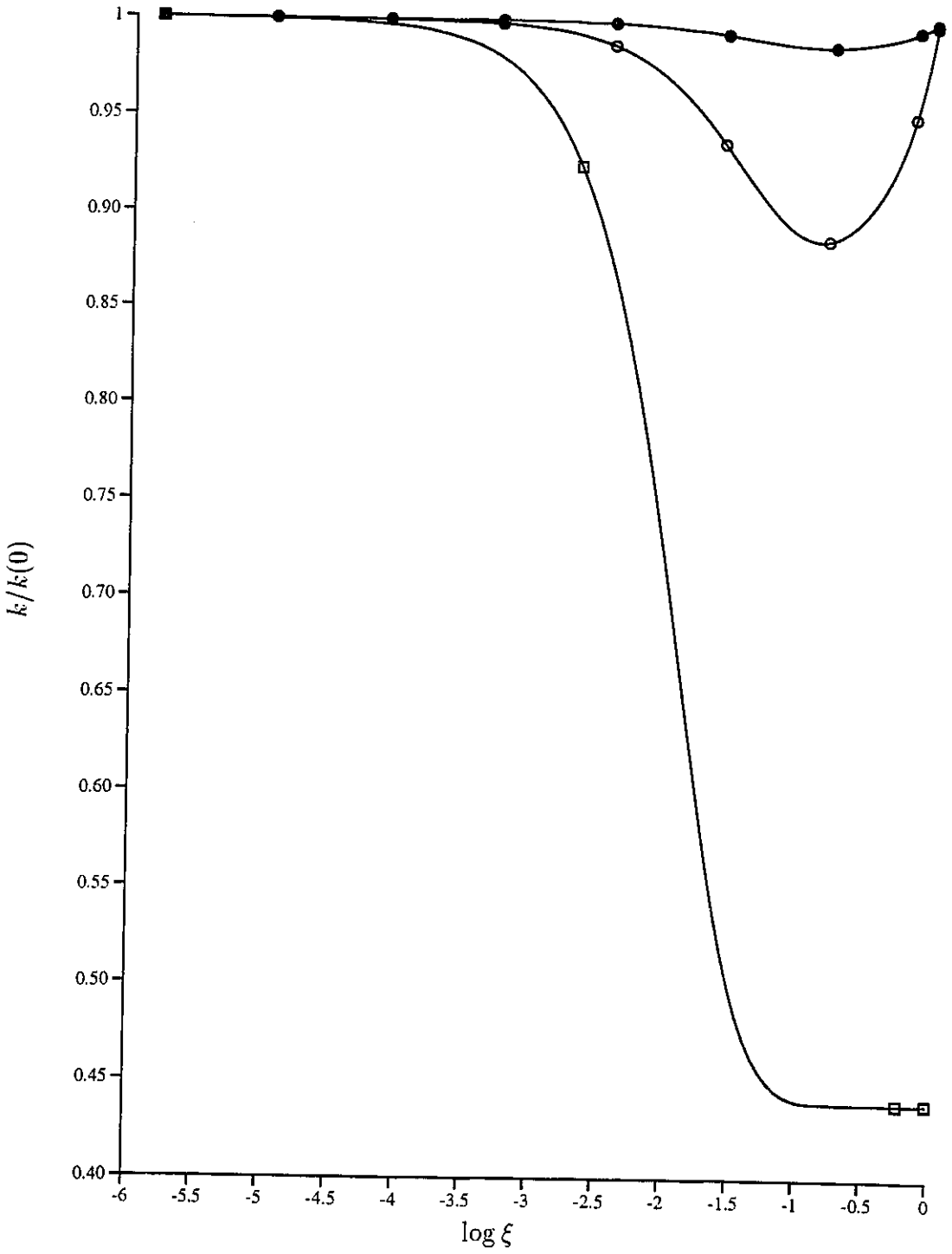

Figure B.10: The ratio $k/k(0)$ vs. $\log \xi$ for R = HCl, $T = 3000\,\mathrm{K}$ and various values of [Br]/[R]: (●) 0.01, (○) 0.1, (□) 1.0

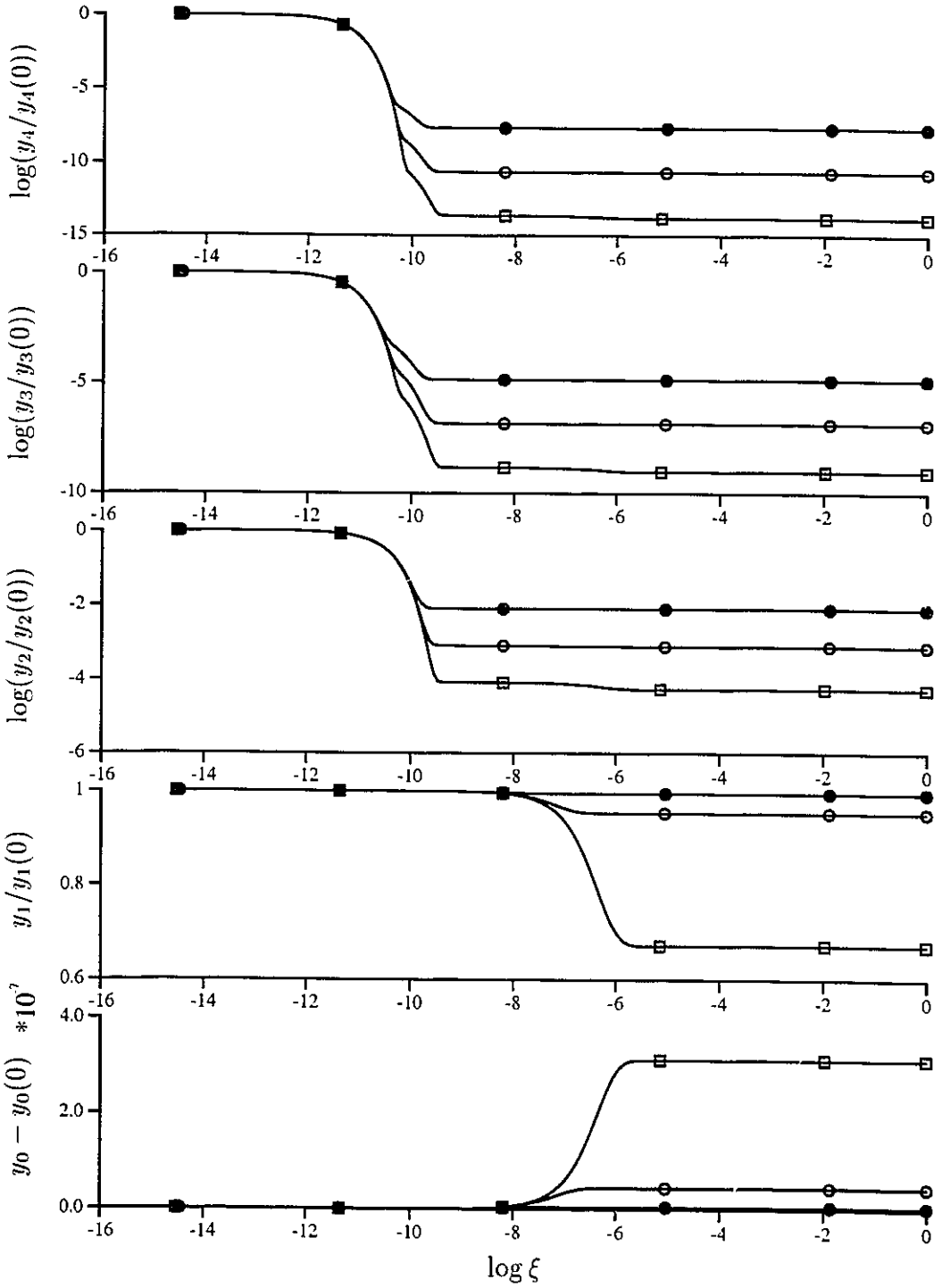

Figure B.11: Changes in y_v vs. $\log \xi$ for v = 1–4, R = He, T = 300 K and various values of [Br]/[R]: (●) 0.01, (○) 0.1, (□) 1.0

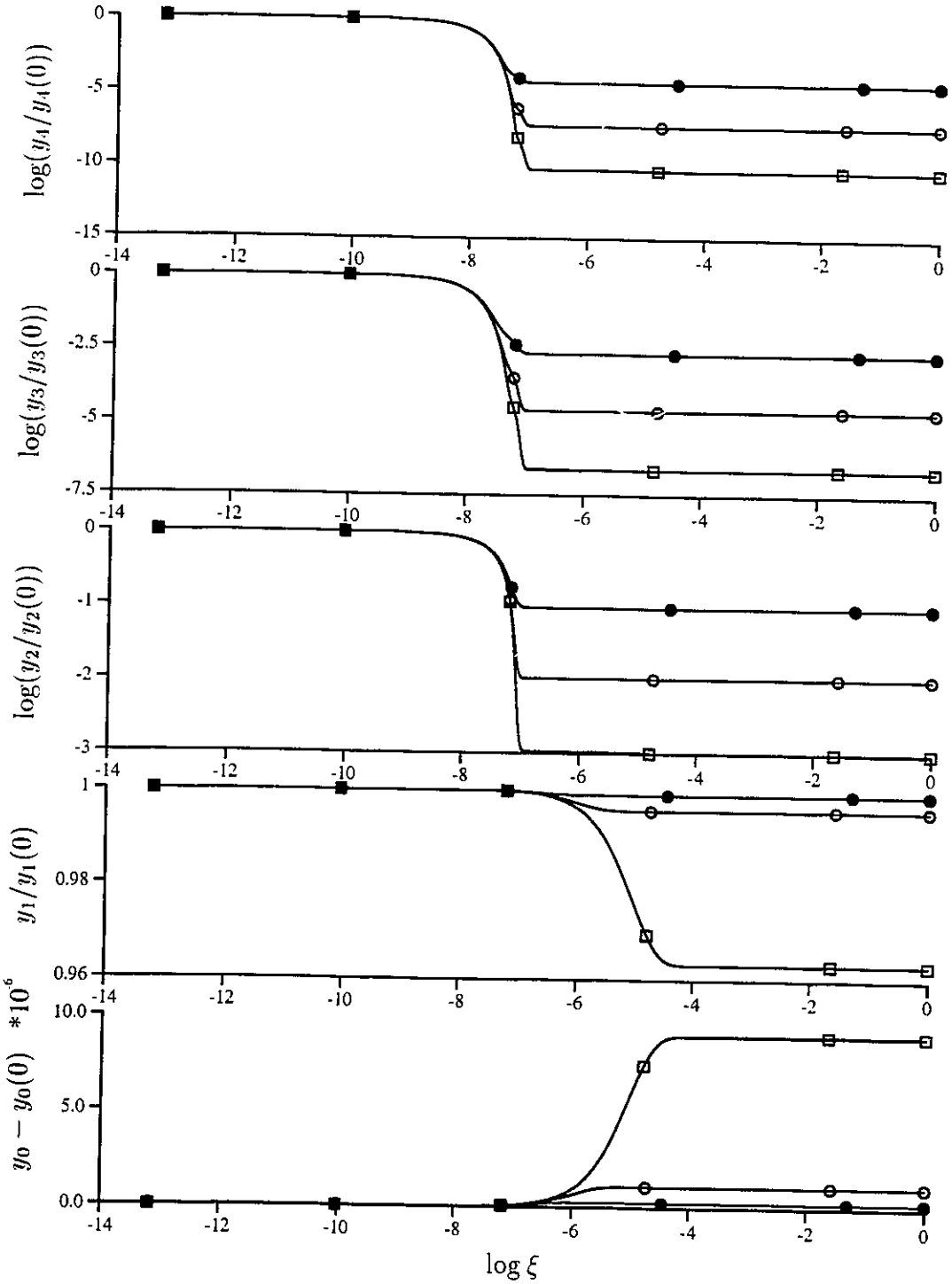

Figure B.12: Changes in y_v vs. $\log \xi$ for v = 1–4, R = He, T = 500 K and various values of [Br]/[R]: (●) 0.01, (○) 0.1, (□) 1.0

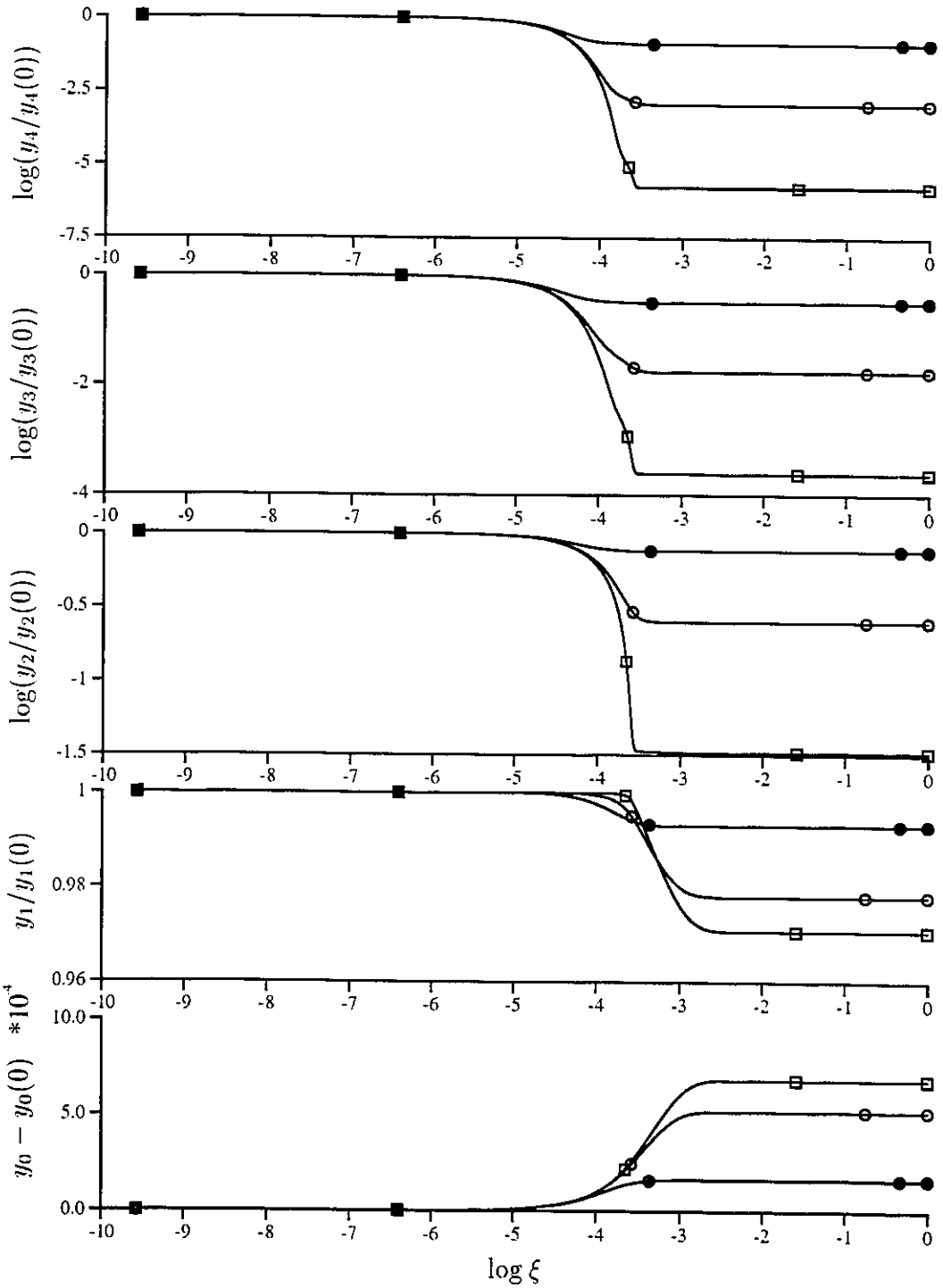

Figure B.13: Changes in y_v vs. $\log \xi$ for $v = 1$–4, R = He, $T = 1000\,\mathrm{K}$ and various values of [Br]/[R]: (●) 0.01, (○) 0.1, (□) 1.0

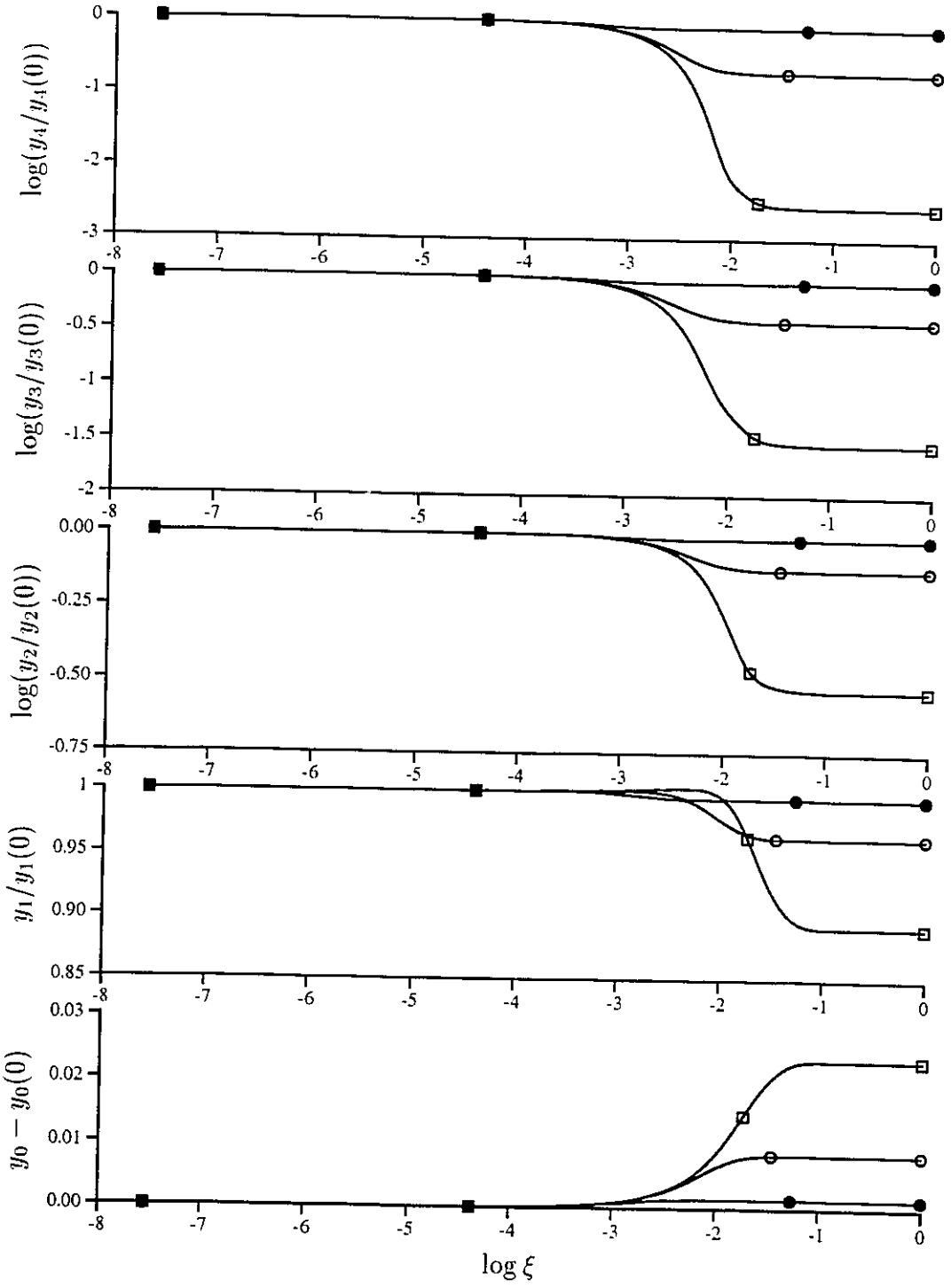

Figure B.14: Changes in y_v vs. $\log \xi$ for v = 1–4, R = He, T = 2000 K and various values of [Br]/[R]: (●) 0.01, (○) 0.1, (□) 1.0

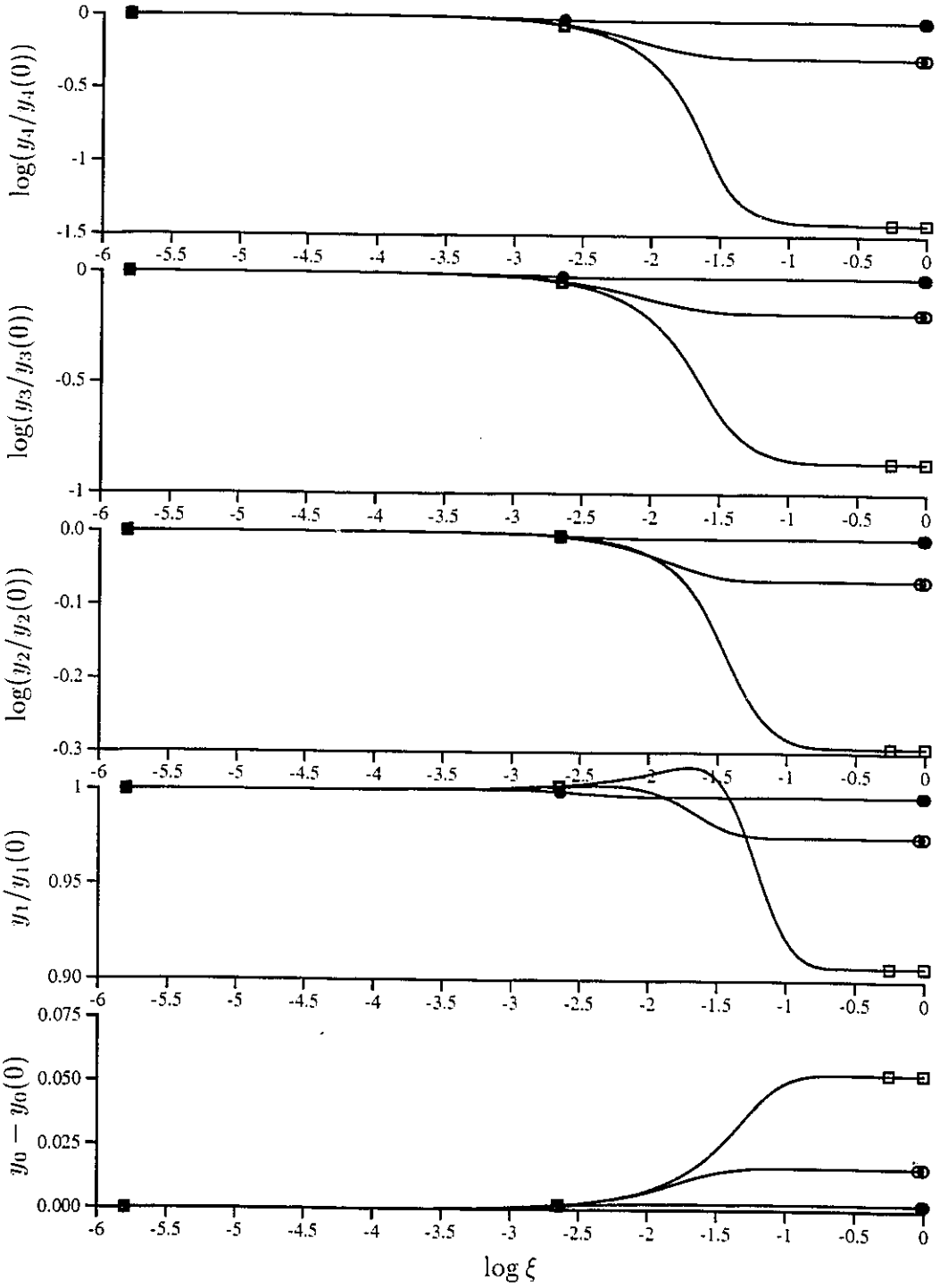

Figure B.15: Changes in y_v vs. $\log \xi$ for $v = 1$–4, R = He, $T = 3000\,\mathrm{K}$ and various values of [Br]/[R]: (●) 0.01, (○) 0.1, (□) 1.0

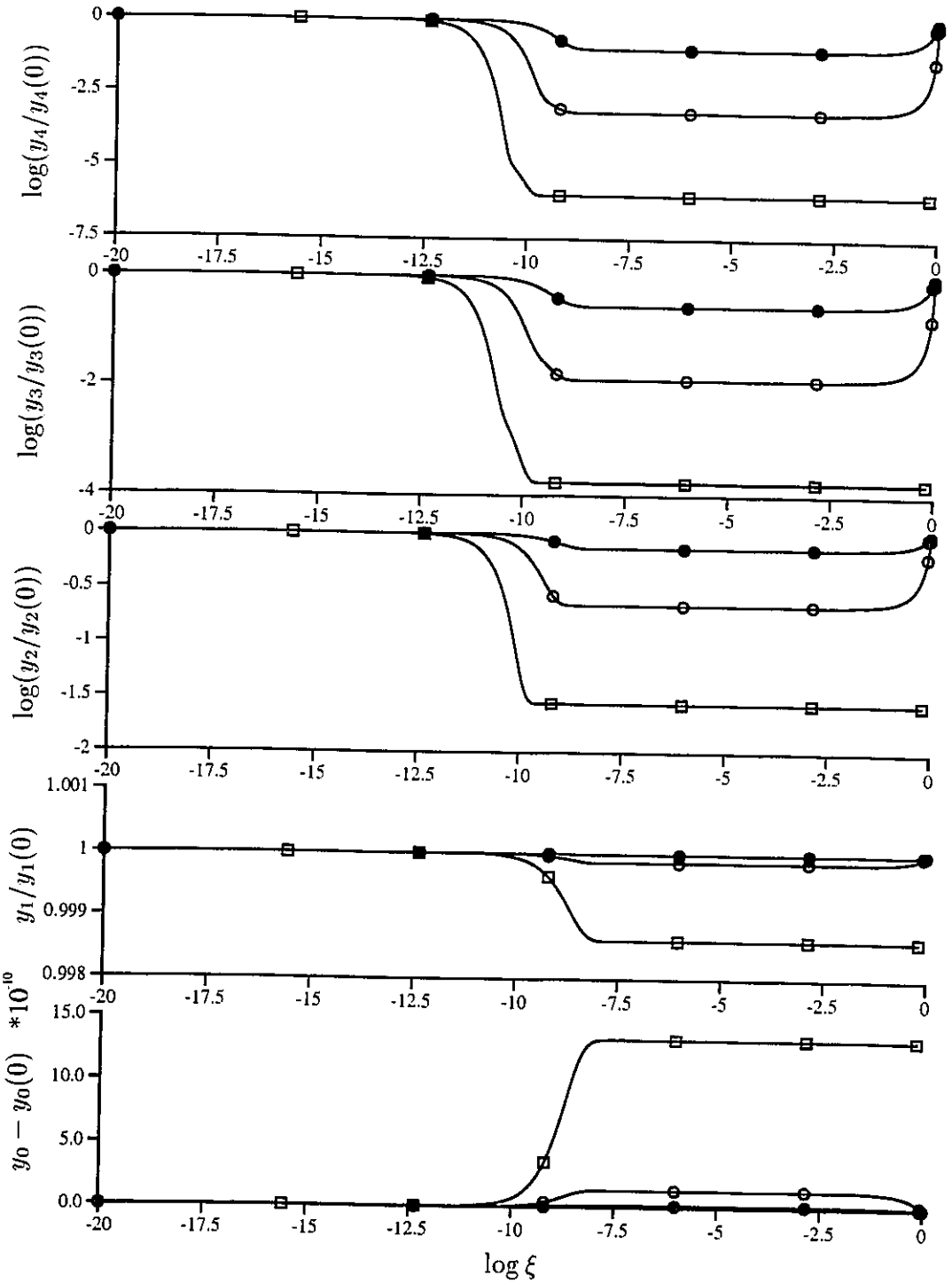

Figure B.16: Changes in y_v vs. $\log \xi$ for $v = 1$–4, R = HCl, $T = 300\,\mathrm{K}$ and various values of [Br]/[R]: (●) 0.01, (○) 0.1, (□) 1.0

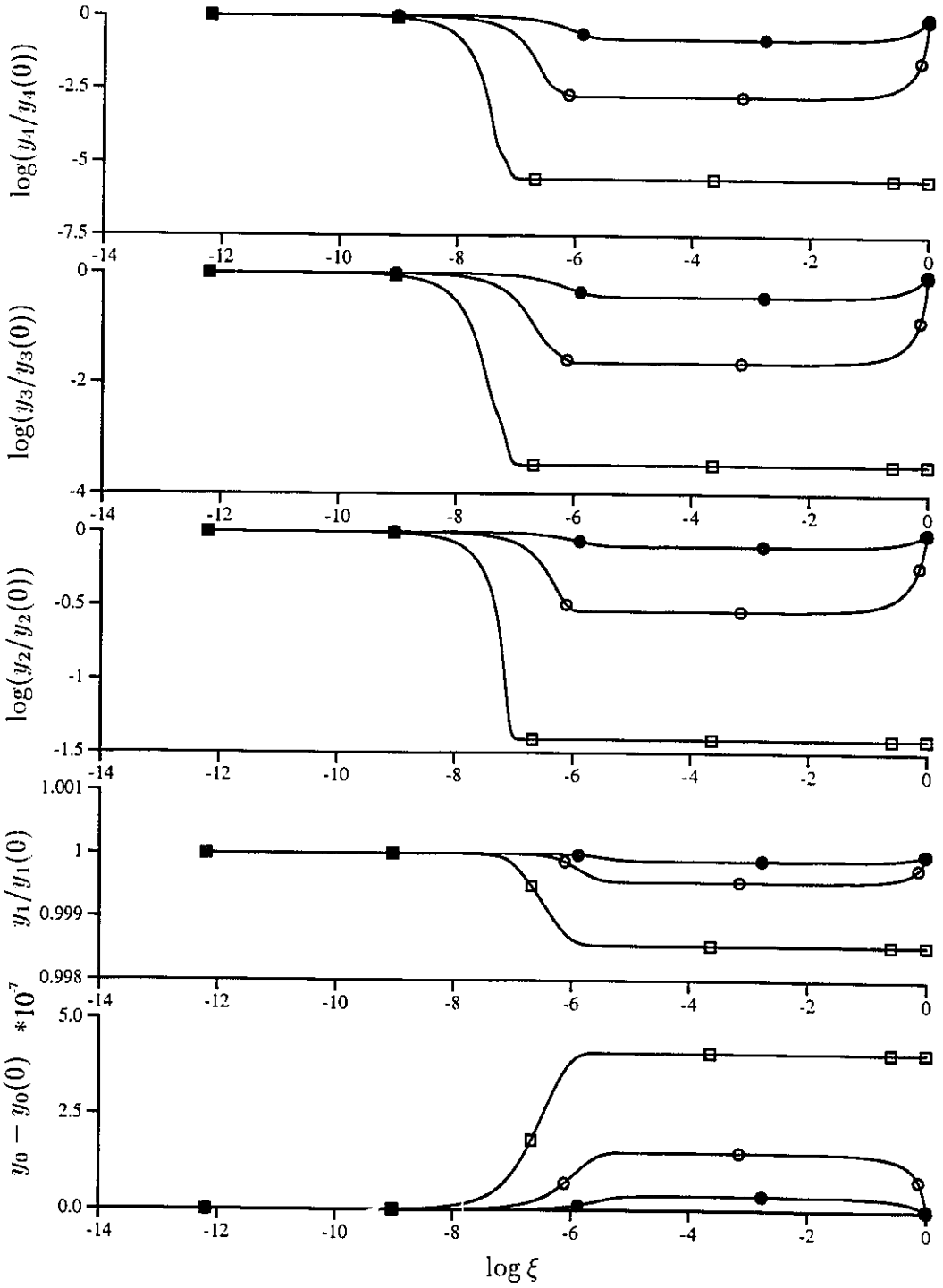

Figure B.17: Changes in y_v vs. $\log \xi$ for $v = 1$–4, R = HCl, $T = 500\,\mathrm{K}$ and various values of [Br]/[R]: (●) 0.01, (○) 0.1, (□) 1.0

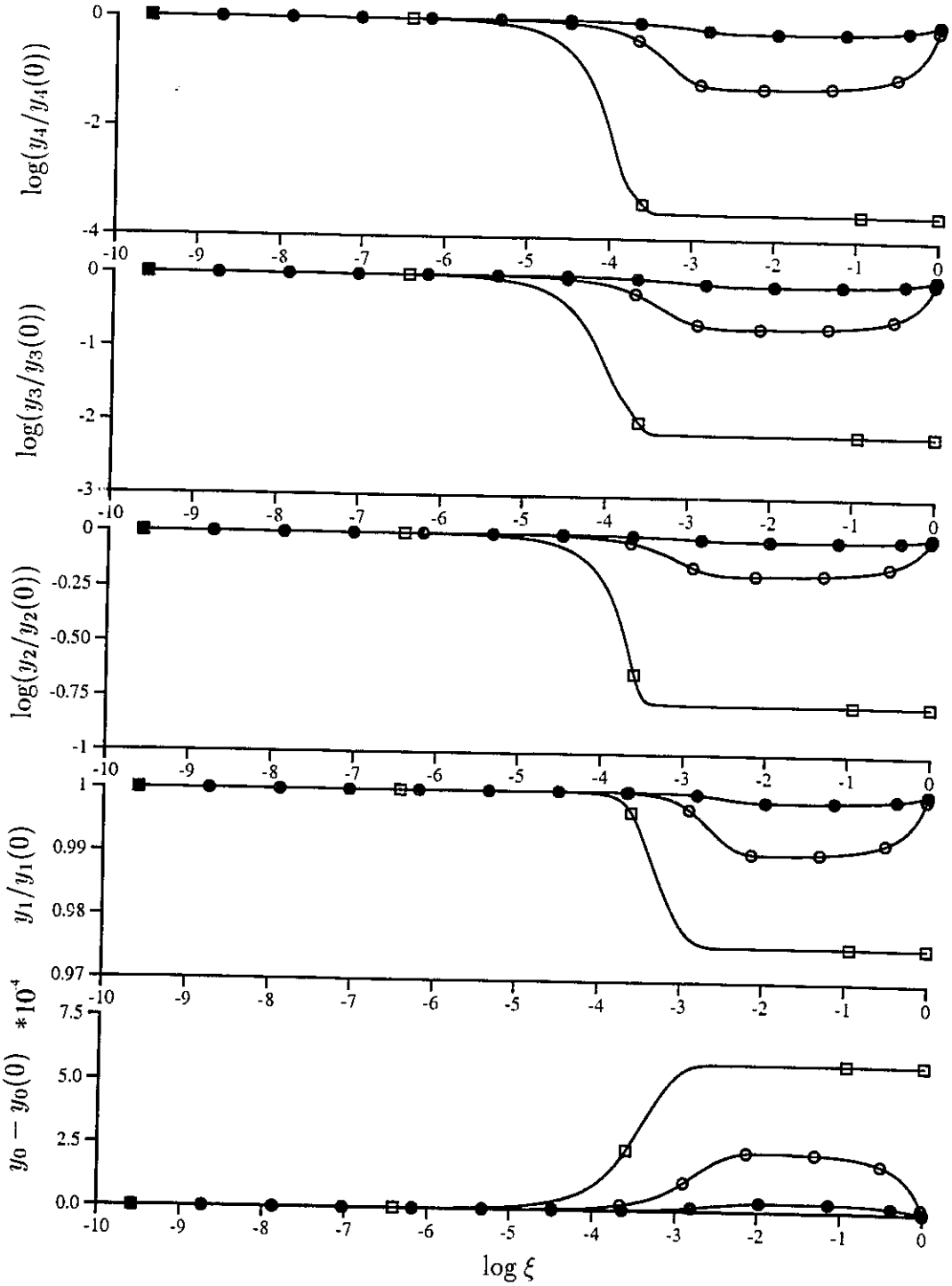

Figure B.18: Changes in y_v vs. $\log \xi$ for $v = 1$–4, R = HCl, $T = 1000\,\mathrm{K}$ and various values of [Br]/[R]: (●) 0.01, (○) 0.1, (□) 1.0

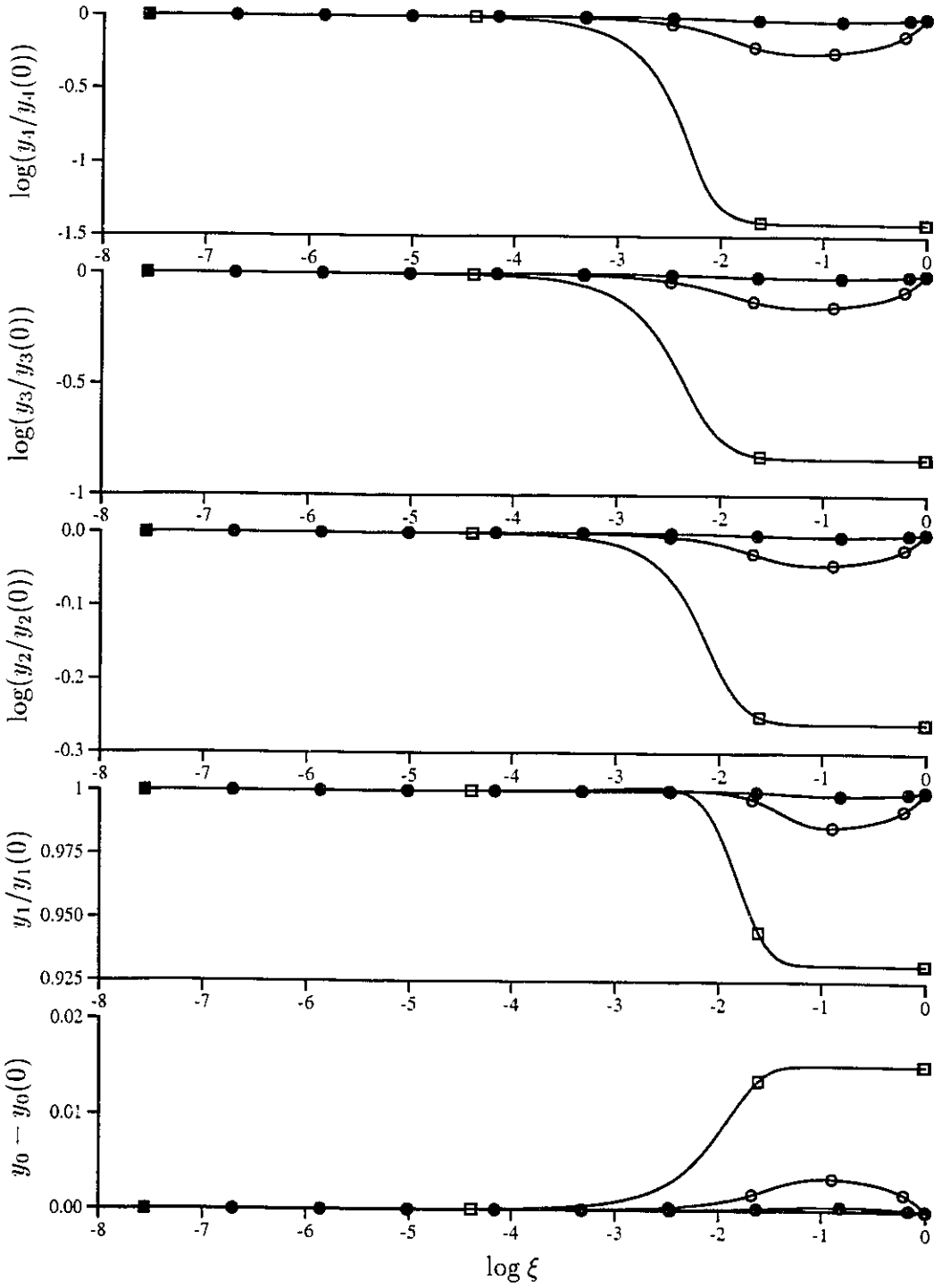

Figure B.19: Changes in y_v vs. $\log \xi$ for v = 1-4, R = HCl, T = 2000 K and various values of [Br]/[R]: (●) 0.01, (○) 0.1, (□) 1.0

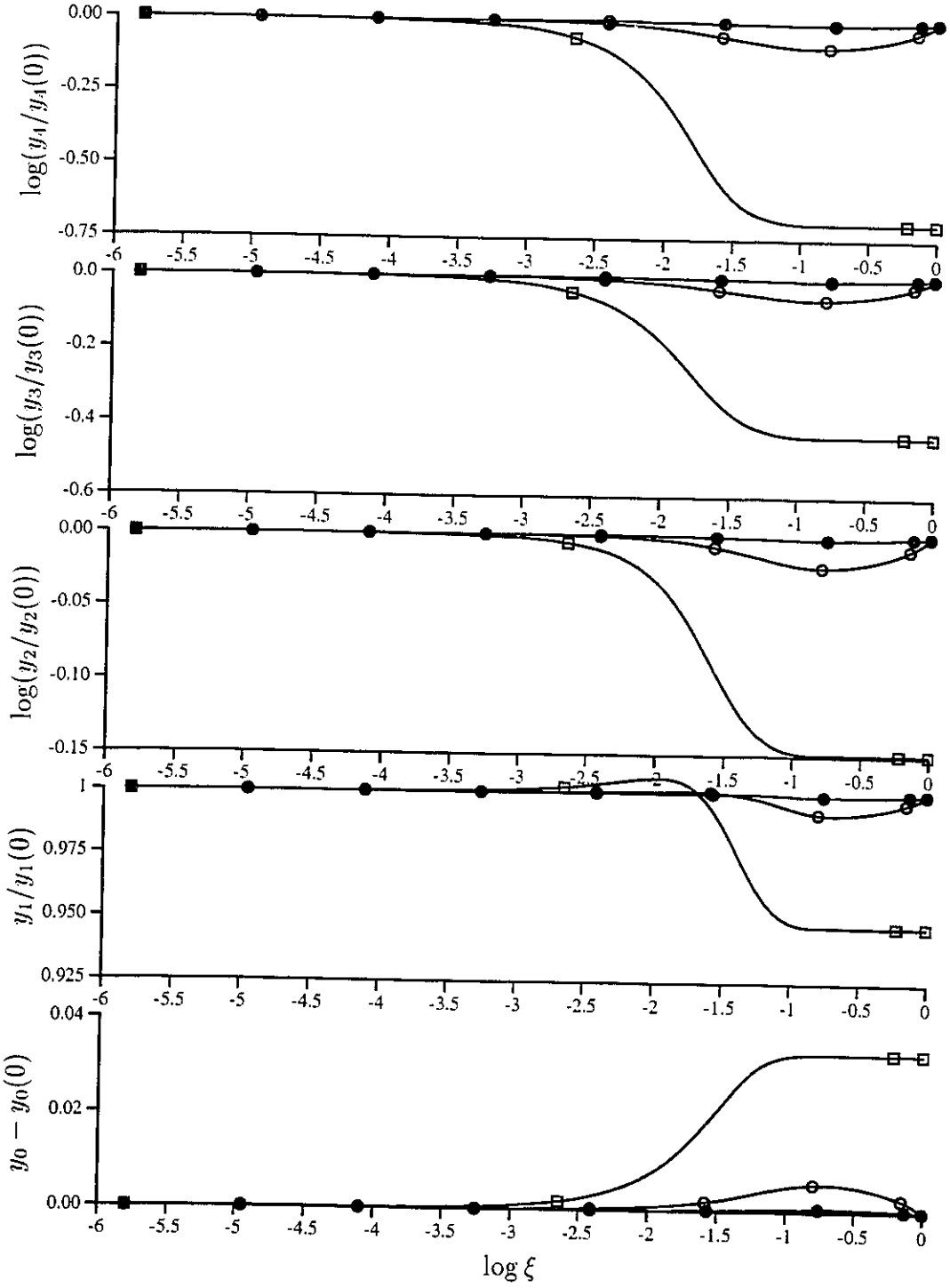

Figure B.20: Changes in y_v vs. $\log \xi$ for $v = 1$–4, R = HCl, $T = 3000\,\mathrm{K}$ and various values of [Br]/[R]: (●) 0.01, (○) 0.1, (□) 1.0

Appendix C. Source Code

The source code for BIMSIM is presented on the following pages. In order to conserve space and since the source is presented mainly for archival reasons a very small typeface and a double column layout is used. (A larger typeface would require single column layout and require four times as many pages.)

```
C This program simulates the kinetics of bimolecular reactions.

   Rotational levels included.

     Variables:
       The assumptions made for the allowed v, J ---> v', J' transitions
       are: a change of 0 or 1 or -1 in the vibrational level
       and a change of 0 or 2 or -2 in the rotational levels.

FUPUP  FUPI  FUP0 --
        These two-dimensional arrays contain the energy transfer rate
        constants in the 'up' direction.  The convention is as
        follows: FUPUP(v,J)  => the transition v,J -> v+1,J+2
                 FUPD(v,J)   => the transition v,J -> v+1,J-2
                 FUP0(v,J)   => the transition v,J -> v+1,J
FDOWNU  FDOWND  FDOWN0 --
         Similarly,these two-dimensional arrays contain the energy
         transfer rate constants in the 'down' direction.
        The convention is as follows:
                 FDOWNU(v,J) => the transition v,J  -> v-1,J+2
                 FUPD(v,J)   => the transition v,J -> v+1,J-2
                 FUP0(v,J)   => the transition v,J -> v+1,J
The J-->J-2 transition rate coefficients can be determined
 by microscopic reversibility (along with v-->v-1 rates).

The following relationships exist through microscopic reversibility:
                 FUPUP(v,J) <------> FDOWND(v,J)
                 FUPD(v,J)  <------> FDOWNU(v,J)
                 FUP0(v,J)  <------> FDOWN0(v,J)
                 FDUP(v,J)  <------> FDOWN(v,J)

  Note that these matrices occupy only a subset of the allocated
  space.  For example,
  KDOWND(0:10,0:20) actually only has values over (1:10,2:19);
  and FUPU(0:10,0:20) would only have values over (0:9,0:18).

  KROT -- When the vibrational level does not change, but
          the rotational level changes, to 2 or -2
  FVJBEI -- The bimolecular reaction rate coeffi . .t for
            each particular v,J level.
  DESLEL -- (in SOLVER) The desired "delta" or fractional decrement
            in population before a value is output for a given
            v level.

  IOUT -- an integer, holds the unit on to which prompts will go.  If
input is from the terminal then prompts go to the terminal.  If input
is coming from a file (including a file saved from a previous run)
then prompts are redirected to a dummy file and are not sent to the
terminal.

  Values are classified according to what kind of quantity they
  represent (energy, mass, temperature, etc.).
  To determine what 'kind' a value is, divide its IREAD code by 10.
  The general divisions are:
       (1) energy
       (2) rate constants
       (3) distance, inverse distance
       (4) mass
       (5) temperature
       (6) population

The reason that the number representing the type of conversion
needed must be integer-divided by 10
is that this number serves to store more than one
type of information: not only the kind of quantity (e.g., energy
rate constant, etc.) but the physical
meaning associated with that number for purposes of reading in
values in REAVJ and READIN.  For example there is more than one type
of rate cons . . . antity type 2) that is read in but each must have
the same units . . . r/on applied and therefore the code associated
with those value . . . is numbered 21, 22, 23, etc.
This has the drawba . . . that the program will have to be changed
substantially if, for example, there were more than 10 types of rate
constants between which we wished to distinguish.
The advantage of this approach is that the program has consistency
in the codes used by subroutines READIN, READVJ, SIU, NOVICE.

  This program converts units specified for the following
  types to SI units:

       (1) energy
            10 - 'generic' energy
            11 - E(v), disregarding J
            12 - E(v,J)
            13 - E(J), for a set v
       (2) rate constants
            20 - 'generic' rate constants
            21 - k(v), disregarding J
            22 - k(v,J)
            23 - k(J), for a set v
            24 - kbim(v), disregarding J
            25 - kbim(v,J)
       (3) distance, inverse distance
            30 - distance
            39 - inverse distance
       (4) mass
            40 - mass
       (5) temperature
            50 - temperature
       (6) population
            60 - 'generic' population
            61 - n(v), disregarding J
            62 - n(v,J)
            63 - n(J), for a set v

Conversion factors are based on W.H. Flygare(1978).

If data are being read in from particular files, the user must
 be sure to define the locations on which the files are located
according to the following assignments (from subroutine READIN):

 IREAD                      UNIT FOR READING IN FILE

 (from subroutine READIN):
     11 -- v, E(v,J=0)              1
     12 -- v, kvp(v,0)              2
     13 -- v, kbim(v,0)             3
     61 -- v, nv(v,0)               4
     63 -- J, E(v=0,J)              5

      PROGRAM BIMSIM
      IMPLICIT DOUBLE PRECISION (A-H,P,R-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-N,V)
      IMPLICIT CHARACTER*6 (S)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,IHEAT,INOV,VONLY,VUNIT(0:12),
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXJN,INPTS,ICOLL
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/SPCIES/ISPEC,SPEC(6),
      COMMON/POP/ANV(0:10,0:20),NTOT,CONC(3),JCOLL(3)
         ...  for concentrations CONCA,CONCBC,CONCM respectively

   IOUT and IDEBUG must have values when IPRSUP is called but
   their values
   are assigned proper in subroutine INIT.
     IDEBUG = 0
     IOUT   = 6
     ISUP=SPRSUB( BIMSIM )
C                          Change comment stmt with version number.
     PRINT*,   Version 0.64  of BIMSIM,  27 May    1990
     PRINT*, ' '
     CALL INIT
     CALL RXNSYS
     IF(INOV.EQ.1) CALL ASSUME
     CALL TMPTUR
C  The temperature should be known first, in case the user specifies
C  partial pressure in the CONC subroutine.
     CALL CONCN
     CALL PIPMET(ANMETH)
C         For each species in the reaction, the rate coefficients
C         must be found.

C  The vibrotational structure of diatomic BC is of interest,
C  but not that of anything such as the relaxant gas M which
C  collides with BC, and thus PIKRXN, where the potential energy
C  curve is specified, is not within the DO 100 loop.

     CALL PIKDUB(ANDBU)

C  Now that the ELEV array for BC has been calculated and filled in
C  from (0,0) to (maxv,maxJ), we can consider the effect of collisions
C  on BC.

     ICOLL=0
     DO 100 ISPEC=1,3
C                        ...the rate coefficients depend on what SPECies
C                           collide with BC:  A, BC (rel), or M.
 901   WRITE(IOUT,*)
       WRITE(IOUT,*)  ****************************************
       WRITE(IOUT,*)
       WRITE(IOUT,702)  (SSPEC(L),L=1, 6)
 702   FORMAT(1X,6A6)
       WRITE(IOUT,*) Type 1 to continue with these collision partners.
       WRITE(IOUT,*) (Otherwise, type another number.)  <Default = 1>
       READ(INP,*,END=902) ANCOLL
       GO TO 903
 902   IF(IOUT.EQ.ISUP) THEN
         INP=5
         IOUT=6
         GO TO 901
       ELSE
         CALL DFALT1(ANCOLL,1)
       ENDIF
 903   WRITE(ISAVE,*) ANCOLL
       IF(ANCOLL.NE.1) THEN
         GO TO 100
       ELSE
         ICOLL=ICOLL+1
         JCOLL(ICOLL)=ISPEC
C The values of the array JCOLL are set up according to the following
C collision partners with BC:
C         1  -  A + BC
C         2  -  BC + BC
C         3  -  relaxant M + BC
C    e.g.  if we only want to consider A + BC and M + BC cases
C          then JCOLL(1) = 1 and JCOLL(2) = 3.
C The JCOLL array is initialized to zero, so that the total number of
C collision types may be determined by finding the position of
C the last non-zero value.
       ENDIF

C Calculation of  FOUP vector for each v level...
C  where  FOUP(v,m) means k(v,m)-->k(v+2)
C   (only called if ro-vib. solution wanted)

       IF(VONLY.EQ.0) CALL PIKFOU(ANFOUP)
C The J-->J-2 transition rate coefficients can be determined
C  by microscopic reversibility (along with v-->v-1 rates).

       IF (ANMNU.NE.1) THEN
C                                ANMNU=1 means that F01 values are determined
         CALL PIKF01(ANF01)
       ELSE
         CALL F01SCL(1)
C                      .... arg. 1 means Landau-Teller scaling to be done
       ENDIF
       CALL CHECYR
C MICREV must be called before COLLCT!
       CALL MICREV
       CALL COLLCT
 100   CONTINUE
       CALL PIKRXN(ANRXN)
C  After the user has specified the method of obtaining the
C  microscopic rate coefficient in this subroutine, and the krxn(v,J)
C  are found in the various subroutines PIKRXN branches to,
C  then the krxn should be examined to see which values are
C  not reasonable for the system of interest.
C  For a bimolecular reaction, this would imply that those krxn
```

```
C  which exceed the collision frequency possible, according
C  to the temperature and pressure of the system.
C  Thus, the subroutine DIFFUS is called.
C  In this subroutine  the collision frequency is calculated
C  and any RKN(V,J) which exceeds it is set equal to it.
C  If the user is modelling a unimolecular reaction, or an ideal
C  reaction which is not limited by the rate of diffusion, then
C  'no upper limit' in the subroutine DIFFUS should be specified.
      IF(ANMETH.NE.3) THEN
        CALL PIPPOP(ANPOP)
        CALL DIFFUS
        CALL SOLVER
      ENDIF
      IF(ANMETH.NE.1) THEN
        CALL FINIT(ICTR1)
        CALL EIGEN (ICTR1)
      ENDIF
 907  WRITE(IOUT *) Type 1 if you want tabulated output, <default: 1>
      READ(INP * END=905) ANTAB
      GO TO 906
 905  IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 907
      ELSE
        CALL DFALTI(ANTAB,1)
      ENDIF
 906  WRITE(ISAVE *) ANTAB
      IF (ANTAB.EQ.1) CALL TABOUT
 908  WRITE(IOUT,*) Type 1 if you want output plotted <Def.: 1>
      READ(INP * END=909) ANPLOT
      GO TO 910
 909  IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 908
      ELSE
        CALL DFALTI(ANPLOT,0)
      ENDIF
 910  WRITE(ISAVE *) ANPLOT
      IF (ANPLOT.EQ.1) CALL PLTOUT
      STOP 1006
      END
```

```
      SUBROUTINE INIT
C This subprogram initializes the physical constants used.
C  The constants are given for SI units.
C  (specify more significant figures for these constants)
C& No Document default changeability.
      IMPLICIT DOUBLE PRECISION (B-H,P,R-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,O)
      IMPLICIT CHARACTER*6 ($)
      REAL*8 APAD,AMASS
      DOUBLE PRECISION AVOGAD
      CHARACTER*36 $SPGEN(3)
      COMMON/SPGEN/$SPGEN
      COMMON/PHYSCT/PLANCK,KBOLTZ,ROAS,SCEXPI,CLIGHT,PI,I4EVER
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INEAT,INEV,VUNIT(5:23)
      COMMON//COUNT/MAXV,MAXJ(0:19),MAXUN,INPTC,ICOLL
      COMMON/POP/NV(0:10,0:20),NTOT,CONCA,CONCBC,CONCM,JCOLL(3)
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/MASS/EMASS,CMASS,AMASS,REDMAS,BRAD,CRAD,APAD,RE
      PARAMETER (AVOGAD=6.022045D23)
      EQUIVALENCE(SPGEN(1),SPEC(1,1))
C  The long strings which are specific for the various kinds of
C  collision are written into a *36 character string  which is
C  then effectively broken into 6 - *6 character strings by means
C  of equivalencing with an array $SPEC of dimension 6.
C  The string-length is chosen because CHARACTER*6 is conventionally
C  used in other subroutines.

C  ERRSET is called because underflow errors may occur in the program.
C  ERRSET is an IBM system program.  Other adaptations of this
C  program to different systems will require a call to  a different
C  underflow-adjustment system routine.  If no such subroutine is
C  available  the user may have to modify the program.

      CALL ERRSET (208  0  -1  1  1)
      MAXUN = 11
      INP   = 5
      IDEBUG= 1
C                ...debug toggled  on   for the time being
      IOUT  = 6
C           IOUT and IDEBUG must be set  for $PRSUB to work
      $SUB=$PRSUB( INIT   )

C  If this program has been run once  the answers will be saved
C  on unit ISAVE so that a restart option is available.
C  To avoid meaningless prompting on the terminal,  the
C  output will be suppressed by writing it to unit ISUP.
      ISUP=59
      ISAVE=98
      IFLAG=0
      WRITE(IOUT,*)   Defaults are explicitly noted, or are
     &   indicated by a "*" on the menus.
      WRITE(IOUT,*)   Defaults may be obtained by entering only a
     &   carriage return after the prompt.
      WRITE(IOUT,*)
 904  WRITE(IOUT,*)  Is this a restart? (*Yes = 1  No = 0.)
      READ(INP,*,END=911) ANRES
      GO TO 912
 911  CALL DFALTI(ANRES,1)
 912  IF(ANRES.LT.0.OR.ANRES.GT.1) THEN
        WRITE(IOUT,*)  Error in restart specification.  ANRES
     &                              Choose 0 or 1
        IFLAG=IFLAG+1
        IF(IFLAG.LE.1) GO TO 904
        CALL ISTERM(ANRES,$SUB)
      ENDIF
      IF (ANRES.EQ.1) THEN
        INP=9
C                 ...to avoid colliding with filedefs for v-levels
        IOUT=ISUP
      ENDIF
      PLANCK=6.63cD-34
      KBOLTZ=1.3805D-23
C  R obtained from Rev. Mod. Phys. vol. 27, p. 363 (1955).
      ROAS=KBOLTZ * AVOGAD
      CLIGHT=3.0D+8
      I4EVER=2000000000
      SCEXPI=1.0D0
      PI=3.14159265358979D0
      EMASS=0.0D0
      CMASS=0.0D0
      AMASS=0.0D0
      REDMAS=0.0D0
      BRAD=0.0D0
      CRAD=0.0D0
      APAD=0.0D0
      RE=0.0D0
C  The maximum number of output units is set by MAXUN.  For
C  v-levels ranging from 0 to 10  this means 11 output files
C  each of which will contain population information  for
C  appropriate J sub-levels. The rate-of-population-change
C  information is contained on 11 more output files.
C  2 additional files will monitor total population
C  and total rate-of-population-change.
C                                        unit                VUNIT index
C        NV data for V=0....10          units 10....20        5--15
C        NV data summed (NV,T)                21              16
C        DNV data for v=0...11          units 40....50        20--31
C        DNDT  data                     unit 51               32
C
      MAXV=0
      DO 20 V=0,10
        MAXJ(V)=0
        VUNIT(V)=10+V
        VUNIT(V+12)=40+V
 20   CONTINUE
C        The matrix ELEV must have one extra level  so that the
C        energy difference can be calculated in EDEL for a given level.
      DO 10 V = 0,10
        DO 10 J= 0,20
          ELEV(V,J)=0.0
 10   CONTINUE
C NV initialized  value unimportant  normalized later.
```

```
      INV(10 0) = 1.0

C
C     NOTES ON DEFAULTS:
C
C  The default value is either explicitly denoted as such
C  or it is indicated with a  *  in front of the menu number.
C
C  If the input is from the terminal, a carriage return will
C  signify that the default value is desired.
C  If the input is from a file  then reset the input unit
C  to that of the terminal, so that the user can pick up
C  from where s/he left off...
C
C  The following default subroutines are called:
C        DFALTD - for double precision numbers
C        DFALTI - for integer numbers
C        DFALTC - for character input.
C  If the user wishes to change the default value, only the
C  prompting default reminder, and the second argument in the
C  DFALTn call must be changed.
C
900   WRITE(IOUT,*)'  Type a  1  if prompting for units is
     &  ' required. Any other number--no prompting. <Default: 0>'
C           (The routine for prompting for units is NOVICE.)'
      READ(INP,*,END=901) INOV
      GO TO 902
901   IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 900
      ELSE
        CALL DFALTI(INOV,0)
      ENDIF
902   WRITE(ISAVE,*) INOV
      JCOLL(1)=0
      JCOLL(2)=0
      JCOLL(3)=0
      SSPGEN(1)='  For collisions between A and BC: '
      SSPGEN(2)=' For collisions between BC and BC: '
      SSPGEN(3)=' For collisions between M and BC: '
      IF(INOV.NE.1) RETURN
      WRITE(IOUT,*)' This program accepts many kinds of units. '
      WRITE(IOUT,*)' The user must know which two-letter codes refer'
      WRITE(IOUT,*)' to which units, e.g.  EV means electron volts.'
      WRITE(IOUT,*)' The letter code for inverse units is identical to'
      WRITE(IOUT,*)' that of the non-inverted units.  The user must'
      WRITE(IOUT,*)'  input values and units  with the units enclosed
     1   ' in single quotes.'
     2   ' and separated by commas, e.g., "4.3, cm  ."'
      WRITE(IOUT,*)
      RETURN
      END
```

```
      SUBROUTINE RXNSYS
      IMPLICIT CHARACTER*6 (S)
      IMPLICIT DOUBLE PRECISION (A-H,P,N-Q,W-Z)
      IMPLICIT INTEGER (I-J,L-M,V)
      CHARACTER*2 UNIT1,UNIT2
      COMMON/IO/INP,IOUT,ISAVE,ISUP,ITEBUG,INDAT,INOV,VONLY,VUNIT*6,2A,
      COMMON/MASS/BMASS,CMASS,AMASS,REDMAS,BRAD,CRAD,ARAD,PE
      DIMENSION XMASS(106), XRAD(106),IMASS(3),IRAD(3),IZ(3)
      EQUIVALENCE (IMASS(1),BMASS)  ,(IRAD(1),BRAD)
      EQUIVALENCE (IZ(1),IZB) ,(IZ(2),IZC) ,(IZ(3),IZA)
C The following is an admittedly schizophrenic combination
C  of the weights of the most prevalent isotopes (from
C  W.H. Flygare  1978) and the averaged atomic weight
C  (from table from Chemical Services  Box 45124
C  Los Angeles, Ca. 90045).  The s- and p-shell atoms
C  have the "most prevalent isotope" weights; the d- and
C  f-shell species are the averaged values.
C  Values are in gm/mole.
C
      DATA (XMASS(L),L=1,50)/1.00783,4.00260,7.01601,9.01219,11.00931,
     1     12.00000,14.00307,15.99491,18.99840,19.99244,
     2     22.98977,23.98504,26.98153,27.97693,30.97376,
     3     31.97207,34.96885,39.96238,38.96371,39.96259,
     4     44.9559 ,47.90  ,50.9415 ,51.996  ,54.9380 ,
     5     55.847  ,58.9332 ,58.70   ,63.546  ,65.38   ,
     6     68.9257 ,73.9219 ,74.9216 ,78.9163 ,78.9183 ,
     7     84.9115 ,84.9117 ,87.9056 ,88.9059 ,91.22   ,
     8     92.9064 ,95.94   ,97.0    ,101.07  ,102.9055,
     9     106.4   ,107.868 ,112.41  ,114.9039 ,117.9018 /
      DATA  (XMASS(L),L=51,106) /
     1     120.9038,129.9062,126.90447,131.9042,132.9054,
     2     137.9055,138.9055,140.12   ,140.9077,144.24  ,
     3     145.0   ,150.4   ,151.96   ,157.25  ,158.9254,
     4     162.50  ,164.9304,167.26   ,168.9342,173.04  ,
     5     174.967 ,178.49  ,180.9479 ,183.85  ,186.207 ,
     6     190.2   ,192.22  ,195.09   ,196.9665,200.59  ,
     7     204.9745,207.9766,208.9804 ,209.0   ,210.0   ,
     8     222.0   ,223.0   ,226.0254 ,227.0278,232.0381,
     9     231.0359,238.029 ,237.0482 ,244.0   ,243.0   ,
     1     247.0   ,247.0   ,251.0    ,254.0   ,257.0   ,
     2     258.0   ,259.0   ,260.0    ,261.0   ,262.0   ,
     3     263.0  /
C  Values of atomic radii obtained from Chemical Principles
C  by Masterton and Slowinski (W.B.Saunders  1969).
C  Interpolated values for Tc  Pm  Sm  At  Fr and Pa.
C  Values extend only up to U (element 92).
C   Values are in angstrom.
      DATA (XRAD(L),L=1,50)/0.37    ,0.93    ,1.52    ,1.11    ,0.88    ,
     1     0.77    ,0.70    ,0.66    ,0.64    ,1.12    ,
     2     1.86    ,1.60    ,1.43    ,1.17    ,1.10    ,
     3     1.04    ,0.99    ,1.54    ,2.31    ,1.97    ,
     4     1.60    ,1.40    ,1.31    ,1.25    ,1.29    ,
     5     1.26    ,1.25    ,1.24    ,1.28    ,1.33    ,
     6     1.22    ,1.22    ,1.22    ,1.17    ,1.14    ,
     7     1.69    ,2.44    ,2.15    ,1.60    ,1.57    ,
     8     1.43    ,1.36    ,1.34    ,1.33    ,1.34    ,
     9     1.38    ,1.44    ,1.49    ,1.62    ,1.40   /
      DATA  (XRAD(L),L=51,92) /
     1     1.41    ,1.37    ,1.33    ,1.90    ,2.62    ,
     2     2.17    ,1.87    ,1.82    ,1.82    ,1.82    ,
     3     1.9     ,1.9     ,2.04    ,1.79    ,1.97    ,
     4     1.77    ,1.76    ,1.75    ,1.74    ,1.93    ,
     5     1.74    ,1.67    ,1.43    ,1.37    ,1.37    ,
     6     1.34    ,1.35    ,1.38    ,1.44    ,1.55    ,
     7     1.71    ,1.75    ,1.46    ,1.65    ,2.0     ,
     8     2.1     ,2.2     ,2.2     ,2.6     ,1.8     ,
     9     1.6     ,1.4    /
C Values of atomic radii for atomic number > 92
C unavailable to us at the present time.
      DATA  (XRAD(L),L=93,106) /
     1     3.,     3.,     3.,     3.,     3.,
     2     3.,     3.,     3.,     3.,     3.,
     3     3.,     3.,     3.,     3.    /
      IOUR=SPRAUD('RXNSYS',
900   WRITE(IOUT,*)  ' This program simulates the reaction system
      WRITE(IOUT,*)  '      BC + A --> Products'
      WRITE(IOUT,*)  ' Input the atomic numbers for B  C and A in that
     4   ' order.'
      WRITE(IOUT,*) '<Default:  1,17,35 (HCl+Br)>'
      WRITE(IOUT,*) ' (Type 0,0,0 if you wish to use the built-in
     1   ' periodic table.)'
      READ(INP,*,END=901)IZB  IZC  IZA
      GO TO 902
901   IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 900
      ELSE
        CALL DFALTI(IZB,1)
        CALL DFALTI(IZC,17)
        CALL DFALTI(IZA, 35)
      ENDIF
902   WRITE(ISAVE,*) IZB, IZC  IZA
C Check to see whether the atomic numbers correspond to
C to a range available to known elements.
      IF(IZB .LT. 1 .OR. IZB .GT. 106 .OR.
     &   IZC .LT. 1 .OR. IZC .GT. 106 .OR.
     &   IZA .LT. 1 .OR. IZA .GT. 106) THEN
        WRITE(IOUT,*)  ' Atomic numbers range from 1 to 106.'
        WRITE(IOUT,*)  ' Choose from the following chart according
     &   ' to atomic number.'
        CALL CHART(IZB  IZC, IZA)
      ENDIF
      WRITE(IOUT,*)  ' These are the masses and covalent radii
      WRITE(IOUT,*)  ' corresponding to the atomic numbers chosen:'
      WRITE(IOUT,*)' Atomic Number      Mass                    Radius'
      WRITE(IOUT,*)'                   (g/mole)                (angst.)'
      WRITE(IOUT,*) IZB, XMASS(IZB),XRAD(IZB)
      WRITE(IOUT,*) IZC, XMASS(IZC),XRAD(IZC)
      WRITE(IOUT,*) IZA, XMASS(IZA),XRAD(IZA)
      IFLAG=0
904   WRITE(IOUT,*)  'Do you wish to accept these values?'
```

```
     &                    (*Yes = 1, No = 0):
      READ(INP,*,END=905) IANACC
      GO TO 906
905   IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 904
      ELSE
        CALL DFALTI(IANACC,1)
      ENDIF
906   WRITE(ISAVE,*) IANACC
      IF (IANACC.EQ.0) THEN
C     prompt for new values, a mass/radius pair at a time
        DO 100 I=1,3
907       WRITE(IOUT,*) ' What is the mass and radius of element no.
     &              ,IZ(I),' including units?
          WRITE(IOUT,*)    <Default:   XMASS(IZ(I)),'  GM  '
     &                  XRAD(IZ(I)),'  AG   .>
          IF(I.EQ.1.AND.INOV.EQ.1) THEN
C             only call novice once in the loop (overkill otherwise)
            CALL NOVICE(40)
            CALL NOVICE(33)
          ENDIF
          READ(INP,*,END=908) XMASS(I),UNIT1,XRAD(I),UNIT2
          GO TO 909
908       IF(IOUT.EQ.ISUP) THEN
            INP=5
            IOUT=6
            GO TO 907
          ELSE
            CALL DFALTD(XMASS(I),XMASS(IZ(I)))
            CALL DFALTC(UNIT1,'GM')
            CALL DFALTD(XRAD(I),XRAD(IZ(I)))
            CALL DFALTC(UNIT2,'AG')
          ENDIF
909       WRITE(ISAVE,*) XMASS(I),UNIT1,XRAD(I),UNIT2
100     CONTINUE
      ELSEIF(IANACC.EQ.1) THEN
          XMASS=XMASS(IZR)
          CMASS=XMASS(IZC)
          AMASS=XMASS(IZA)
          BRAD=XRAD(IZR)
          CRAD=XRAD(IZC)
          ARAD=XRAD(IZA)
          UNIT1='GM'
          UNIT2='AG'
      ELSE
          WRITE(IOUT,*)  Error in attept specification:  IANACC.
     &                          Choose 0 or 1
          IFLAG=IFLAG+1
          IF(IFLAG.LE.1) GO TO 906
          CALL IBTERM(IANACC,$SUB)
      ENDIF
      DO 200 I=1,3
        CALL SIU(45,XMASS(I),UNIT1)
        CALL SIU(50,XRAD(I),UNIT2)
200   CONTINUE
      REDMAS=(XMASS*CMASS)/(XMASS + CMASS)
      RETURN
      END
```

```
      SUBROUTINE ASSUME
      IMPLICIT CHARACTER*6 ($)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      $SUB=$PRSUB('ASSUME')
      WRITE(IOUT,*)  This program is based on a number of assumptions
      WRITE(IOUT,*)  namely, v changes by +1, 0 or -1, and
      WRITE(IOUT,*)  J changes by +2, 0 or -2.  No reverse reaction is
      WRITE(IOUT,*)  assumed to occur.  The temperature is constant
      WRITE(IOUT,*) during the course of the reaction. The microscopic
      WRITE(IOUT,*)  rate constant is assumed to be independent of the
      WRITE(IOUT,*) final product state. No V-V or P-R energy transfer
      WRITE(IOUT,*) is assumed to occur.
      RETURN
      END
```

```
      SUBROUTINE TMPTUR
      IMPLICIT DOUBLE PRECISION(A-H,J-Z)
      IMPLICIT INTEGER (I)
      IMPLICIT CHARACTER*6 ($)
      CHARACTER*3 UNIT1
      INTEGER VUNIT, VONLY
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      $SUB=$PRSUB('TMPTUR')
115   WRITE(IOUT,*)    Input the temperature. <Default: 300   K  >
      IF(INOV.EQ.1) CALL NOVICE(50)
      READ(INP,*,END=903)   TEMP,UNIT1
      GO TO 904
903   IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 115
      ELSE
        CALL DFALTD(TEMP,300.0D0)
        CALL DFALTC(UNIT1,'K')
      ENDIF
904   WRITE(ISAVE,*) TEMP,UNIT1
      CALL SIU(50,TEMP,UNIT1)
      RETURN
      END
```

```
      SUBROUTINE CONCN
      IMPLICIT DOUBLE PRECISION(A-H,J-Z)
      IMPLICIT INTEGER (I)
      IMPLICIT CHARACTER*6 ($)
      CHARACTER*3 UNIT1
      INTEGER VUNIT, VONLY, JCOLL
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/POP/NV(0:10,0:20),NTOT,CONCA,CONCBC,CONCM,JCOLL(3)
      $SUB=$PRSUB('CONCN')
110   WRITE(IOUT,*)  ' What is the unit of concentration used?
      WRITE(IOUT,*)    <Default:   MO   (molar)>
      IF(INOV.EQ.1) CALL NOVICE(86)
      READ(INP,*,END=900)   UNIT1
      GO TO 901
900   IF(IOUT.EQ.ISUP)  THEN
        INP=5
        IOUT=6
        GO TO 110
      ELSE
        CALL DFALTC(UNIT1,'MO')
      ENDIF
901   WRITE(ISAVE,*) UNIT1
C       Dummy call to SIU, just to check if units for concentration
C       are kosher.
      CALL SIU(60,1.0D0,UNIT1)
111   WRITE(IOUT,*)    What is the concentration of species A?
      WRITE(IOUT,*)    <Default: 1.0D-6>
      READ(INP,*,END=902)   CONCA
      GO TO 903
902   IF(IOUT.EQ.ISUP)  THEN
        INP=5
        IOUT=6
!!!!! what happened to IFLAG...?
        GO TO 111
      ELSE
        CALL DFALTD(CONCA,1.0D-6)
      ENDIF
903   WRITE(ISAVE,*) CONCA
      CALL SIU(60,CONCA,UNIT1)
112   WRITE(IOUT,*)    What is the concentration of species BC?
      WRITE(IOUT,*)    <Default: 1.0D-3>
      READ(INP,*,END=904)   CONCBC
      GO TO 905
904   IF(IOUT.EQ.ISUP)  THEN
        INP=5
        IOUT=6
        GO TO 112
      ELSE
        CALL DFALTD(CONCBC,1.0D-3)
      ENDIF
905   WRITE(ISAVE,*) CONCBC
      CALL SIU(60,CONCBC,UNIT1)
113   WRITE(IOUT,*)    What is the concentration of relaxant gas?
      WRITE(IOUT,*)    <Default: 1.0D-1>
      READ(INP,*,END=906)   CONCM
      GO TO 907
906   IF(IOUT.EQ.ISUP)  THEN
        INP=5
        IOUT=6
        GO TO 113
      ELSE
        CALL DFALTD(CONCM,1.0D-1)
      ENDIF
907   WRITE(ISAVE,*)   CONCM
      CALL SIU(60,CONCM,UNIT1)
C  The three gases present in the default concentrations give a
C  combined pressure of about 40 torr.
      RETURN
      END
```

```
      SUBROUTINE PIKMET(ANMETH)
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,ICEBUG,INDAT,INCV,VONLY,VUNIT(0:23)
      SSUB=SPRSUB('PIKMET')
C  The structure of the IF and GO TO is meant to simulate
C  a DOWHILE loop (do while IFLAG is less than 2).
C
      IFLAG=0
  100 WRITE(IOUT,*)  ' Indicate the desired method of solution:'
      WRITE(IOUT,*)  ' (Vibrational/Rotational Solution):'
      WRITE(IOUT,*)  '      1) time integration.'
      WRITE(IOUT,*)  '      2) eigenvalue solution (steady state
     &                          approximation);'
      WRITE(IOUT,*)  '      3) both.'
      WRITE(IOUT,*)  ' (Vibrational Solution Only):'
      WRITE(IOUT,*)  '      4) time integration.'
      WRITE(IOUT,*)  '      5) eigenvalue solution (steady state
     &                          approximation);'
      WRITE(IOUT,*)  '      *6) Both.'
      READ(INP,*,END=900) ANMETH
      GO TO 901
  900 IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 100
      ELSE
        CALL DFALTI(ANMETH,6)
      ENDIF
  901 WRITE(ISAVE,*) ANMETH
      IF(ANMETH.LT.1.OR.ANMETH.GT.6) THEN
        WRITE(IOUT,*)' Error in method specification.  ANMETH
     &                              Choose 1, 2, 3, 4, 5, or 6.'
        IFLAG=IFLAG+1
        IF(IFLAG.LE.1) GO TO 100
        CALL IBTERM(ANMETH,SSUB)
      ENDIF
C  Check to see whether both vib/rotational solutions are desired
C  or only vibrational solutions.  If only the latter are desired  a
C  flag is set so that the user does not receive prompts relating
C  to the rotational levels.  (VONLY = 1 means vibrational only.)
      VONLY=0
      IF(ANMETH.GE.4.AND.ANMETH.LE.6) THEN
        ANMETH=ANMETH-3
        VONLY=1
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE PIKNNU(ANNNU)
C  Each subroutine for modelling the potential curve might find
C  rotational levels implicitly, or it will call a
C  further subroutine PIKROT, for determining the rotational
C  energy levels (unless
C  the user has specified the curve is obtained from a file of E(v,J)
C  being read in, or spectroscopically
C  so there is no need to determine how the rotational levels
C  are obtained  since this is implicit in the spectroscopic eqn.
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,ICEBUG,INDAT,INCV,VONLY,VUNIT(0:23)
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      SSUB=SPRSUB('PIKNNU')
      IFLAG = 0
  102 WRITE(IOUT,*)'The potential curve is modelled on:'
      WRITE(IOUT,*)                    '1) Simple Harmonic Oscillator'
      WRITE(IOUT,*)                    '2) 3-parameter Morse function.'
      WRITE(IOUT,*)                    '3) File of v, E(v) pairs.'
      WRITE(IOUT,*)                    '4) Partial spectroscopic fit
     &   --we, wexe, etc.'
      IF(VONLY.EQ.0) THEN
        WRITE(IOUT,*)'                      (Vibrational constants only)'
        WRITE(IOUT,*)'                   5) Spectroscopic fit'
     &  '   --we, wexe, etc., alphae, Be, etc.'
        WRITE(IOUT,*)'                   6) File of v, J, E(v,J) pairs'
      ENDIF
      READ(INP,*,END=900) ANNNU
      GO TO 901
  900 IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 102
      ELSE
        CALL DFALTI(ANNNU,1)
      ENDIF
  901 WRITE(ISAVE,*) ANNNU
      IF (ANNNU.EQ.1) THEN
        CALL SHO
C  Once MAXV is found in SHO, the user selects the method of
C  determining the rotational levels.
        IF(VONLY.EQ.0) CALL PIKROT
      ELSEIF (ANNNU.EQ.2) THEN
        CALL MORSE
        IF(VONLY.EQ.0) CALL PIKPOT
      ELSEIF (ANNNU.EQ.3) THEN
        CALL READIN(11,ELEV)
        IF(VONLY.EQ.0) CALL PIKROT
      ELSEIF (ANNNU.EQ.4) THEN
C The argument of SPEC is a flag, to signify rotational constants
C  will also be used---hence, the appropriate spec. eqn. must be
C  used in the subroutine SPEC.
C  However, first we must find out the value of MAXJ.
        IF(VONLY.EQ.0) CALL JMAX
        CALL SPEC(0)
        IF(VONLY.EQ.0) CALL PIKROT
      ELSEIF (ANNNU.EQ.5.AND.VONLY.EQ.0) THEN
        CALL SPEC(2)
      ELSEIF (ANNNU.EQ.6.AND.VONLY.EQ.0) THEN
        CALL READVJ(12,ELEV)
      ELSE
        WRITE(IOUT,*)' Error in potential curve specification.  ANNNU
        IF(VONLY.EQ.0) THEN
          WRITE(IOUT,*)  Specify 1, 2, 3, 4, 5 or 6.'
        ELSE
          WRITE(IOUT,*)  Specify 1, 2, 3 or 4.
        ENDIF
        IFLAG = IFLAG + 1
        IF (IFLAG.LE.1) GO TO 102
        CALL IBTERM(ANNNU,SSUB)
      ENDIF
      CALL PRNTVJ(12,ELEV)
C                          ....to print out ELEV
      RETURN
      END
```

```
      SUBROUTINE SHO
C     Developed for the model of a truncated harmonic oscillator.
C     SHO finds the (v,0) row of the ELEV array for v=0 to maxv,
C     the latter which is also determined in this subroutine.
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      CHARACTER*3 UNIT1, UNIT2
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:33)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN, INPTS, ICOLL
      JSUB=$PRSUB('SHO   ')
C  We have to know MAXV before we find the MAXJ of each v-level.
      IFLAG=0
  101 WRITE(IOUT,*) ' What is the vibrational spacing '
     &              'and its units? <Default: 2991  /cm  > '
C                                      .... for HCl
      IF(INOV.EQ.1) CALL NOVICE(10)
      READ(INP,*,END=901) EDEL0,UNIT1
      GO TO 902
  901 IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 101
      ELSE
        CALL DFALTD(EDEL0,2991.0D0)
        CALL DFALTC(UNIT1,'CM')
      ENDIF
  902 WRITE(ISAVE,*) EDEL0,UNIT1
      CALL SIU(10,EDEL0,UNIT1)
  903 WRITE(IOUT,*) ' What is the dissociation energy '
     &              'and its units? <Default: 4.62  eV  > '
      IF(INOV.EQ.1) CALL NOVICE(10)
      READ(INP,*,END=904)      DE,UNIT2
      GO TO 905
  904 IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 903
      ELSE
        CALL DFALTD(DE,4.62D0)
        CALL DFALTC(UNIT2,'EV')
      ENDIF
  905 WRITE(ISAVE,*) DE,UNIT2
      CALL SIU(10,DE,UNIT2)
      MAXV=   (DE/EDEL0) - 1/2
C                  ....this is based on the SHO equation we(v+1/2)=De.
      WRITE(IOUT,701)DE, EDEL0, MAXV
  701 FORMAT(' A De of ',F10.2,  ' jo/mole and a vibrational level'/,
     &       ' spacing of ',F9.2,' jo/mole corresponds to ',
     &       'v-levels from 0 to ',I3, '.')
      IF(MAXV.GT.(MAXUN-1)) THEN
        WRITE(IOUT,702) MAXUN, MAXUN-1
  702   FORMAT(' There are more than ',I3,' vibrational
     &         'levels, so vmax is being set to ',I3, '.')
        MAXV=MAXUN-1
      ENDIF
      DO 100 V=0,MAXV
        ELEV(V,0)=DBLE(V)*EDEL0
  100 CONTINUE
  105 WRITE(IOUT,703) MAXUN-1
  703 FORMAT(' Respecify v-gap, De--type -1 /.  accept MAXV value
     &'--type 0. /.  new MAXV value--specify (between 1 and ',I3,')
     &/.  corresponding to the highest v-level you want. <Default:3> ')
      READ(INP,* END=906) ANMAXV
      GO TO 907
  906 IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 105
      ELSE
        CALL DFALTI(ANMAXV,3)
      ENDIF
  907 WRITE(ISAVE,*) ANMAXV
      IF (ANMAXV .EQ. -1) GOTO 101
      IF (ANMAXV .NE. 0) THEN
         IF (ANMAXV.LT.0 .OR. ANMAXV.GT.MAXUN) THEN
           WRITE(IOUT,*) Input error on maximum v-level  ANMAXV
           IFLAG = IFLAG + 1
           IF (IFLAG.LE.1) GOTO 105
           STOP 1000
         ENDIF
         MAXV = ANMAXV
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE PIKROT
C   This subroutine determines how the J-levels for the given v-levels
C   will be superimposed on the v-levels.
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT DOUBLE PRECISION (B-H F N-U,W-Z)
      IMPLICIT CHARACTER*6 (S)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:33)
      JSUB=$PRSUB('PIKROT')
      IFLAG = 0
  105 WRITE(IOUT,*) 'The rotational energy levels are obtained through:
      WRITE(IOUT,*)              '1) A rigid rotor approximation.
      WRITE(IOUT,*)              2) Partial fit of spectroscopic
     &   'equation with B_e  D_e  etc.
      READ(INP,*,END=900) ANROT
      GO TO 901
  900 IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 105
      ELSE
        CALL IFALTI(ANROT,1)
      ENDIF
  901 WRITE(ISAVE,*) ANROT
      IF (ANROT.EQ.1) THEN
          CALL JMAX
          CALL RIGROT
        ELSEIF (ANROT.EQ.2) THEN
          CALL JMAX
          CALL SPECRO
C   Conceivably, another option for E(J) could have been specified:
C   that of reading them in for each given v-level.
C   At the moment, we will leave this to be treated as a subset of
C   the READVG program, where the energies of both the v and J
C   levels are read in at once.
        ELSE
          WRITE(IOUT,*) 'Error in rotational E 'n specification',ANROT
          WRITE(IOUT,*) 'Specify 1. or 2.'
          IFLAG = IFLAG + 1
          IF (IFLAG.LE.1) GO TO 105
          CALL IBTERM(ANROT,$SUB)
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE JMAX
C  This subroutine determines the maximum J-level for each given
C  V-level.
C  && Eventually, this program should allow one to specify
C  && a maximum energy, and then calculate all V,J levels
C  && available within the energy bounds.
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INEAT,INOV,VONLY,VUNIT(6:33)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXJN,  INPTS,  ICOLL
Cass  COMMON/MASS/EMASS,CMASS,AMASS,REDMAS,BRAD,CRAD,ARAD,RE
      $SUB=$PRSUB('JMAX    ')
C  Check to see if JMAX has already been called.  (This could happen
C  if ANRMAX=4.)  If so, then return immediately from this subroutine.
      IF(MAXJ(0).NE.0 .OR. VONLY.EQ.1) RETURN
      IFLAG = 0
107   WRITE(IOUT,*) 'The maximum rotational level available:'
      WRITE(IOUT,*)'      "I" is a fixed number for all v-levels'
      WRITE(IOUT,*)'      "J" differs for each v-level'
      READ(INP, * END=900) ANRMAX
        GO TO 901
900   IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 107
      ELSE
        CALL DFALTI(ANRMAX,1)
      ENDIF
901   WRITE(ISAVE,*) ANRMAX
      IFLAG=0
      IF(ANRMAX.EQ.1) THEN
902     WRITE(IOUT,*)  'What is the maximum J level
     &      for each v-level? (max. 200) <Default: 10>'
        READ(INP,*,END=903) MAX
        GO TO 904
903     IF(IOUT.EQ.ISUP) THEN
          INP=5
          IOUT=6
          GO TO 902
        ELSE
          CALL DFALTI(MAX,10)
        ENDIF
904     WRITE(ISAVE,*) MAX
        IF(MAX.GT.101 .OR. MAX.LT.1) THEN
          WRITE(IOUT,*)   Max. J value out of bounds.',MAX
          IFLAG=IFLAG+1
          IF(IFLAG.LE.1) GO TO 902
          CALL ISTERM(MAX,$SUB)
        ENDIF
        DO 100 I = 0,MAXV
          MAXJ(I)= MAX
100     CONTINUE
      ELSEIF (ANRMAX.EQ.2) THEN
905     WRITE(IOUT,*)' Input Jmax for each vibrational level. '
     &     ,'(Input ',MAXV,' values.) <Default: 10 for all v-levels>'
        READ(INP,*,END=906) (MAXJ(I), I=0,MAXV)
        GO TO 907
906     IF(IOUT.EQ.ISUP) THEN
          INP=5
          IOUT=6
          GO TO 905
        ELSE
          DO 908 I=0,MAXV
            CALL DFALTI(MAXJ(I),10)
908       CONTINUE
        ENDIF
907     WRITE(ISAVE,*) (MAXJ(I), I=0,MAXV)
      ELSE
        WRITE(IOUT,*)' Error in choosing Jmax specification.  ANRMAX
        WRITE(IOUT,*)' Specify 1 or 2.'
        IFLAG = IFLAG + 1
        IF (IFLAG.LE.1) GO TO 107
        CALL ISTERM(ANRMAX,$SUB)
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE RIGROT
C  This subroutine will fill in the ELEV array up to ELEV(MAXV,MAXJ),
C  by determining the J-levels for each v-level.
C  RIGROT should only be called after JMAX is called  and only
C  when VONLY=0.
C  A rigid rotor approximation assumes no coupling between
C  V and J (the alpha_e term in the spectroscopic equation)
C  and no centrifugal distortion (D_e term).
C
      IMPLICIT DOUBLE PRECISION (B-H,P,R-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      REAL*8 ARAD,AMASS,AVOGAD
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      C*MMON/COUNT/MAXV,MAXJ(0:10),MAXJN,  INPTS,  ICOLL
      COMMON/PHYSCY/PLANCK,YFOOTS,KGAS,SCEXP1,CLIGHT,PI,14EVEN
      COMMON/MASS/BMASS,CMASS,AMASS,REDMAS,BRAD,CRAD,ARAD,RE
      PARAMETER (AVOGADS6.022169E23)
      $SUB=$PRSUB('RIGROT')
      IF (RE.EQ.0.0D0) CALL $SUBDIE
C  Calculate B_e and then the rotational energy spacing...
C  The reduced mass was found in RXNSYS and is used to
C  calculate XMOMIN, the moment of inertia.
C   Value for PLANCK ---pass through phys. cnts. common.
C   Formula from W. J. Moore, Physical Chemistry
C  (1962) p.587.
      XMOMIN= REDMAS * RE * RE/AVOGAD
C             divide by N_av because the masses are per mole and the
C             moment of inertia of a single molecule is desired.
      BE= PLANCK/(8.0D0 * PI*PI*XMOMIN)
      DO 100 V=0,MAXV
         DO 105 J=0,MAXJ(V)
           XJJ1 = DBLE(J*(J+1))
           ELEV(V,J)=BE*PLANCK*XJJ1*AVOGAD + ELEV(V,0)
C                  here, we multiply by N_av because the energy of
C                  the rotational level is only given for each molecule.
105   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SPECRO
C  This subroutine will fill in the ELEV array up to ELEV(MAXV,MAXJ)
C  by determining the J-levels for each v-level.
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      DIMENSION RE(6),
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IERROR,INDAT,INDV,VONLY,VUNIT(3:33)
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
      CHARACTER*2 UNIT
      $SUB=$PRSUB('  SPECRO  ')
      IFLAG=0
900   WRITE(IOUT,*) ' How many spectroscopic constants (i.e the
     1     rotational energy) will you be inputting? <Def: 3>
        READ(INP,*,END=9010) IRE
        GO TO 902
901     IF(IOUT.EQ.ISUP) THEN
          INP=5
          IOUT=6
          GO TO 900
        ELSE
          CALL DFALTI(IRE,3)
        ENDIF
902     WRITE(ISAVE,*) IRE
        IF(IRE.LT.0.OR.IRE.GT.6) THEN
          IF(IFLAG.LE.1) THEN
          WRITE(IOUT,*)' Number of spectroscopic constants specified
     &                 ' is out of bounds.  IRE  Please respecify.'
            IFLAG=IFLAG+1
            GO TO 900
          ELSE
            CALL IBTERM(IRE,$SUB)
          ENDIF
        ENDIF
903     WRITE(IOUT,*) ' What is the unit used for the constants?'
        WRITE(IOUT,*)   <Default:  cm-1 >
        READ(INP,*,END=9040) UNIT
        GO TO 905
904     IF(IOUT.EQ.ISUP) THEN
          INP=5
          IOUT=6
          GO TO 903
        ELSE
          CALL DFALTC(UNIT,'cm-1')
        ENDIF
905     WRITE(ISAVE,*) UNIT
906     WRITE(IOUT,*) ' Input the spectroscopic constants
     &   alpha_e, B_e  D_e,
     &   etc. separated by commas :  IRE = constants)'
        IF(INDV.EQ.1) CALL NOVICE(10)
        READ(INP,*,END=9070) (RE(I),I=1,IRE)
        GO TO 908
907     IF(IOUT.EQ.ISUP) THEN
          INP=5
          IOUT=6
          GO TO 906
        ELSE
          CALL DFALTD(RE(1),0.3077162)
          CALL DFALTD(RE(2),10.5934116)
C           ...values for HCl:  Pank et al. J Mol Spectr 17 (1965) 132.
        ENDIF
908     WRITE(ISAVE,*) (RE(I),I=1,IRE)
        DO 203 I=1,IRE
          CALL SIU(30,RE(I),UNIT)
203     CONTINUE
        WRITE(IOUT,*)  V     J           E/V J)
        DO 600 V=0,MAXV
          DO 700 J=1,MAXJ(V)
            DV=DBLE(V)
            DJ=DBLE(J)
C     The first constant read in is, by convention  alpha_e
            ELEV(V,J)=ELEV(V,J)+ RE(1)*(DJ+1/2)*DV+0.5)
            DO 800 I=2,IRE
              ELEV(V,J)=ELEV(V,J)+ RE(I)*(DJ*(DJ+1))**I
800         CONTINUE
700       CONTINUE
600   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MORSE
C  MORSE will find the (V,0) row of the ELEV array for V=0 to MAXV.
C  which is also determined in this subroutine.
      IMPLICIT DOUBLE PRECISION (B-H,K-N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      REAL*8 AVOGAD, ARAD,AMASS
      CHARACTER*2 UNIT1,UNIT2
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IERROR,INDAT,INDV,VONLY,VUNIT(3:33)
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/PHYSCT/PLANCK,FBOLTZ,PMAC,SCEXP1,CLIGHT,PI,PAEVER
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
      COMMON/MASS/EMASS,GMASS,AMASS,REEMAS,RPAD,ZPAD,ARAD,RE
      PARAMETER (AVOGAD=6.022045D23)
      $SUB='MORSE  '
100   WRITE(IOUT,*)      Input the dissociation energy and its
     &   units, <default: 4.62    eV' >
      IF(INDV.EQ.1)  CALL NOVICE(10)
      READ(INP,*,END=9000) DE,UNIT1
      GO TO 901
900   IF(IOUT.EQ.ISUP)  THEN
        INP=5
        IOUT=6
        GO TO 100
      ELSE
        CALL DFALTD(DE,4.62D0)
        CALL DFALTC(UNIT1,'eV')
      ENDIF
901   WRITE(ISAVE,*) DE,UNIT1
      CALL SIU(10,DE,UNIT1)
      IF (RE.EQ.0.0D0) CALL BONDIS
905   WRITE(IOUT,*)  Input beta and the units of distance
     &   <default: 1.0951 bohr>
      IF(INDV.EQ.1) CALL NOVICE(30)
      READ(INP,*,END=906)            BETA,UNIT2
      GO TO 907
906   IF(IOUT.EQ.ISUP)  THEN
        INP=5
        IOUT=6
        GO TO 905
      ELSE
        CALL DFALTD(BETA,1.0951D0)
C          ...from Wright and Barclay  Chem Phys (1988)
C          ***  get full reference
        CALL DFALTC(UNIT2,'BO')
      ENDIF
907   WRITE(ISAVE,*) BETA,UNIT2
      DE=ABS(DE)
C The units for beta are input as distance; the inverse conversion
C  factor is needed, so the value of BETA is 'divided' by the
C  distance conversion factor (XXTOME)
      XXTOME=1.0
      CALL SIU(30,XXTOME,UNIT2)
      BETA   = BETA/XXTOME
C Formula from P.W.Atkins.  Physical Chemistry, Freeman &Co. (1984)
      HBARRE = AVOGAD*BETA*PLANCK/(2.0D0*PI)
      WE     = 3.0D0*HBARRE*DSQRT(DE/(2.)D0*REEMAS))
      WEXE   = HBARRE*HBARRE/(2.0D0*REEMAS)

C  Solve quadratic equation
C    wexe*(v+1/2)**2 - we(v+1/2) + DE = 0

      CALL QUAD(WEXE,-WE,DE,XX1,XX2)
C  Both roots will be positive.  We must decide  which root is
C  physically reasonable (i.e. which is smallest  since during
C  vibrational excitation to dissociation, the smallest v value
C  will be reached first and dissociation will occur.'
      XX=MIN(XX1,XX2)
      MAXV=  (XX-0.5)
C Check to make sure the value for MAXV is positive.
      IF(MAXV.LT.0) CALL IBTERM(MAXV,ISUB)
      WRITE(IOUT,*)    A De of   DE    Joules/mole  an Re of   RE
     &' meters and a Beta of   BETA   meters**-1 gave   MAXV+1
     &  vibrational levels.'
      IF(MAXV.GT.(MAXUN-1))THEN
        WRITE(IOUT,*)  ' There are more than   MAXUN   vibrational '
     &   levels so the max v is being set to   MAXUN-1
        MAXV=MAXUN-1
        ENDIF
103   WRITE(IOUT,701)  MAXUN-1
701   FORMAT (  Respecify De  Re  Beta (units with each)' /
     & '--type -1--//   accept max. v value--type 0--//  specify new
     & max. v value--type 1 to   I3   corresponding to the highest v
     & level you want. <Default: maxv=3> )
      READ(INP,*,END=9080) ANMAXV
      GO TO 909
908   IF(IOUT.EQ.ISUP)  THEN
        INP=5
        IOUT=6
        GO TO 103
      ELSE
        CALL DFALTI(ANMAXV,3)
      ENDIF
909   WRITE(ISAVE,*) ANMAXV
      IF(ANMAXV.EQ.-1) GO TO 102
      IF(ANMAXV.EQ.0)  THEN
        WRITE(IOUT,*)    MAXV =  ,MAXV
      ELSEIF(ANMAXV.GE.1.AND.ANMAXV.LE.MAXUN)  THEN
        MAXV=ANMAXV
      ELSE
        IFLAG=IFLAG+1
        WRITE(IOUT,*)    ANMAXV   is out of range. Respecify.
        IF(IFLAG.LE.1) GO TO 103
        CALL IBTERM(ANMAXV,$SUB)
      ENDIF
      DO 200 V=0,MAXV
        DV=DBLE(V)
        ELEV(V,0)=WE*(DV+0.5)-WEXE*(DV+0.5)**2
200   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE BONDIS
C  To determine the equilibrium bond length of species BC.
      IMPLICIT DOUBLE PRECISION (B-H,P,R-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      REAL*8 ARAD,AMASS
      CHARACTER*2 UNIT2
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/MASS/EMASS,CMASS,AMASS,REDMAS,BRAD,CRAD,ARAD,PE
      SSUP=$PRSUB('BONDIS')
      BCRAD=BRAD+CRAD
C                    ...default bond length is sum from RXNSYS.
  902 WRITE(IOUT,*)  'Input the equilibrium bond distance
     & ' and its units. <Default: ',BCRAD,' Bo >'
      IF(INOV.EQ.1) CALL NOVICE(30)
      READ(INP,*,END=9021) RE,UNIT2
      GO TO 904
 9021 IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 902
      ELSE
        CALL DFALTD(RE,BCRAD)
        CALL DFALTC(UNIT2,'Bo')
      ENDIF
  904 WRITE(ISAVE,*) RE,UNIT2
      CALL SIU(30,RE,UNIT2)
      RETURN
      END
```

```
      SUBROUTINE CHART(INDEX1, INDEX2, INDEX3)
C  CHART displays the periodic table, and acts as a look-up for the
C  relative atomic masses.
C  CHART returns, via its arguments, the atomic numbers of A, B and C.
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      IMPLICIT INTEGER(I-N)
      IMPLICIT CHARACTER*6 ($)
      INTEGER*4 VUNIT, VONLY
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      SSUB=$PRSUB('CHART')
      IFLAG=0
      WRITE(IOUT,*)
      WRITE(IOUT,10)
   10 FORMAT(20X,'PERIODIC CHART OF THE ELEMENTS')
      WRITE(IOUT,*)
      WRITE(IOUT,11)
   11 FORMAT(1X,'  H                                                 He')
      WRITE(IOUT,12)
   12 FORMAT(1X,'  1                                                 2')
      WRITE(IOUT,*)
      WRITE(IOUT,23)
   23 FORMAT(1X,'  3   4                                   5   6   7   8   9  10')
      WRITE(IOUT,24)
   24 FORMAT(1X,'  Li  Be                                  B   C   N   O   F  Ne')
      WRITE(IOUT,*)
      WRITE(IOUT,31)
   31 FORMAT(1X,' 11  12                                  13  14  15  16  17  18')
      WRITE(IOUT,32)
   32 FORMAT(1X,'  Na  Mg                                 Al  Si  P   S   Cl  Ar')
      WRITE(IOUT,*)
      WRITE(IOUT,41)
   41 FORMAT(1X,' 19  20  21  22  23  24  25  26  27  28  29',
     &       '  30  31  32  33  34  35  36')
      WRITE(IOUT,42)
   42 FORMAT(1X,'  K   Ca  Sc  Ti  V   Cr  Mn  Fe  Co  Ni  Cu',
     &       '  Zn  Ga  Ge  As  Se  Br  Kr')
      WRITE(IOUT,*)
      WRITE(IOUT,51)
   51 FORMAT(1X,' 37  38  39  40  41  42  43  44  45  46  47',
     &       ' 48  49  50  51  52  53  54')
      WRITE(IOUT,52)
   52 FORMAT(1X,'  Rb  Sr  Y   Zr  Nb  Mo  Tc  Ru  Rh  Pd  Ag',
     &       '  Cd  In  Sn  Sb  Te  I   Xe')
      WRITE(IOUT,*)
      WRITE(IOUT,61)
   61 FORMAT(1X,' 55  56  57  72  73  74  75  76  77  78  79',
     &       ' 80  81  82  83  84  85  86')
      WRITE(IOUT,62)
   62 FORMAT(1X,'  Cs  Ba  La  Hf  Ta  W   Re  Os  Ir  Pt  Au',
     &       '  Hg  Tl  Pb  Bi  Po  At  Rn')
      WRITE(IOUT,*)
      WRITE(IOUT,71)
   71 FORMAT(1X,' 87  88  89 104 105 106')
      WRITE(IOUT,72)
   72 FORMAT(1X,'  Fr  Ra  Ac  Ku  Ha')
      WRITE(IOUT,*)
      WRITE(IOUT,*)  'LANTHANIDE SERIES'
      WRITE(IOUT,81)
   81 FORMAT(1X,' 58  59  60  61  62  63  64  65  66  67  68',
     &       ' 69  70  71')
      WRITE(IOUT,82)
   82 FORMAT(1X,'  Ce  Pr  Nd  Pm  Sm  Eu  Gd  Tb  Dy  Ho  Er',
     &       '  Tm  Yb  Lu')
      WRITE(IOUT,*)
      WRITE(IOUT,*)  'ACTINIDE SERIES'
      WRITE(IOUT,91)
   91 FORMAT(1X,' Th  Pa   U  Np  Pu  Am  Cm  Bk  Cf  Es  Fm',
     &       '  Md  No  Lr')
      WRITE(IOUT,92)
   92 FORMAT(1X,' 90  91  92  93  94  95  96  97  98  99 100',
     &       ' 101 102 103')
  101 WRITE(IOUT,*)
      WRITE(IOUT,*) 'Select the number of three elements from the',
     &  'periodic table shown above.'
      READ(INP,*) INDEX1, INDEX2, INDEX3
      IF(INDEX1.LT.1.OR.INDEX1.GT.106) THEN
        IFLAG=IFLAG+1
        WRITE(IOUT,*) 'Incorrect selection of first element number.'
        WRITE(IOUT,*) 'Number should lie between 1 and 106.'
        IF(IFLAG.LE.1) GO TO 101
        CALL IBTERM(INDEX1,$SUB)
      ENDIF
      IF(INDEX2.LT.1.OR.INDEX2.GT.106) THEN
        IFLAG=IFLAG+1
        WRITE(IOUT,*) 'Incorrect selection of second element number.'
        WRITE(IOUT,*) 'Number should lie between 1 and 106.'
        IF(IFLAG.LE.1) GO TO 101
        CALL IBTERM(INDEX2,$SUB)
      ENDIF
      IF(INDEX3.LT.1.OR.INDEX3.GT.106) THEN
        IFLAG=IFLAG+1
        WRITE(IOUT,*) 'Incorrect selection of third element number.'
        WRITE(IOUT,*) 'Number should lie between 1 and 106.'
        IF(IFLAG.LE.1) GO TO 101
        CALL IBTERM(INDEX3,$SUB)
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE QUAD(AA,BB,CC,XX1,XX2)
C  To find the roots of A*X**2 + B*X + C = 0.
      IMPLICIT DOUBLE PRECISION (A-H O-Z)
      IMPLICIT CHARACTER*6 ($)
      INTEGER*4 VUNIT,VONLY
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      $SUB=$PRSUB('QUAD   ')
C  Set value of threshold so that calculation can be checked.
      THRESH=1.0D-06
      DTERM=BB*BB-4.0*AA*CC
      IF(DTERM.LT.0.0) THEN
        WRITE(IOUT,*)  ' Error in QUAD--determinant < 0
        CALL ABTERM(DTERM,$SUB)
      ENDIF
      DTERM=DSQRT(DTERM)
      XX1=(-BB-DTERM)/(2.0*AA)
      XX2=(-BB+DTERM)/(2.0*AA)
C  Check the parameters determined.
      RESULT=(AA*XX1 +BB)*XX1 + CC
      IF(RESULT.LT.-THRESH .OR. RESULT.GT.THRESH) THEN
        WRITE(IOUT,*)  ' Error in QUAD for XX1 calculation.
        CALL ABTERM(XX1,$SUB)
      ENDIF
      RESULT=(AA*XX2 +BB)*XX2 + CC
      IF(RESULT.LT.-THRESH .OR. RESULT.GT.THRESH) THEN
        WRITE(IOUT,*)  ' Error in QUAD for XX2 calculation.
        CALL ABTERM(XX2,$SUB)
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE SPEC(IROT)
C  If the argument in the call to this subroutine is 0, the
C  rotational levels will be calculated using spectroscopic parameters.
      IMPLICIT DOUBLE PRECISION (B-H K,N-U,W-Z)
      IMPLICIT INTEGER (A I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      DIMENSION WE(6) BE(6)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/ENERGY/ELEV(0:10,0:20), TEMP
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS  ICOLL
      CHARACTER*2 UNIT
      $SUB=$PRSUB('SPEC   ')
      DO 200 I=1,6
200   WE(I)=0.0
101   WRITE(IOUT,*)  ' What is the dissociation energy?
     &  <Default: 4.6175   eV' >
      IF(INOV.EQ.1) CALL NOVICE(10)
      READ(INP,* END=900) DE,UNIT
      GO TO 901
900   IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 101
      ELSE
        CALL DFALTD(DE,4.6175D0)
C           ...default values for HCl from Huber and Herzberg (1979).
        CALL DFALTC(UNIT,'EV')
      ENDIF
901   WRITE(ISAVE,*) DE, UNIT
      CALL SIU(10,DE,UNIT)
902   WRITE(IOUT,*)  ' How many spectroscopic constants for the
     1   vibrational energy will you be inputting? <Default: 3>
      READ(INP,*,END=903)   IWE
      GO TO 904
903   IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 902
      ELSE
        CALL DFALTI(IWE,3)
      ENDIF
904   WRITE(ISAVE,*)  IWE
908   WRITE(IOUT,*)  ' Input the spectroscopic constants We  WeXe  ',
     &  'WeY, etc. separated by commas (between 2 and 6 constants).',
     &  ' and at the end, specify the unit to be used
     &  ' (e.g. "CM" means cm**-1)'
      WRITE(IOUT,*)  '<Default: 2990.9463,  52.8185  0.3244, ' cm '>
      IF(INOV.EQ.1) CALL NOVICE(10)
      READ(INP,*,END=9080) (WE(I),I=1 IWE),UNIT
      GO TO 907
906   IF(IOUT.EQ.ISUP)THEN
        INP=5
        IOUT=6
        GO TO 908
      ELSE
        CALL DFALTD(WE(1)  2990.9463D0)
        CALL DFALTD(WE(2),-52.8185D0)
        CALL DFALTD(WE(3)  0.3244D0)
C           ...values for HCl:  Rank et al. J Mol Spectr 17 (1965) 122.
        CALL DFALTC(UNIT,'CM')
      ENDIF
907   WRITE(ISAVE,*)  (WE(I) I=1 IWE) UNIT
      DO 202 I=1 IWE
        CALL SIU(10,WE(I),UNIT)
202   CONTINUE
C
C The Newton-Raphson algorithm is used to determine the maximum v-level
C to a tolerance of 0.01 (relative).
C The initial guess is based on a second order quadratic.
C       WeXe*(v+1/2)**2 - We*(v+1/2)  + De = 0
C
      CALL QUAD(WE(2),-WE(1) DE X1 X2)
      XOLD=MIN(X1,X2)
C  Both roots should be positive; check to make sure this is true.
      IF(XOLD.LT.0)  CALL ABTERM(XOLD,$SUB)
      CALL NEWTRA(XOLD,FX,WE,IWE)
      MAXV=    XOLD-0.5)
      WRITE(IOUT,*)  ' An We of   WE(1)   and an WeXe of   WE(2)
     &' joules/molec gave   MAXV+1
     &  vibrational levels.
      IF(MAXV.GT.(MAXUN-1))THEN
        WRITE(IOUT,27)  ' There are more than   MAXUN   vibrational  .
     &  'levels so the max v is being set to  'MAXUN-1
27    FORMAT(2(1X))
        MAXV=MAXUN-1
        ENDIF
909   WRITE(IOUT,701)  MAXUN-1
701   FORMAT(/  Respecify De  and spectroscopic constants:
     &'--type -1:  /  accept max. v value--type 0: specify new max. v
     &'value--type 1 to   I3 /   corresponding to the highest v
     &'level you want. <Default: 1> '
      READ(INP,*,END=910) ANMAXV
      GO TO 911
910   IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 909
      ELSE
        CALL DFALTI(ANMAXV,1)
      ENDIF
911   WRITE(ISAVE,*) ANMAXV
      IF(ANMAXV.EQ.-1) GO TO 101
      IF(ANMAXV.EQ.0) THEN
        WRITE(IOUT,*)   MAXV =  ' MAXV
      ELSEIF(ANMAXV.GE.1.AND.ANMAXV.LE.MAXUN) THEN
        MAXV=ANMAXV
      ELSE
        IFLAG=IFLAG+1
        IF(IFLAG.LE.1) THEN
          WRITE(IOUT,*)   Answer is out of bounds: respecify.' ANMAXV
          GO TO 909
        ENDIF
```

```
        STOP 1011
      ENDIF
      IF(IROT.EQ.0) THEN
  912   WRITE(IOUT,*) ' How many spectroscopic constants for the'
     1    ' rotational energy will you be inputting? <Default: 2>'
        WRITE(IOUT,*)   '(You can specify up to 6, including v-J '.
     &    ' coupling terms.)'
        READ(INP,*,END=913) IRE
        GO TO 914
  913   IF(IOUT.EQ.ISUP) THEN
          INP=5
          IOUT=6
          GO TO 912
        ELSE
          CALL DFALTI(IRE,2)
        ENDIF
  914   WRITE(ISAVE,*) IRE
        IF(IRE.LT.1.OR.IRE.GT.6) THEN
          IF(IFLAG.LE.1) THEN
            WRITE(IOUT,*) ' Number of rotational constants specified '
     &        ' is out of bounds.   IRE   Please respecify.'
            GO TO 912
          ELSE
            CALL IBTERM(IRE,3SUB)
          ENDIF
        ENDIF
  915   WRITE(IOUT,*)   Input the spectroscopic constants '.
     &    'alpha_e, B_e, D_e, '
     &    'etc, separated by commas',
     &    ' and at the end, specify the unit to be used.'
        READ(INP,*,END=916) (RE(I),I=1,IRE),UNIT
        IF(INOV.EQ.1) CALL NOVICE(10)
        GO TO 917
  916   IF(IOUT.EQ.ISUP) THEN
          INP=5
          IOUT=6
          GO TO 915
        ELSE
          CALL DFALTD(RE(1),0.1D0)
          CALL DFALTD(RE(2),1.0D0)
          CALL DFALTC(UNIT,'CM')
        ENDIF
  917   WRITE(ISAVE,*) (RE(I),I=1,IRE),UNIT
        DO 203 I=1,IRE
          CALL SIU(30,RE(I),UNIT)
  203   CONTINUE
        WRITE(IOUT,*) '  V    J          E(V,J)'
      ELSE
        WRITE(IOUT,*) '  V              E(V,0)'
      ENDIF
      DO 600 V=0,MAXV+1
        DO 500 I=1,IWE
          ELEV(V,0)=WE(I)*(-(DBLE(V)+0.5)**I + ELEV(V,0)
  500   CONTINUE
        IF(IROT.EQ.0) THEN
          DO 700 J=0,MAXJ(V)+1
C   Sign convention for this summation is according to Farplus &
C   Porter p.482.  Note: check signs for higher rotational terms (e.g.,
C   H_e) and higher order vibrational terms (e.g., w_e*Y_e)
C    The first constant read in is, by convention, alpha_e
            DJ=DBLE(J)
            ELEV(V,J)= -RE(1)*(DBLE(V)+.5)*(DJ*(DJ+1))
            DO 800 I=2,IRE
              ELEV(V,J)=ELEV(V,J)+ RE(I)*((DJ*(DJ+1))**(I-1)*(-1)**I
  800       CONTINUE
            WRITE(IOUT,*) V,J,ELEV(V,J)
  700       CONTINUE
        ELSE
            WRITE(IOUT,*) V,ELEV(V,0)
        ENDIF
  600 CONTINUE
C  If the rotational constants have not been specified, then
C  some other method of calculating rotational energy levels
C  must be found.
      IF(IROT.NE.0) CALL PIKROT
      RETURN
      END
```

```
C
      SUBROUTINE NEWTRA(XOLD,OLDFX,WE,IWE)
C
C The Newton-Raphson algorithm is used to determine the maximum v-level
C to a tolerance of 0.01 (relative)
C The initial guess is based on a second order quadratic.
C       wexe(v+1/2)**2 - we(v+1/2) + De = 0
C
      IMPLICIT DOUBLE PRECISION (B-H,P-Z) N-O W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      COMMON/IO/INP,IOUT
      DIMENSION WE(6)
      SSUB=$PRSUB('NEWTRA')
      NRTOL=0.01
      DFDX=0.0
      ICOUNT=0
  400 FX=-OLDFX
C  Calculation of function (polynomial) and its derivative.
      I4EVER=1000
      DO 300 I=1,IWE
        FX=-WE(I)*((-XOLD)**I) + FX
        DFDX=WE(I)*DBLE(I)*((-XOLD)**(I-1)) + DFDX
  300 CONTINUE
      XNEW=XOLD-FX/DFDX
      IF(ICOUNT.GT.I4EVER) THEN
        WRITE(IOUT,*) ' Newton Raphson not converging '
        CALL ABTERM(XNEW,SSUB)
      ENDIF
      IF(DABS((XNEW-XOLD)/XNEW).GT.NRTOL) THEN
        ICOUNT=ICOUNT+1
        XOLD=XNEW
        GO TO 400
      ENDIF
      RETURN
      END
```

```
C
      FUNCTION EXIST(VV,JJ,VV1,JJ1)
C This function simply verifies that, for the v,J ---> v1,J1
C transition, the energy levels exist as defined in the potential
C energy curve construction.
      IMPLICIT DOUBLE PRECISION (B-H,P-Z) N-O W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      LOGICAL*1 EXIST
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS, ICOLL
C To see whether a transition exists, we check whether
C each value lies within the range of existing values calculated
C or specified earlier in the program.
      IF(VV.GE.0 .AND. VV.LE.MAXV .AND.
     &  VV1.GE.0 .AND. VV1.LE.MAXV ) THEN
        IF (JJ.GE.0 .AND. JJ.LE.MAXJ(VV)  .AND.
     &      JJ1.GE.0 .AND. JJ1.LE.MAXJ(VV1) ) THEN
          EXIST=.TRUE.
        ELSE
          EXIST=.FALSE.
        ENDIF
      ELSE
        EXIST=.FALSE.
      ENDIF
C       ...we must also ascertain that the initial level is unique
C       from the final level.
      IF(VV.EQ.VV1 .AND. JJ.EQ.JJ1) EXIST=.FALSE.
      RETURN
      END
```

```
      SUBROUTINE PIKK0U(ANK0UP)
C  Select method for obtaining K0UP vector for each v level...
C  where  K0UP(v,m) means k(v,m)-->k(v,m+2)
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INGV,VONLY,VUNIT(0:23)
      COMMON/KVIB/K0UP(0:10,0:20),KUPD(0:10,0:20),KUP0(0:10,0:20),
     1  KDOWNU(0:10,0:20),KDOWND(0:10,0:20),KDOWN0(0:10,0:20),
     2  K0UP(0:10,0:20),K0DOWN(0:10,0:20)
      COMMON/SPCIES/ISPEC,SSPEC(6,3)
      $SUB=$PRSUB('PIKK0U')
      IFLAG=0
      WRITE(IOUT,701) (SSPEC(L,ISPEC),L=1,6)
 701  FORMAT(1X,6A6)
 140   WRITE(IOUT,*)' Rate constants between rotational
     &  ' levels J and J+2 are:'
       WRITE(IOUT,*)'    *1) Calculated by scaling according to
     &  ' the rotational energy gap'
       WRITE(IOUT,*)'     2) Input from data file'
       READ(INP,* END=900) ANK0UP
       GO TO 901
 900   IF(IOUT.EQ.ISUP) THEN
         INP=5
         IOUT=6
         GO TO 140
       ELSE
         CALL DFALTI(ANK0UP,1)
       ENDIF
 901   WRITE(ISAVE,*) ANK0UP
       IF (ANK0UP.EQ.1) THEN
           CALL PK0TUP
         ELSEIF (ANK0UP.EQ.2) THEN
          CALL READIN(23,K0UP)
         ELSE
           WRITE(IOUT,*)' Error in specification of K(J-->J+2)'.
     &     ANK0UP
           WRITE(IOUT,*)' Answer should be 1. or 2.'
           IFLAG = IFLAG + 1
           IF (IFLAG.LE.2) GO TO 140
           CALL ISTERM('ANK0UP $SUB')
       ENDIF
      IF(IDEBUG.EQ.1) CALL PRNTVJ(23,K0UP)
      RETURN
      END
```

```
      SUBROUTINE PK0TUP
C  Calculation of k(0,J--->0,J+2) rate constant.
C  This subroutine mimics K01ALT (which determines the
C  rate coefficients for vibrational levels based on
C  the energy gap). PK0TUP determines the rate
C  coefficients for rotational levels when there is
C  no change in vibrational levels.
C
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      CHARACTER*2 UNIT
      LOGICAL*1 EXIST
      COMMON/SPCIES/ISPEC,SSPEC(6,3)
      COMMON/KVIB/K0UP(0:10,0:20),KUPD(0:10,0:20),KUP0(0:10,0:20),
     1  KDOWNU(0:10,0:20),KDOWND(0:10,0:20),KDOWN0(0:10,0:20),
     2  K0UP(0:10,0:20),K0DOWN(0:10,0:20),
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INGV,VONLY,VUNIT(0:23)
      COMMON/PHYSCT/PLANCK,KBOLTZ,RGAS,SCEXP1,CLIGHT,PI,14EVEP
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
      $SUB=$PRSUB('PK0TUP')
      WRITE(IOUT,701) (SSPEC(L,ISPEC),L=1,6)
 701  FORMAT(1X,6A6)
 900  WRITE(IOUT,*) 'Input k for v=0  j=0 --> v=0  j=2 transition
     &                 and its unit. <Default: 80.0  'cm'>'
      IF(INGV.EQ.1) CALL NGVICE(30)
      READ(INP,*,END=901) K0UP(0,0),UNIT
      GO TO 902
  901 IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 900
      ELSE
        CALL DFALTD(K0UP(0,0), 80.0D0)
        CALL DFALTC(UNIT,  'SE ')
      ENDIF
  902 WRITE(ISAVE,*) K0UP(0,0),UNIT
      CALL SIU(20,K0UP(0,0),UNIT)
      WRITE(IOUT,*) '$$ Rotational Transition Rate Coefficients $$
      WRITE(IOUT,*) ' THE  k(0,J-->0 J+2) CONSTANTS ARE :
      WRITE(IOUT,*) '       J        k(0,J-->0,J+2)
C user can set value of constant (recip) for alternate scaling
C  or change function altogether.
      RECIP=1.0
C Should be Rgas to make energy*recip unitless (delE/RT)
      RECIP=-RECIP/(RGAS*TEMP)
C pre-exponential factor needed to return consistent k(0,0-->0,2)
      PREEXP = DEXP(-RECIP*EDEL(0,0,0,2))
      PREEXP = PREEXP*K0UP(0,0)
      DO 10 J=0,MAXJ(0)
        IF (EXIST(0,J,0,J+2) )
     &    K0UP(0,J)=PREEXP*DEXP(RECIP*EDEL(0,J,0,J+2))
        WRITE(IOUT,*) J,K0UP(0,J)
 10   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE F01SCL(ANK01)
C  In the calculation of the rate coefficient for v-->v+1 scaling
C  the value from which all other values are scaled must first be
C  obtained.  This routine prompts for k(0,0-->1,0)
C  and  if rotational levels exist (i.e.,  vonly=0) it prompts
C  for k(0,0-->1,2) and k(0,2-->1,0).
C  When ANK01 is 1, Landau-Teler scaling is desired.
C  If the argument ANK01 is 2  use an alternate form of scaling.
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      CHARACTER*2 UNIT
      COMMON/SPCIES/ISPEC,SSPEC(6,3)
      COMMON/FVIE/K0UP(0:10,0:20),KUPD(0:10,0:20),KUP0(0:10,0:20),
     1  KDOWNU(0:10,0:20),KDOWND(0:10,0:20),KDOWN0(0:10,0:20),
     2  K0UP(0:10,0:20),K0DOWN(0:10,0:20),
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INGV,VONLY,VUNIT(0:23)
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/PHYSCT/PLANCK,KBOLTZ,RGAS,SCEXP1,CLIGHT,PI,14EVEP
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
      $SUB=$PRSUB('F01SCL')
      WRITE(IOUT,701) (SSPEC(L,ISPEC),L=1,6)
 701  FORMAT(1X,6A6)
 900  WRITE(IOUT,*) 'Input the units to be used for v --> v+1
     &                rate constants: <Default: 'cm'>'
      IF(INGV.EQ.1) CALL NGVICE(20)
      READ(INP,* END=901) UNIT
      GO TO 902
 901  IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 900
      ELSE
        CALL DFALTC(UNIT,  'SE ')
      ENDIF
 902  WRITE(ISAVE,*) UNIT
C         ... dummy call to SIU checks for valid unit
      CALL SIU(20,0.D0,UNIT)
C  Originally we prompted for k01 but this turned out to be inconvenient
C  (the literature usually cites k10)  so now we prompt for k10 and
C  immediately convert to k01 so that the remainder of the code is not
C  affected.  Users may wish to prompt for k01...
  903 WRITE(IOUT,*)  'Input k for v=1 J=0 -> v=0 J=0 transition:
      WRITE(IOUT,*)  '<Default: 1.4427>
      READ(INP,*,END=904) K1000
      GO TO 905
  904 IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 903
      ELSE
        CALL DFALTD(K1000,1.4427D0)
      ENDIF
  905 WRITE(ISAVE,*) K1000
      CONST=-1.0/(RGAS*TEMP)
      KUP0(0,0) = K1000*DEXP(EDEL(0,0,1,0)*CONST)
      IF (IDEBUG.EQ.1) WRITE(6,*)  KUP0:    KUP0(0,0),
      CALL SIU(20,KUP0(0,0),UNIT)
      IF (VONLY.EQ.0) THEN
  906   WRITE(IOUT,*)  'Input k for v=1 J=2 -> v=0 J=0 transition:
        WRITE(IOUT,*)  '<Default: 1.8D7>
        READ(INP,*,END=907) K1200
        GO TO 908
  907   IF(IOUT.EQ.ISUP) THEN
          INP=5
          IOUT=6
          GO TO 906
        ELSE
          CALL DFALTD(K1200,1.8D7)
        ENDIF
  908   WRITE(ISAVE,*) K1200
        KUP0(0,2) = K1000*DEXP(EDEL(0,0,1,2)*CONST)
        CALL SIU(20,KUP0(0,2),UNIT)
  909   WRITE(IOUT,*)  'Input k for v=1 J=0 -> v=1 J=0 transition:
        WRITE(IOUT,*)  '<Default: 1.2D7>'
        READ(INP,*,END=910) K1002
        GO TO 911
  910   IF(IOUT.EQ.ISUP) THEN
          INP=5
          IOUT=6
          GO TO 909
        ELSE
          CALL DFALTD(K1002,1.2D7)
        ENDIF
  911   WRITE(ISAVE,*) K1002
        KUP0(0,0) = K1002*DEXP(EDEL(0,0,1,0)*CONST)
        CALL SIU(20,KUP0(0,0),UNIT)
C        Must find KUPUP(0,all J) and KUP0(0,all J)
C      --this will be done using a modified PK0TUP routine.
        CALL PK0T2
      ENDIF
      IF(ANK01.EQ.1) THEN
        CALL F01LT
      ELSEIF(ANK01.EQ.2) THEN
        CALL K01ALT
      ELSE
        CALL ISTERM(ANK01 $SUB)
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE K01LT
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      LOGICAL*1 EXIST
      COMMON/KVIB/KUPUP(0:10,0:20),KUPD(0:10,0:20),KUP0(0:10,0:20),
     1  KDOWNU(0:10,0:20),KDOWND(0:10,0:20),KDOWN0(0:10,0:20),
     2  K0UP(0:10,0:20),K0DOWN(0:10,0:20)
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
      $SUB=$PRSUB('K01LT  ')
      DO 100 V = 0, MAXV
         DV1=DBLE(V+1)
         DO 100 J = 0, MAXJ(V)
            IF(EXIST(V,J,V+1,J-2)) KUPD(V,J)=DV1*KUPD(0,J)
            IF(EXIST(V,J,V+1,J   )) KUP0(V,J)=DV1*KUP0(0,J)
            IF(EXIST(V,J,V   ,J+2)) K0UP(V,J)=DV1*K0UP(0,J)
            IF(EXIST(V,J,V+1,J+2)) KUPUP(V,J)=DV1*KUPUP(0,J)
100      CONTINUE
      RETURN
      END
```

```
      SUBROUTINE KROT2
C  This subroutine mimics KROTUP, except that KROT2 determines the
C  rate coefficients for rotational levels when there is
C  a change of +2 in vibrational levels.
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      LOGICAL*1 EXIST
      COMMON/KVIB/KUPUP(0:10,0:20),KUPD(0:10,0:20),KUP0(0:10,0:20),
     1  KDOWNU(0:10,0:20),KDOWND(0:10,0:20),KDOWN0(0:10,0:20),
     2  K0UP(0:10,0:20),K0DOWN(0:10,0:20)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/PHYSCT/PLANCK,KBOLTZ,RGAS,SCEXP1,CLIGHT,PI,I4EVER
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
      $SUB=$PRSUB('KROT2  ')
C     k for v=0, j=0 ---> v=1, j=2 (KUPUP(0,0)) transition is known.
C     k for v=0, j=2 ---> v=1, j=0 (KUPD(0,0)) transition is known.
C     Now we will just build KUPUP(0,all J) and KUPD(0,all J) vectors.
C
      WRITE(IOUT,*)'$ Rotational Transition Rate Coefficients: KUPUP'
      WRITE(IOUT,*) ' THE  k(0,J-->1,J+2) constants are : '
      WRITE(IOUT,*) '          J        k(0,J-->1,J+2) '
C user can set value of constant (recip) for alternate scaling.
C  or change function altogether.
      RECIP=1.0
C Should be Rgas to make energy*recip unitless (dalE/RT)
      RECIP=-RECIP/(RGAS*TEMP)
C pre-exponential factor needed to return consistent k(0,0-->1,3)
      PREEXP = DEXP(-RECIP*EDEL(0,0,1,3))
      PREEXP = PREEXP*KUPUP(0,0)
      DO 100 J=0,MAXJ(0)-2
C       With the pre-exponential factor as defined, using (0,J,1,J+2)
C       will amount to taking the difference between the lengths
C       of the 2 transition vectors.
        IF (EXIST(0,J,0,J+2))
     &       KUPUP(0,J)=PREEXP*DEXP(RECIP*EDEL(0,J,1,J+2))
        WRITE(IOUT,*) J,KUPUP(0,J)
100   CONTINUE
      WRITE(IOUT,*) ' '
      WRITE(IOUT,*)'$ Rotational Transition Rate Coefficients:KUPD $'
      WRITE(IOUT,*) ' THE  k(0,J-->1,J-2) constants are : '
      WRITE(IOUT,*) '          J        k(0,J --> 1,J-2) '
C pre-exponential factor needed to return consistent k(0,2-->1,0)
      PREEXP = DEXP(-RECIP*EDEL(0,2,1,0))
      PREEXP = PREEXP*KUPD(0,2)
      DO 200 J=2,MAXJ(0)
        IF(EXIST(0,J,1,J-2))
     &       KUPD(0,J)=PREEXP*DEXP(RECIP*EDEL(0,J,1,J-2))
        WRITE(IOUT,*) J,KUPD(0,J)
200   CONTINUE
      WRITE(IOUT,*) ' '
      WRITE(IOUT,*)'$ Rotational Transition Rate Coefficients:KUP0 $'
      WRITE(IOUT,*) ' THE  k(0,J-->1,J) constants are : '
      WRITE(IOUT,*) '          J        k(0,J --> 1,J) '
      PREEXP=DEXP(-RECIP*EDEL(0,0,1,0))
      PREEXP=PREEXP*KUP0(0,0)
      DO 300 J=0,MAXJ(0)
        IF(EXIST(0,J,1,J)) KUP0(0,J)=PREEXP*DEXP(RECIP*EDEL(0,J,1,J))
        WRITE(IOUT,*) J,KUP0(0,J)
300   CONTINUE
      WRITE(IOUT,*) ' '
      RETURN
      END
```

```
      SUBROUTINE PIKK01(ANK01)
C  Selection of method to calculate rate coefficients for adjacent
C  vibrational level transitions.
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/KVIB/KUPUP(0:10,0:20),KUPD(0:10,0:20),KUP0(0:10,0:20),
     1  KDOWNU(0:10,0:20),KDOWND(0:10,0:20),KDOWN0(0:10,0:20),
     2  K0UP(0:10,0:20),K0DOWN(0:10,0:20)
      $SUB=$PRSUB('PIKK01')
101   WRITE(IOUT,*)' Rate constants between adjacent v-levels are:'
      WRITE(IOUT,*)'      *1) Calculated by Landau-Teller scaling'
      WRITE(IOUT,*)'       2) Calculated by alternate scaling'
      WRITE(IOUT,*)'       3) Input from data file'
      READ(INP,*,END=900) ANK01
      GO TO 901
900   IF(IOUT.EQ.ISUP)  THEN
        INP=5
        IOUT=6
        GO TO 101
      ELSE
        CALL DFALTI(ANK01 1)
      ENDIF
901   WRITE(ISAVE,*) ANK01
      IF (ANK01.EQ.1 .OR. ANK01.EQ.2) THEN
        CALL K01SCL(ANK01)
C                 ... arg. 1 means Landau-Teller scaling to be done
      ELSEIF (ANK01.EQ.3) THEN
        CALL READIN(21,K0UP)
      ELSE
        WRITE(IOUT,*)' Error in selection of rate constant '.
     &                ' calculation method.',ANK01
        WRITE(IOUT,*)' Answer should be 1, 2 or 3.'
        IFLAG = IFLAG + 1
        IF (IFLAG.LE.1) GO TO 101
        CALL IBTERM(ANK01 $SUB)
      ENDIF
      IF(IDEBUG.EQ.1) CALL PRNTVJ(23,K0UP)
      RETURN
      END
```

```
      SUBROUTINE K01ALT
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      LOGICAL*1 EXIST
      COMMON/KVIB/KUPUP(0:10,0:20),KUPD(0:10,0:20),KUP0(0:10,0:20),
     1  KDOWNU(0:10,0:20),KDOWND(0:10,0:20),KDOWN0(0:10,0:20),
     2  K0UP(0:10,0:20),K0DOWN(0:10,0:20)
      COMMON/PHYSCT/PLANCK,KBOLTZ,RGAS,SCEXP1,CLIGHT,PI,I4EVER
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
      $SUB=$PRSUB('K01ALT')
      CONST=1.0D0
      PREEXP=1.0D0
C           ....user can change values of exponential and
C                    pre-exponential constants.
      CONST=-CONST/(PLANCK*TEMP)
      DO 100 V = 0, MAXV
        DO 100 J = 0, MAXJ(V)
          IF(EXIST(V,J,V+1,J-2))
     &      KUPD(V,J)=PREEXP*DEXP(CONST*EDEL(V,J,V+1,J-2))*KUPD(0,J)
          IF(EXIST(V,J,V+1,J   ))
     &      KUP0(V,J)=PREEXP*DEXP(CONST*EDEL(V,J,V+1,J   ))*KUP0(0,J)
          IF(EXIST(V,J,V   ,J+2))
     &      K0UP(V,J)=PREEXP*DEXP(CONST*EDEL(V,J,V   ,J+2))*K0UP(0,J)
          IF(EXIST(V,J,V+1,J+2))
     &      KUPUP(V,J)=PREEXP*DEXP(CONST*EDEL(V,J,V+1,J+2))*KUPUP(0,J)
100     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CHECKR
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      LOGICAL*1 EXIST
      COMMON/KVIB/KUPUP(0:10,0:20),KUPD(0:10,0:20),KUPO(0:10,0:20),
     1  KDOWNU(0:10,0:20),KDOWND(0:10,0:20),KDOWNO(0:10,0:20),
     2  K0UP(0:10,0:20),K0DOWN(0:10,0:20)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INGV,VONLY,VUNIT(0:23)
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS  ICOLL
      $SUB=IPRSUB('CHECKR')
      $BLANK = '      '
C This routine checks whether the arrays used by MICREV are not 0 in
C the positions 0 0 -> MAXV, MAXJ(MAXV).
C If 0 then a warning is generated and the V, J position is flagged.
      DO 100 V = 0, MAXV
         DO 100 J = 0, MAXJ(V)
           IF(EXIST(V,J,V+1,J+2)) THEN
C  N.B. --if transition does not exist, then we do not want to
C  look up the rate coefficient in the array  since the index
C  might be out of bounds.
             IF (KUPUP(V,J).EQ.0.D0) WRITE(IOUT,706) V,J
           ENDIF
           IF(EXIST(V,J,V,J+2)) THEN
             IF (K0UP(V,J).EQ.0.D0) WRITE(IOUT,707)  V,J
           ENDIF
           IF(EXIST(V,J,V+1,J)) THEN
             IF (KUPO(V,J).EQ.0.D0) WRITE(IOUT,708)  V,J
           ENDIF
           IF(EXIST(V,J,V+1,J-2)) THEN
            IF (KUPD(V,J).EQ.0.D0) WRITE(IOUT,709)  V,J
           ENDIF
 100  CONTINUE
C
 706  FORMAT(' Array Kup,up has a 0 in position ' 2I4)
 707  FORMAT(' Array K 0,up has a 0 in position ' 2I4)
 708  FORMAT(' Array Kup,0 has a 0 in position ' 2I4)
 709  FORMAT(' Array Kup,down has a 0 in position ' 2I4)
C
C Values for the Kup,x (x=up,down,0) and K0,up are tabulated.
      WRITE(IOUT,705)
 705  FORMAT(2X,'V',2X,'J',2X,'K(V,J-->V+1,J+2)',2X,'F(V,J-->V,'
     &       'J+2)',2X,'K(V,J-->V+1,J)',2X,'K(V,J-->V+1,J-2)')
      DO 200 V = 0, MAXV
         DO 200 J = 0  MAXJ(V)
           IF(EXIST(V,J,V+1,J+2)) THEN
             WRITE(IOUT,711) V,J,KUPUP(V,J)
           ELSE
            WRITE(IOUT,712) V,J,$BLANK
           ENDIF
           IF(EXIST(V,J,V,J+2)) THEN
             WRITE(IOUT,713)      K0UP(V,J)
           ELSE
            WRITE(IOUT,714)      $BLANK
           ENDIF
           IF(EXIST(V,J V+1,J)) THEN
             WRITE(IOUT,715) KUPO(V,J)
           ELSE
            WRITE(IOUT,716)      $BLANK
           ENDIF
           IF(EXIST(V,J,V+1 J-2)) THEN
             WRITE(IOUT,717)      KUPD(V J)
           ELSE
            WRITE(IOUT,718)      $BLANK
           ENDIF
 200  CONTINUE
 711   FORMAT(2I3,D12.4,3X)
 712   FORMAT(2I3,3X,A6)
 713   FORMAT(21X,D12.4)
 714   FORMAT(21X,3X,A6)
 715   FORMAT(36X,D12.4)
 716   FORMAT(36X 3X,A6)
 717   FORMAT(51X D12.4)
 718   FORMAT(51X,3X,A6)
      RETURN
      END
```

```
      SUBROUTINE MICREV
C The J-->J-2 transition rate coefficients can be determined
C  by microscopic reversibility (along with V-->V-1 rates).
C
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      LOGICAL*1  EXIST
      COMMON/KVIB/KUPUP(0:10,0:20),KUPD(0:10,0:20),KUPO(0:10,0:20),
     1  KDOWNU(0:10,0:20),KDOWND(0:10,0:20),KDOWNO(0:10,0:20),
     2  K0UP(0:10,0:20),K0DOWN(0:10,0:20)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INGV,VONLY,VUNIT(0:23)
      COMMON/PHYSCT/PLANCK,PBOLTZ,RGAS,SCEXP1,CLIGHT,PI,I4EVER
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS  ICOLL
      $SUB=IPRSUB('MICREV')
      $BLANK = '      '
C This constant should be negative, so that kdown,x>kup
      CONST=-1.0/(RGAS*TEMP)
      WRITE(IOUT,*)              ****** K DOWN (USING MICROSCOPIC REVERSI
     &  BILITY) ******
C
C Values for the Kdown,x (x=up,down,0) and K0,down are tabulated.
      WRITE(IOUT,705)
 705  FORMAT(2X,'V',2X,'J',2X,'K(V,J-->V+1,J-2)',2X,'K(V,J-->V,'
     &       'J-2)',2X,'K(V,J-->V+1,J)',2X,'K(V,J-->V-1,J+2)')
      DO 200 V = 0, MAXV
         DO 200 J = 0  MAXJ(V)
           IF(EXIST(V,J,V+1,J-2)) THEN
             KDOWND(V,J)=KUPUP(V-1,J-2)*DEXP(EDEL(V,J,V-1,J-2)*CONST)
             WRITE(IOUT,711)  V,J KDOWND(V,J)
           ELSE
            WRITE(IOUT,712)  V,J,$BLANK
           ENDIF
           IF(EXIST(V,J V,J-2)) THEN
             K0DOWN(V,J)=K0UP(V,J-2)*DEXP(EDEL(V,J V,J-2)*CONST)
             WRITE(IOUT,713)  K0DOWN(V,J)
           ELSE
            WRITE(IOUT,714)      $BLANK
           ENDIF
           IF(EXIST(V,J,V-1 J)) THEN
             KDOWNO(V,J)=KUPO(V - 1,J)*DEXP(EDEL(V,J,V-1,J)*CONST)
             WRITE(IOUT,715)  KDOWNO(V,J)
           ELSE
            WRITE(IOUT,716)      $BLANK
           ENDIF
           IF(EXIST(V J,V-1,J-2)) THEN
             KDOWNU(V,J)=KUPD(V-1 J+2)*DEXP(EDEL(V,J,V-1,J+2)*CONST)
             WRITE(IOUT,717)  KDOWNU(V,J)
           ELSE
            WRITE(IOUT,718)      $BLANK
           ENDIF
 200  CONTINUE
 711   FORMAT(2I3,D12.4,3X)
 712   FORMAT(2I3,3X,A6)
 713   FORMAT(21X,D12.4)
 714   FORMAT(21X,3X,A6)
 715   FORMAT(36X,D12.4)
 716   FORMAT(36X 3X,A6)
 717   FORMAT(51X D12.4)
 718   FORMAT(51X,3X,A6)
      RETURN
      END
```

```
      SUBROUTINE PIFFRX(ANFMRX)
C Scaling can be done as a function of the energy of the rovibrational
C state in question or directly as a function of the vibrational and
C rotational quantum numbers involved.
C    For option 1) we chose scaling
C as a function of energy since this broad-brush approach seems to be
C more appropriate for global scaling.  Thus,
C   in BIMSCL, the v,J levels are viewed as unique energy levels.
C   The microscopic rate constants Kbim(v,J) are then mapped as
C   Kbim(E).  Choosing option (1) on this menu therefore
C   implies that vibrational and rotational excitation can be
C   'pooled' in a system where only their relative energies are
C   seen as significant.
C   If, for example, Elev(v,J) happens to be accidentally
C   degenerate with Elev(v',J'), then their krxn's will be equal.
C   The energy-scaled approach has another characteristic
C   that harmonic and anharmonic oscillators will yield different
C   scaling, since the Egap changes for the anharmonic oscillator.
C                For options 3, and 4)
C where a more detailed (and therefore more closely tied to the
C to the quantum number involved) description of the system is given
C scaling as a function of quantum number seems more appropriate.
C   It could alternatively be argued that, since vibrational motion
C   is fundamentally different from rotational motion, the
C   function krxn should be fitted separately to v, and then
C   separately to the J values.  For example,
C   one could intuitively picture a system whose probability of
C   reacting increases as its vibrations grow stronger,
C   although the rotation of the system has little or no effect
C   on whether it will react.  Such a system should then be
C   modelled by selecting options (3) or (4) from the menu.
C This approach for 3) and 4) also provides a way of bypassing the
C energy-scaled approach of 1) while still achieving the global
C scaling provided for in 1).
      IMPLICIT DOUBLE PRECISION (B-H,P,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      CHARACTER*2 UNIT
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/MASTR/KBIM(0:10,  0:20)
      $SUB=$PRSUB('PIFFRX')
      IFLAG = 0
103   WRITE(IOUT,*)  'The microscopic Kbim rate constants for '
     &              ,'the reaction A + BC ----> pdt, are:'
      IF(VONLY.EQ.0) THEN
        WRITE(IOUT,*)'    1) Scaled by v,J energy from Frxn(0,0).'
        WRITE(IOUT,*)'    2) Read from a file of Frxn(v,J)-v,J pairs.'
     &              ,'(Unit 5.)'
        WRITE(IOUT,*)'    3) Scaled from datafile of Frxn(v,0)-v pairs.'
     &              ,'(Unit 3.)'
        WRITE(IOUT,*)'    4) Scaled separately for the v-levels'
     &              ,'and subsequently for the J-levels.'
C       This last option has been added, to account for cases, where,
C       for example, the microscopic rate coefficient increases
C       monotonically w.r. to v-level, but has a distribution
C       peaked around an optimum J-level.
      ELSE
        WRITE(IOUT,*)'    1) Scaled by v-level energy from Frxn(0,0).'
        WRITE(IOUT,*)'    2) Read from a file of Frxn(v,0)-v pairs.'
     &              ,'(Unit 3.)'
      ENDIF
      READ(INP,*,END=900) ANFMRX
      GO TO 901
900   IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 103
      ELSE
        CALL DFALTI(ANFMRX,1)
      ENDIF
901   WRITE(ISAVE,*) ANFMRX
      IF (ANFMRX.EQ.1) THEN
          CALL BIMSCL
        ELSEIF (ANFMRX.EQ.2 .AND. VONLY.EQ.0) THEN
          CALL READVJ(25,KBIM)
        ELSEIF (ANFMRX.EQ.2 .AND. VONLY.EQ.1) THEN
          CALL READIN(24,KBIM)
        ELSEIF (ANFMRX.EQ.3.AND.VONLY.EQ.0) THEN
          CALL READIN(24,KBIM)
          CALL BMSCLJ(UNIT)
C In BMSCLJ, fo. a given v-level, the krxn(J) can be made to fit a
C predefined function (e.g. gaussian values w.r.to J-value).
        ELSEIF (ANFMRX.EQ.4.AND.VONLY.EQ.0) THEN
          CALL BMSCLV(UNIT)
C  BMSCLV is analogous to BMSCLJ; with J set at 0, the krxn(v)
C  is made to fit a predefined function (e.g. exponential increase
C  over v.)
          CALL BMSCLJ(UNIT)
        ELSE
          WRITE(IOUT,*)'Error in Frxn(v) specification ',ANFMRX
          IF(VONLY.EQ.0) WRITE(IOUT,*) 'Input 1, 2, 3 or 4.'
          IF(VONLY.EQ.1) WRITE(IOUT,*)'Input 1 or 2.'
          IFLAG = IFLAG + 1
          IF (IFLAG.LE.1) GOTO 103
            CALL IBTERM(ANFMRX,$SUB)
        ENDIF
      IF(IDEBUG.EQ.1) CALL PRNTVJ(24,KBIM)
      RETURN
      END
```

```
      SUBROUTINE BIMSCL
C  This subroutine allows the user to select how KBIM(v,J) will be
C  represented as a function of E.
C  In each form of scaling 3 guidelines are followed:
C    (1) the energy gap will be the independent variable (occurring
C        on the RHS).
C    (2) the equation must return the FVJBIM(0,0) value for the
C        v,J=0 level.
C    (3) the dimensions of the rate coefficient must be preserved.
C
      IMPLICIT DOUBLE PRECISION (B-H,P,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      CHARACTER*2 UNIT
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXJN,  INPTS,  ICOLL
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      DIMENSION SCEXP(10)
      COMMON/MASTR/FVJBIM(0:10,  0:20)
      $SUB=$PRSUB('BIMSCL')
      CALL FBIM00
      FBIMA=FVJBIM(0,0)
      IFLAG = 0
102   WRITE(IOUT,*)'Select  the form of the bimolecular rate'
     &            ,'vs. E curve according to number:
C put a call to novice here to write a message about defaults
C  Units will be energy**(-1)*time**(-1) for slope
C  and time**(-1) for intercept, so that the rate coefficient
C  will have overall units of time**(-1)
      WRITE(IOUT,*)'          1) Linear'
      WRITE(IOUT,*)'          2) Simple exponential'
      WRITE(IOUT,*)'          3) Cusp (two piecewise-continuous
     &   exponentials)'
      WRITE(IOUT,*)'          4) Sum of exponentials'
      READ(INP,*,END=903) ANSCL
      GOTO 904
903   IF(IOUT.EQ.ISUP) THEN
        INP  = 5
        IOUT = 6
        GOTO 102
      ELSE
        CALL DFALTI(ANSCL,1)
      ENDIF
904   WRITE(ISAVE,*) ANSCL
      IF (ANSCL.EQ.1) THEN
        CALL SCLK1(KBIMA)
      ELSEIF (ANSCL.EQ.2) THEN
        CALL SCLK2(KBIMA)
      ELSEIF (ANSCL.EQ.3 .OR. ANSCL .EQ. 4) THEN
        CALL SCLK34(KBIMA, ANSCL)
      ELSE
        WRITE(IOUT,*)' Error in potential curve specification ',ANSCL
        WRITE(IOUT,*)'  Specify 1, 2, 3 or 4.'
        IFLAG = IFLAG + 1
        IF (IFLAG.LE.2) GOTO 102
        CALL IBTERM(ANSCL, $SUB)
      ENDIF
      WRITE(IOUT,*)          $$$$$$ K MICROSCOPIC  BIMOLECULAR $$$$$$
      WRITE(IOUT,*)  ' THE  K(V-->RXN) CONSTANTS ARE :'
      WRITE(IOUT,*)           V              K(V-->RXN)  '
      DO 10 V=0,MAXV
        WRITE(IOUT,*) V,FVJBIM(V,0)
10    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE FBIM00
      IMPLICIT DOUBLE PRECISION (B-H,P,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      CHARACTER*2 UNIT
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXJN,  INPTS,  ICOLL
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      DIMENSION SCEXP(10)
      COMMON/MASTR/FVJBIM(0:10,  0:20)
      $SUB=$PRSUB('KBIM00')
900   WRITE(IOUT,*)  'Input the microscopic rate constant (and its '
     &            ,'  units) for the v=0 J=0 level'
      WRITE(IOUT,*)  ' <Default:  3.0D-2  cc/s> '
      IF(INOV.EQ.1) CALL NOVICE(20)
      READ(INP,*,END=901) KBIMA,UNIT
      GOTO 902
901   IF(IOUT.EQ.ISUP) THEN
        INP  = 5
        IOUT = 6
        GOTO 900
      ELSE
        CALL DFALTD(KBIMA,3.0D-2)
        CALL DFALTC(UNIT,'SE')
      ENDIF
902   WRITE(ISAVE,*) KBIMA, UNIT
      CALL SIU(20,KBIMA,UNIT)
      FVJBIM(0,0) = KBIMA
      RETURN
      END
```

```
      SUBROUTINE SCLK1(FBIMA)
C  SCLK1 is one of three scaling routines called by SIMSCL.
C  It scales linearly the elements of the
C  FVJBIM(V 0) vector from FVJBIM(0.0)  according to the size of
C  the energy gap between levels.
      IMPLICIT DOUBLE PRECISION (B-H F N-U W-Z)
      IMPLICIT INTEGER (A I-J L-M V)
      IMPLICIT CHARACTER*6 (S)
      COMMON/IO/INP IOUT ISAVE ISUP IDEBUG INDAT INOV VONLY VUNIT(0:23)
      COMMON/COUNT/MAXV MAXJ(0:10) MAXUN  INPTS  ICOLL
      COMMON/ENERGY/ELEV(0:10 0:20) TEMP
      COMMON/MASTR/FVJBIM(0:10   0:20)
      $SUB='SPRSUB('SCLK1 ')
 905      WRITE(IOUT,*)  By what factor does Fbim(1,0)  increase
     &                  ' relative to Fbim(0,0)?
          WRITE(IOUT,*)  (This will not be the actual slope since
     &                  'a further energy
          WRITE(IOUT,*)  scaling will be applied.) <default: 1.0>
          READ(INP,*,END=906)  SLOPE
          GOTO 907
 906      IF(IOUT.EQ.ISUP) THEN
            INP  = 5
            IOUT = 6
            GOTO 905
          ELSE
            CALL DFALTD(SLOPE,1.0D0)
          ENDIF
 907      WRITE(ISAVE,*)  SLOPE
        DO 200 V = 0, MAXV
         DO 200 J = 0, MAXJ(V)
           FVJBIM(V,J) = SLOPE*EDEL(V,J,0,0)*FBIMA/EDEL(1,0,0,0)*FBIMA
 200    CONTINUE
        RETURN
        END
```

```
      SUBROUTINE SCLK2(FBIMA)
C  SCLK2 is one of three scaling routines called by SIMSCL.
C  It scales exponentially the elements of the
C  FVJBIM(V,J) vector from FVJBIM(0,0).
      IMPLICIT DOUBLE PRECISION (B-H F N-U W-Z)
      IMPLICIT INTEGER (A I-J L-M V)
      IMPLICIT CHARACTER*6 (S)
      COMMON/IO/INP IOUT ISAVE ISUP IDEBUG INDAT INOV VONLY VUNIT(0:23)
      COMMON/COUNT/MAXV MAXJ(0:10) MAXUN  INPTS  ICOLL
      COMMON/ENERGY/ELEV(0:10 0:20) TEMP
      DIMENSION SCEXP(10)
      COMMON/MASTR/FVJBIM(0:10   0:20)
      $SUB = 'SCLK2'
 908      WRITE(IOUT,*)  What factor should be applied to the
     &       'argument of the exponential?
C The sign of the constant is positive  so that reactivity
C  will increase as v level increases.
          READ(INP,*,END=909)  CONST1
          GOTO 910
 909      IF(IOUT.EQ.ISUP)  THEN
            INP  = 5
            IOUT = 6
            GOTO 908
          ELSE
            CALL DFALTD(CONST1,1.0D-3)
          ENDIF
 910      WRITE(ISAVE,*) CONST1
          DO 300 V = 0  MAXV
           DO 300 J = 0, MAXJ(V)
             FVJBIM(V,J)   FBIMA*DEXP(ELEL(V,J,0,0)/ELEV(0,0)*CONST1)
 300      CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SCLK34(FBIMA,ANSCL)
C  SCLK34 is one of three scaling routines called by SIMSCL.
C  It scales exponentially (using more than one exponential) the
C elements of the FVJBIM(V,0) vector from FVJBIM(0,0).
C Note that choice 4 is not a set of piecewise-continuous exponentials
C but a true sum.
      IMPLICIT DOUBLE PRECISION (B-H F N-U W-Z)
      IMPLICIT INTEGER (A I-J L-M V)
      IMPLICIT CHARACTER*6 (S)
      CHARACTER*1 UNIT
      COMMON/IO/INP IOUT ISAVE ISUP IDEBUG INDAT INOV VONLY VUNIT(0:23)
      COMMON/COUNT/MAXV MAXJ(0:10) MAXUN  INPTS  ICOLL
      COMMON/ENERGY/ELEV(0:10 0:20),TEMP
      DIMENSION SCEXP(10)
      COMMON/MASTR/FVJBIM(0:10,  0:20)
      $SUB='SPRSUB('SCLK34 ')
      NEXP=2
C                 ...for option 4
          IF(ANSCL.EQ.4) THEN
 913        WRITE(IOUT,*)  How many exponentials will be
     &          summed? <default: 2>
            READ(INP,*,END=911)  NEXP
            GO TO 912
 911        IF(IOUT.EQ.ISUP) THEN
              INP=5
              IOUT=6
              GO TO 913
            ELSE
              CALL DFALTI(N,     2)
            ENDIF
 912        WRITE(ISAVE,*) NEXP
          ENDIF
C For a cusp the first value should be positive and the second negative.
C This is the sign that the calculation expects.
 914      WRITE(IOUT,*)  Input the constants for the exponentials
     &                  '(N.B. for a cusp  the first constant is positive
     &                  ' and the second is negative).
          READ(INP,*,END=915)  (SCEXP(I)  I=1,NEXP)  UNIT
          GOTO 916
 915      IF(IOUT.EQ.ISUP) THEN
            INP  = 5
            IOUT = 6
            GOTO 914
          ELSE
            DO 200 I=1,NEXP
              CALL DFALTD(SCEXP(I),10.0D0**(2-I))
 200        CONTINUE
C    No special significance to this formula; just wanted constants
C    to be 10.  1, 1/10 etc.
          ENDIF
 916      WRITE(ISAVE,*)  (SCEXP(I) I=1,NEXP)
C Since constants are in inverse energy units  the reciprocal units will
C  be found using DUMMY.
          DUMMY=1.0
          CALL SIU(10,DUMMY,UNIT)
          DO 303 I = 1,NEXP
            SCEXP(I)=SCEXP(I)/DUMMY
  303     CONTINUE
          IF(ANSCL.EQ.3) THEN
C The first exponential may be discontinuous from the second
C exponential at E = Ecrit  so they are forced to be piecewise
C continuous by a simple ratio applied to the second exponential.
C This seems to be the physically reasonable thing to do.
 917        WRITE(IOUT,*)  What is the energy Ecrit (and its units) at
     &          which the maximum of the cusp occurs?
            IF(KNOV.EQ.1)  CALL NOVICE(10)
            READ(INP,*,END=918)  CRITE  UNIT
            GOTO 919
 918        IF(IOUT.EQ.ISUP)  THEN
              INP  = 5
              IOUT = 6
              GOTO 917
            ELSE
              CALL DFALTD(CRITE,0.0D0)
              CALL DFALTC(UNIT,  'EV  ')
            ENDIF
 919        WRITE(ISAVE,*)  CRITE UNIT
            CALL SIU(10,CRITE,UNIT)
C crite = elev(0,0) is analogous to edel(V,J,0,0)
            FC1    = KBIMA*DEXP(1,CRITE - ELEV(0,0))/ELEV(0,0)*SCEXP(1);
            FC2    = FBIMA*DEXP(2,CRITE - ELEV(0,0))/ELEV(0,0)*SCEXP(2);
            FBIMB = FC1/FC2*FBIMA
            DO 100 V=0 MAXV
             DO 100 J=0 MAXJ(V)
              IF(ELEV(V,J).LE.CRITE) THEN
                FVJBIM(V,J) = FBIMA*DEXP(EDEL(V,J,0,0)/ELEV(0,0)*SCEXP(1))
              ELSE
                FVJBIM(V,J) = FBIMB*DEXP(EDEL(V,J,0,0)/ELEV(0,0)*SCEXP(2))
              ENDIF
 100        CONTINUE
          ELSE
            DO 400 V = 1  MAXV
              FVJBIM(V,0)=0.0D0
              DO 30 I = 1  NEXP
                FVJBIM(V,0) = FVJBIM(V,0) +
     &                        FBIMA*DEXP(EDEL(V,0,0,0)/ELEV(0,0)*SCEXP(I))
 30           CONTINUE
 400        CONTINUE
          ENDIF
        RETURN
        END
```

```
      SUBROUTINE BMSCLJ(UNIT)
C  The scaling of the microscopic bimolecular reaction rate
C   constants could have a variety of J-dependancies.
C   (i) The simplest model assumes no predictable
C   J-dependence---but this could
C   mean one of two things:  that k_bim varies randomly
C   according to J  or that it is a constant value over
C   all values of J.
C   The random number generated will be bounded by 0 and 2
C   (so that the average is 1) and
C   for a given v,J level, will multiply the value for Kvjbim(v, 0)
C   to produce a value for kvjbim(v,J).
C   (ii) K_bim might increase monotonically over J  so that the
C   reaction coefficient becomes larger as the rotational
C   excitation increases.  (This could also occur as a function
C   simply of the energy.)
C   (iii) k_bim might "decrease" monotonically over J  although this
C   does not seem entirely defensible (i.e.,  a small amount of
C   rotation would enable the species to align in the most
C   reactive position--see also Sathyamurthy's review  1983).
C   N.B. options (ii) and (iii) can be handled together since the same
C   function with different input parameters can give the desired
C   behavior (e.g. for a linear function, negative or positive slope).
C   (iv) Alternatively, k_bim might show a J-dependence which is
C   peaked (strongly or broadly) about a non-zero J.
C   In a very flexible program, the peak-J should be able to
C   be varied according to v-level.
C
C%% consider the use of UNIT in this section--should it be parsed?
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      CHARACTER*3 UNIT
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXJN,  INPTS,  ICOLL
      COMMON/MASTR/KVJBIM(0:10,0:20)
      $SUB='PRSUB('BMSCLJ')
      IFLAG = 0
102   WRITE(IOUT,*)'Select the scaling required for Krxn:'
      WRITE(IOUT,*)'                1) constant with respect to J'
      WRITE(IOUT,*)'                2) varies randomly between '
     &' 0 and 2 times the Krxn(V,J=0) value.'
      WRITE(IOUT,*)'                3) increases or decreases'
     &' monotonically over the energy of the J-levels'
      WRITE(IOUT,*)'                4) gaussian peaked at a given J value'
      READ(INP,*,END=903) ANSCL
      GOTO 904
903   IF(IOUT.EQ.ISUP) THEN
        INP  = 5
        IOUT = 6
        GOTO 102
      ELSE
        CALL DFALTI(ANSCL,1)
      ENDIF
904   WRITE(ISAVE,*) ANSCL
      IF (ANSCL.EQ.1) THEN
       CALL SCLJ1
      ELSEIF (ANSCL.EQ.2) THEN
       CALL SCLJ2
      ELSEIF (ANSCL.EQ.3) THEN
       CALL SCLJ3
      ELSEIF (ANSCL.EQ.4) THEN
       CALL SCLJ4
      ELSE
       WRITE(IOUT,*)' Error in potential curve specification  ANSCL
       WRITE(IOUT,*)' Specify 1, 2, 3 or 4.'
       IFLAG = IFLAG + 1
       IF (IFLAG.LE.1) GOTO 102
       CALL IBTERM(ANSCL,  $SUB)
      ENDIF
      WRITE(IOUT,*)'          ***** F MICROSCOPIC, BIMOLECULAR *******'
      WRITE(IOUT,*)' THE  K(v,J-->PRXN) CONSTANTS ARE :'
      WRITE(IOUT,*)'          V             J          K(v,J-->PRXN)'
      DO 10 V=0,MAXV
       DO 10 J=0,MAXJ(V)
        WRITE(IOUT,*) V,J,KVJBIM(V,J)
 10   CONTINUE
      IF(IDEBUG.EQ.1) CALL PRNTVJ(20,KVJBIM)
      RETURN
      END
```

```
      SUBROUTINE SCLJ1
C  SCLJ1 is one of four scaling routines called by BMSCLJ.
C  It maps KVJBIM(v,0) values into all remaining KVJBIM(v,J) positions
C  for a given v.
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      CHARACTER*3 UNIT
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXJN  INPTS  ICOLL
      COMMON/MASTR/KVJBIM(0:10,  0:20)
      $SUB='PRSUB('SCLJ1  ')
      DO 200 V = 0, MAXV
       DO 200 J = 1, MAXJ(V)
         KVJBIM(V,J) = KVJBIM(V,0)
200   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SCLJ2
C  SCLJ2 is one of four scaling routines called by BMSCLJ.
C  It generates random values for KVJBIM(v,J) centred on KVJBIM(v,0)
C  bounded by 0 and 2*KVJBIM(v,0) for a given v.
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      DOUBLE PRECISION R(1111)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXJN  INPTS  ICOLL
      COMMON/MASTR/KVJBIM(0:10,  0:20)
      $SUB='SCLJ2'
908       WRITE(IOUT,*)' Input   seed   for random number generator
     &        (real number between 1.0E0 and 2.0E20).'
          READ(INP,*,END=909) DSEED
          GOTO 910
909       IF(IOUT.EQ.ISUP) THEN
            INP  = 5
            IOUT = 6
            GOTO 908
          ELSE
            CALL DFALTD(DSEED,142857D0)
          ENDIF
910       WRITE(ISAVE,*) DSEED
          IF (DSEED.LT.1.0D0 .OR. DSEED.GT.2147483647.D0) THEN
            WRITE(IOUT,*) 'Seed is out of range.'
            IFLAG = IFLAG + 1
            IF (IFLAG.LE.1) GOTO 908
            CALL ABTERM(DSEED  $SUB)
          ENDIF
C We generate all the random numbers we need with a single call to the
C IMSL routine GGUBS.  Note that the numbers 11 and 101 are dependent
C on the size of the vibrational and rotational arrays.
          NR = 11*101
          CALL GGUBS(DSEED,NR,R)
          DO 300 V = 0, MAXV
           DO 300 J = 1, MAXJ(V)
             KVJBIM(V,J) = KVJBIM(V,0)*2.0D0*R(V*101+J+1)
300       CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SCLJ3
C  SCLJ3 is one of four scaling routines called by BMSCLJ.
C  It scales linearly by the energy gap the elements
C  of the KVJBIM(V,J) array from KVJBIM(V,0).
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXJN,  INPTS,  ICOLL
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/MASTR/KVJBIM(0:10   0:20)
      $SUB=$PRSUB('SCLJ3 ')
905   WRITE(IOUT,*)' Enter the ratio K_rxn(v=0 j=1)/K_rxn(v=0,j=0):'
      READ(INP,*,END=906) SLOPE
      GOTO 907
906   IF(IOUT.EQ.ISUP) THEN
        INP  = 5
        IOUT = 6
        GOTO 905
      ELSE
        CALL DFALTD(SLOPE,1.0D0)
      ENDIF
907   WRITE(ISAVE,*) SLOPE
      DO 300 V = 0, MAXV
        DO 200 J = 1, MAXJ(V)
C  The unitless approach is easy to implement and values
C  on the order of 1.0 seem physically reasonable.
C  Note for scaling subroutines--the energy values should not be
C  based on ELEV(0,0)=0 because numerous zero-divide errors can
C  occur--especially wherever scaling is done (put a check for
C  this in READIN.
          KVJBIM(V,J) = SLOPE*EDEL(V,J,0,0)*KVJBIM(V,0)/EDEL(1 0,0,0)
     &                + KVJBIM(V,0)
200   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SCLJ4
C  SCLJ4 is one of four scaling routines called by BMSCLJ.
C  It scales using a gaussian centred at a given J level.
      IMPLICIT DOUBLE PRECISION (B-H,P,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:*),MAXUN,  INPTS,  ICOLL
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/MASTR/KVJBIM(0:10,  0:20)
      $SUB=$PRSUB('  SCLJ4  ')
C  Unlike the other J-scaling subroutines called by BMSCLJ
C  this routine will have to prompt for input explicitly
C  at each of the allowed v-levels.
C
      DO 100 V=0 MAXV
  901   WRITE(IOUT,*)  *** Scaling for J-values at the v level   V
        WRITE(IOUT,*)  What is the J-value at which the k-bim is'
     &                  ' at its peak? <default: Jpeak = Jmax/2> .'
     &                  '(Please input a decimal number.)'
        READ(INP,*,END=902)  PEAK
        GOTO 903
  902   IF(IOUT.EQ.ISUP) THEN
          INP  = 5
          IOUT = 6
          GOTO 901
        ELSE
          CALL DFALTD(PEAK,DBLE(MAXJ(V)/2))
        ENDIF
  903   WRITE(ISAVE,*) PEAK
C
C  How wide is the Gaussian?  Since chemists are already
C  quite familiar with the idea of "full width at half maximum"
C  the question is tailored to this concept.
C
  905   WRITE(IOUT,*) 'What is a J-value at which the k-bim is'
     &                  ' one-half of its max. height?'
        WRITE(IOUT,*)    <Default (arbitrary):  Jpeak - 1/8*Jmax>
        READ(INP,*,END=906)  FWHM
        GOTO 907
  906   IF(IOUT.EQ.ISUP) THEN
          INP  = 5
          IOUT = 6
          GOTO 905
        ELSE
          FFWHM=PEAK-DBLE(MAXJ(V))/8
          CALL DFALTD(FWHM,FFWHM)
        ENDIF
  907   WRITE(ISAVE,*) FWHM
C
C  Note that it doesn't matter if a J-value to the right
C  or to the left of Jpeak is specified.  Since the formula for
C  a Gaussian squares the difference.
C
C   The formula of interest for the dimensionless case is:
C         f(x) = const*exp[-a*(x-xm)**2]
C                                        where xm is the x value where
C                                        f(x) is at a maximum.
C         f'(x) = -const*2*a*(x-xm)*exp(-a*(x-xm)**2)
C
C        which is zero when x = xm.
C
C        The value of kvjbim at J=0 is used to find the value of
C        the pre-exponential constant:
C
C        f(0) = const*exp[-a*xm**2]
C
C   Thus  const = f(0)*exp[a*xm**2]
C
C        The full-width at half-maximum is used to find  a .
C
C        Note that f(xh) = 1/2*f(xm)
C                                        where xm is the x value at which
C                                        the peak is one-half its max.
C        Therefore,
C                      a =    ln 2
C                             ----------
C                             (xh-xm)**2
C
C   In this subroutine  PEAK = xm and FWHM = xh- ALPHA = a.
C
        ALPHA = DLOG(2.0D0)/((PEAK-FWHM)**2)
        CONST=KVJBIM(V,0)*DEXP(ALPHA*PEAK*PEAK)
        DO 200 J = 1 MAXJ(V)
           KVJBIM(V,J) = CONST*DEXP(-ALPHA*(DBLE(J)-PEAK)**2)
  200   CONTINUE
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE BMSCLV
C The scaling of the microscopic bimolecular reaction rate
C constants could have a variety of v-dependencies.
C (1) The simplest model assumes no predictable
C v-dependence---but this could
C mean one of two things:  that K_bim varies randomly
C according to v  or that it is a constant value over
C all values of v.
C The random number generated will be bounded by 0 and 2
C so that the average is 1) and
C will multiply the value for Kvjbim(0, 0)
C to produce a value for kvjbim(V,0).
C (2) K_bim might increase monotonically over V, so that the
C reactio coefficient becomes larger at the vibrational
C excitation increases.  (This could also occur as a function
C simply of the energy.)
C (3) K_bim might 'decrease' monotonically over V.
C N.B. options 1) and 2)) can be handled together since the same
C function with different input parameters can give the desired
C behavior (e.g  for a linear function, negative or positive slope
C or for an exponential function a negative or positive exponent).
C
      IMPLICIT DOUBLE PRECISION (B-H,P,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
      COMMON/MASTR/KVJBIM(0:10,0:20)
      $SUB=$PRSUB('BMSCLV')
      IFLAG = 0
  102 WRITE(IOUT,*)'Select the scaling required for Prxn:
      WRITE(IOUT,*)'                1) constant with respect to V'
      WRITE(IOUT,*)'                2) varies randomly between
     &' 0 and 2 times the Prxn(V=0 J=0) value.'
      WRITE(IOUT,*)'                3) increases or decreases
     &' linearly over energy of v-level'
      WRITE(IOUT,*)'                4) increases or decreases
     &' exponentially over V'
      READ(INP,*,END=903) ANSCL
      GOTO 904
  903 IF(IOUT.EQ.ISUP) THEN
        INP  = 5
        IOUT = 6
        GOTO 102
      ELSE
        CALL DFALTI(ANSCL,1)
      ENDIF
  904 WRITE(ISAVE,*) ANSCL
C         Prompt for value of KVJBIM(0,0) and its units.
      CALL PBIM00
      IF (ANSCL.EQ.1) THEN
       CALL SCLV1
      ELSEIF (ANSCL.EQ.2) THEN
       CALL SCLV2
      ELSEIF (ANSCL.EQ.3) THEN
       CALL SCLV3
      ELSEIF (ANSCL.EQ.4) THEN
       CALL SCLV4
      ELSE
       WRITE(IOUT,*)  Error in potential curve specification  ANSCL
       WRITE(IOUT,*)  Specify 1, 2, 3 or 4.
       IFLAG = IFLAG + 1
       IF (IFLAG.LE.2) GOTO 102
       CALL ISTERM(ANSCL, $SUB)
      ENDIF
      WRITE(IOUT,*)'          ***** P MICROSCOPIC  BIMOLECULAR *******
      WRITE(IOUT,*)   THE  P(V,J-->PXN) CONSTANTS ARE :
      WRITE(IOUT,*)            V              J              P(V,J-->PXN)
      DO 100 V=0,MAXV
        WRITE(IOUT,*) V,PVJBIM,V 0,
  100 CONTINUE
      IF(IDEBUG.EQ.1) CALL PRNTMX(10, PVJBIM)
      RETURN
      END
```

```
      SUBROUTINE SCLV1
C  SCLV1 is one of four scaling routines called by BMSCLV.
C  It maps PVJBIM(0,0) values into all remaining PVJBIM(V,0) positions
      IMPLICIT DOUBLE PRECISION (B-H,P,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
      COMMON/MASTR/KVJBIM(0:10,  0:20)
      $SUB=$PRSUB('  SCLV1  ')
      DO 100 V = 0  MAXV
         KVJBIM(V,0) = KVJBIM(0,0)
C            ...the value of KVJBIM(0,0) is obtained during READIN.
C               READVJ or PBIM00.
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SCLV2
C  SCLV2 is one of four scaling routines called by BMSCLV.
C  It generates random values for KVJBIM(V,0) centred on KVJBIM(0,0)
C  bounded by 0 and 2*KVJBIM(0,0)
C *** may want to merge with SCLJ2 into a generic random number
C      generator.
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      DOUBLE PRECISION R(11)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
      COMMON/MASTR/KVJBIM(0:10,  0:20)
      $SUB = 'SCLV2'
 908  WRITE(IOUT,*)  'Input ''seed'' for random number generator'
     &     '(real number between 1.0E0 and 2.0E101.'
      READ(INP,* END=909) DSEED
      GOTO 910
 909  IF(IOUT.EQ.ISUP)  THEN
        INP  = 5
        IOUT = 6
        GOTO 908
      ELSE
        CALL DFALTD(DSEED,142857D0)
      ENDIF
 910  WRITE(ISAVE,*) DSEED
      IF (DSEED.LT.1.0D0 .OR. DSEED.GT.2147483647.D0)  THEN
        WRITE(IOUT, *)  'Seed is out of range.'
        IFLAG = IFLAG + 1
        IF (IFLAG.LE.1) GOTO 908
        CALL ABTERM(DSEED, $SUB)
      ENDIF
C We generate all the random numbers we need with a single call to the
C IMSL routine GGUBS.  Note that the number 11 is dependent
C on the size of the vibrational and rotational arrays.
      NR = 11
      CALL GGUBS(DSEED,NR,R)
      DO 300 V = 0  MAXV
        KVJBIM(V,0) = KVJBIM(0,0)*2.0D0*R(V+1)
300   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SCLV3
C  SCLV3 is one of four scaling routines called by BMSCLV.
C  It scales linearly by the energy gap the KVJBIM(V,0) elements
C  of the KVJBIM(V,J) array from KVJBIM(0,0).
C Eqn to use is kvjbim(v,0) = kvjbim(0,0)*v*slope + kvjbim(0,0)
C where slope is input by user
C Rather than have slope input directly program asks for ratio
C of first rate constant to zeroth: ratio = kvjbim(1,0)/kvjbim(0,0)
C then from first eqn (dividing by kvjbim(0,0)) for v=1:
C ratio = 1*slope + 1
C therefore slope = ratio - 1
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/MASTR/KVJBIM(0:10,  0:20)
      $SUB=SPRSUB('SCLV3  ')
 905  WRITE(IOUT,*)  'Enter the ratio k_rxn(v=1,j=0)/k_rxn(v=0,j=0)'
      READ(INP,*,END=906) RATIO
      GOTO 907
 906  IF(IOUT.EQ.ISUP)  THEN
        INP  = 5
        IOUT = 6
        GOTO 905
      ELSE
        CALL DFALTD(RATIO,2.0D0)
      ENDIF
 907  WRITE(ISAVE,*) RATIO
      SLOPE = RATIO - 1.0D0
      DO 200 V = 1, MAXV
C  The unitless approach is easy to implement and values of the
C  on the order of 2.0 seem physically reasonable.
        KVJBIM(V,0)=SLOPE*EDEL(V,0,0,0)*KVJBIM(0,0)/EDEL(1,0,0,0)
     &     + KVJBIM(0,0)
200   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SCLV4
C  SCLV4 is one of four scaling routines called by BMSCLV.
C  It scales exponentially the KVJBIM(V,0) elements
C  of the KVJBIM(V,J) array from KVJBIM(0,0).
C Eqn to use is kvjbim(v,0) = kvjbim(0,0)*exp(alpha*v)
C where alpha is input by user
C Rather than have alpha input directly program asks for ratio
C of first rate constant to zeroth: ratio = kvjbim(1,0)/kvjbim(0,0)
C then from first eqn rearranging (for v=1): ratio = exp(alpha*1)
C therefore alpha = ln(ratio)
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/MASTR/KVJBIM(0:10,  0:20)
      $SUB=SPRSUB('SCLV4  ')
 905  WRITE(IOUT,*)  'Enter the ratio k_rxn(v=1,j=0)/k_rxn(v=0,j=0)'
      READ(INP,*,END=906) RATIO
      GOTO 907
 906  IF(IOUT.EQ.ISUP)  THEN
        INP  = 5
        IOUT = 6
        GOTO 905
      ELSE
        CALL DFALTD(RATIO,1.5D0)
      ENDIF
 907  WRITE(ISAVE,*) RATIO
      ALPHA = DLOG(RATIO)
      DO 200 V = 0, MAXV
C  The unitless approach is easy to implement and values of the
C  on the order of 1.5 seem physically reasonable.
        KVJBIM(V,0) = KVJBIM(0,0)*DEXP(ALPHA*DBLE(V))
200   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE COLLCT
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
      COMMON/KVIB/KUPUP(0:10,0:20),KUPD(0:10,0:20),KUP0(0:10,0:20),
     1   KDOWNU(0:10,0:20),KDOWND(0:10,0:20),KDOWN0(0:10,0:20)
     2   KUUP(0:10,0:20),KDDOWN(0:10,0:20)
      COMMON/KVIB3/KUU3(3,0:10,0:20),KUD3(3,0:10,0:20),
     1   KDU3(3,0:10,0:20),KDD3(3,0:10,0:20),KD03(3,0:10,0:20),
     2   K0U3(3,0:10,0:20),K0D3(3,0:10,0:20),K003(3,0:10,0:20)
      $SUB=SPRSUB('COLLCT')
C     WRITE(6,*)  ' ICOLL = ',ICOLL
C  ICOLL denotes which collision partner is involved: A, M or B2
CSmocheck
      DO 100 V=0,MAXV
        DO 100 J=0,MAXJ(V)
          KUU3(ICOLL,V,J)=KUPUP(V,J)
          KUD3(ICOLL,V,J)=KUPD(V,J)
          KU03(ICOLL,V,J)=KUP0(V,J)
          KDU3(ICOLL,V,J)=KDOWNU(V,J)
          KDD3(ICOLL,V,J)=KDOWND(V,J)
          KD03(ICOLL,V,J)=KDOWN0(V,J)
          K0U3(ICOLL,V,J)=K0UP(V,J)
          K0D3(ICOLL,V,J)=K0DOWN(V,J)
 100  CONTINUE
C$check
      RETURN
      END
```

```
      SUBROUTINE DIFFUS
C  The krxn(V,J) should be examined to see which values are
C  not reasonable for the system of interest.
C  For a bimolecular reaction  this would imply that those krxn
C  which exceed the collision frequency possible  according
C  to the temperature and pressure of the system
C  should have an upper limit which is equal to or less
C  than the rate of diffusion.  The rate-limiting step is
C  no longer the speed of chemical conversion  but the
C  speed at which one molecule can meet another molecule.
C  In this subroutine  the collision frequency is calculated
C  and any krxn(V,J) which exceeds it is set equal to it.
C  If the user is modelling a unimolecular reaction  or an ideal
C  reaction which is not limited by the rate of diffusion  then
C  'no upper limit' in the subroutine DIFFUS should be specified.
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      REAL*8 AVOGAD  AMASS  ARAD
      PARAMETER (AVOGAD=6.02305D23)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:16),MAXOP,  INPTS,  IOLLL
      COMMON/MASTP/KVJBIM(0:10,0:20)
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/PHYSCT/PLANCK,FBOLTZ,RGAS,SCEXP1,CLIGHT,PI,I4EVER
      COMMON/MASS/RMASS,CMASS,AMASS,REDMAS,RRAD,CRAD,ARAD,RE
      COMMON/POP/NV(0:10,0:20),NTOT,CONCA,CONCBC,CONCM,JCOLL(3)
      $SUB=$PRSUB('DIFFUS')
      IFLAG = 0
102   WRITE(IOUT,*)'Should the microscopic rate constants '
     &  ' have an upper limit equal to that of the maximum collision '
     &  ' frequency possible?'
      WRITE(IOUT,*)         '1) yes  establish this as an upper limit
      WRITE(IOUT,*)         '2) the upper limit is proportional to
     &  ' the collision frequency'
      WRITE(IOUT,*)         '3) no upper limit (e.g. unimolecular rxn.
      READ(INP,*,END=903) ANDIFF
      GOTO 904
903   IF(IOUT.EQ.ISUP) THEN
        INP  = 5
        IOUT = 6
        GOTO 102
      ELSE
        CALL DFALTI(ANDIFF,1)
      ENDIF
904   WRITE(ISAVE,*) ANDIFF
      IF (ANDIFF.EQ.3) THEN
       GO TO 104
      ELSEIF (ANDIFF.NE.1.AND.ANDIFF.NE.2) THEN
       WRITE(IOUT,*)  Error in upper limit specification.  ANDIFF
       WRITE(IOUT,*)' Specify 1, 2 or 3.'
       IFLAG = IFLAG + 1
       IF (IFLAG.LE.1) GOTO 102
       CALL IBTERM(ANDIFF,$SUB)
      ENDIF
C
C The rate of collision, Z, may be expressed in a form analogous to
C the rate of reaction, with a diffusion coefficient  k_diff:
C                    Z = k_diff [A] [BC]
C where [A] and [BC] are the concentrations of the species involved
C in collision.
C For each vJ level
C we want to compare
C k_diff with the k_vJbim coefficient input to the
C program.  Since only collisional processes are considered in this
C simulation, no k_vJbim should exceed k_diff.
C The formula for Z is from I.N. Levine, Physical Chemistry
C (Mcgraw-Hill  1978), p. 421:
C     Z = pi*(r_A + r_BC)**2*(8*R*T/pi)**(1/2)
C       *(N_A/V)*(N_BC/V)
C Therefore  k_diff = Z/([A]*[BC]*N_Av)
C where the additional N_Av in the denominator comes in because we are
C considering the number of 'moles' of collisions rather than the
C number of collisions.
C     or       k_diff = pi*(r_A + r_BC)**2*(8*R*T/pi)
C                       *(RECiprocal-Reduced-Mass)**(1/2)*N_Av
C   This will be used to estimate the upper limit for
C   the bimolecular rate coefficient.
C   The molecular diameter is estimated as the sum of atomic
C   radii  whose values are passed through common block MASS.
C   The atomic masses and radii are found in subroutine RXNSYS.
C   Since we are concerned only with reactive collisions (A+BC)
C   we ignore M+BC collisions in determining k_diff.
      RBC = (BRAD+CRAD)
      BCMASS=BMASS+CMASS
      RECRM = (1.0D0/AMASS + 1.0D0/BCMASS)*AVOGAD
C  AMASS and BCMASS come in on a 'per mole' basis....we need them
C  on a 'per molecule' basis  so we divide the mass by N_Av  which
C  is equivalent to multiplying the reciprocal by N_Av as above.
      KDIFF=PI*(ARAD+RBC)**2*DSQRT((8.0D0*RGAS*TEMP/PI)*RECRM)*AVOGAD
      WRITE(IOUT,700) KDIFF
700   FORMAT(' Collision coefficient (k_diff) is ',D12.4,'  sec-1 ')
      IF(ANDIFF.EQ.2) THEN
105     WRITE(IOUT,700) 'What is the ratio of the upper limit to the
     &          ' collision coefficient? <Default: 1.0>'
        READ(INP,*,END=908) RATIO
        GOTO 909
908     IF(IOUT.EQ.ISUP) THEN
          INP  = 5
          IOUT = 6
          GOTO 105
        ELSE
          CALL DFALTD(RATIO,1.0D0)
        ENDIF
909     KDIFF=RATIO*KDIFF
      ENDIF
      DO 100 V=0  MAXV
        DO 100 J=0,MAXJ(V)
          IF(KVJBIM(V,J).GT.KDIFF) THEN
            WRITE(IOUT,701) V,J,KVJBIM(V,J),KDIFF
            KVJBIM(V,J)=KDIFF
701         FORMAT(' Kvjbim(',2I3,' ) exceeds the collision
     &          coefficient:   D12.4    >   D12.4 / ' Value reset. ')
          ENDIF
100   CONTINUE
104   RETURN
      END
```

```
      SUBROUTINE PIFPOP(ANPOP)
C  Selection of the form of population distribution.
      IMPLICIT DOUBLE PRECISION (B-H,K-N,O-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      COMMON/POP/NV(0:10,0:20),NTOT,CONCA,CONCBC,CONCM,JCOLL(3)
      COMMON/PHYSCT/PLANCK,FBOLTZ,RGAS,SCEXP1,CLIGHT,PI,I4EVER
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/COUNT/MAXV,MAXJ(0:16),MAXOP,  INPTS,  IOLLL
      $SUB=$PRSUB('PIFPOP')
      IFLAG = 0
104   WRITE(IOUT,*)'  Is the population distributed according to
      WRITE(IOUT,*)'                1) The Boltzmann equation?
      WRITE(IOUT,*)'                2) Laser excitation?
      WRITE(IOUT,*)'                3) Delta function?
      WRITE(IOUT,*)'                4) A datafile of NV(V,J) v J pairs?
      IF (VONLY.EQ.0) WRITE(IOUT,*)
     &'                5) A datafile of NV(V,J) v J triples?
      READ(INP,*,END=900) ANPOP
      GO TO 901
900   IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 104
      ELSE
        CALL DFALTI(ANPOP,1)
      ENDIF
901   WRITE(ISAVE,*) ANPOP
      IF (ANPOP .EQ.1) THEN
        CALL BOLTZ
      ELSEIF (ANPOP .EQ.2) THEN
        CALL LASER
      ELSEIF (ANPOP .EQ.3) THEN
        CALL DELTA
      ELSEIF (ANPOP .EQ.4) THEN
        CALL READIN(61,NV)
        IF(VONLY.EQ.0) CALL JBOLTZ
      ELSEIF (ANPOP .EQ.5.AND.VONLY.EQ.0) THEN
        CALL READVJ(63,NV)
      ELSE
        WRITE(IOUT,*)' Error in choice of vibrational population.
        IF(VONLY.EQ.1)  THEN
          WRITE(IOUT,*) ANPOP,'  Specify 1, 2, 3 or 4.'
        ELSE
          WRITE(IOUT,*) ANPOP,'  Specify 1, 2, 3, 4 or 5.
        ENDIF
        IFLAG = IFLAG + 1
        IF (IFLAG.LE.1) GO TO 104
        CALL IBTERM(ANPOP,$SUB)
      ENDIF
C The populations of each level (no matter how obtained) must be
C added up to get a total population.
      NTOT=0.0D0
C For accuracy's sake  the smallest numbers should be added together
C  first  and the largest numbers at the end.
C  We will not do a size-sort  however  but instead go backwards
C  through the loop to approximate a smaller-to-larger addition.
C cjc, 1991.5.26: what we are actually doing is calculating
C the partition function, so that we can divide by it to get true
C V_V s.
      DO 404 V=MAXV,0,-1
        DO 404 J=MAXJ(V),0,-1
          NTOT=NV(V,J)+NTOT
404   CONTINUE
      WRITE(IOUT,*)  '  The initial  unnormalized  population is  ,
     &          NTOT
      NCONST=CONCBC/NTOT
      DO 405 V=0,MAXV
        DO 405 J=0,MAXJ(V)
          NV(V,J)=NV(V,J)*NCONST
          WRITE(IOUT,*)  V,J,ELEV(V,J),NV(V,J)
405   CONTINUE
C adjust concbc ever so slightly to avoid small numerical errors
C later on....cjc, 1991.6.26.  Then adjust conca and concm
C to keep them in sync.
      CALC = CONCBC
      CONCBC=0.D0
      DO 406 V=MAXV,0,-1
        DO 406 J=MAXJ(V),0,-1
          CONCBC = CONCBC+NV(V,J)
406   CONTINUE
      CONCA = CONCBC/CALC*CONCA
      CONCM = CONCBC/CALC*CONCM
      CONCBC= CONCBC/CALC*CONCBC
        WRITE(IOUT,*)   The populations have been adjusted to an
     &          initial total value of   CONCBC
     &          using a scaling factor of   NCONST
      IF(IDEBUG.EQ.1)  CALL PRNTV(20,NV)
      RETURN
      END
```

```
      SUBROUTINE BOLTZ
C     Calculation of population based on the Boltzmann distribution.
C degeneracy factor for the population level is from Harplus & Porter
C page 474.
C Since Elev is per-molecule, the Boltzmann constant is used.
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/PHYSCT/PLANCK,FBOLTZ,RGAS,SCEXP1,CLIGHT,PI,I4EVER
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
      COMMON/POP/NV(0:10,0:20),NTOT,CONCA,CONCBC,CONCM,JCOLL(3)
      $SUB=$PRSUB('BOLTZ  ')
      BCONST=-1.0D0/(RGAS*TEMP)
      WRITE(IOUT,*)     ******* INITIAL BOLTZMANN DISTRIBUTION (jo/mol)
     &  *******
      WRITE(IOUT,7)
    7 FORMAT(11X,  'V',  11X,  'J',  12X,  'Elev',  15X,  'N(v)')
      NV0 = NV(0,0)
      DO 100 V=0,MAXV
        DO 100 J=0,MAXJ(V)
          BATERM = BCONST*EDEL(0,0,V,J)
          NV(V,J) = DBLE(2*J + 1)*NV0*DEXP(BATERM)
          IF(NV(V,J).GT.0.0D0) WRITE(IOUT,*)  V,J,ELEV(V,J),NV(V,J)
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE LASER
C This subroutine resets the population distribution to simulate a laser
C excitation experiment in which molecules in a particular v, J level
C make a transition to another level.   Thus the number of molecules in
C the initial level decreases while the number of molecules in the final
C level increases.
C
C Parameters: None.
C Calling routine: PIKPOP.
C Called routines: Boltz.
C Variables:
C       VLO, JLO, VHI, JHI---these are the indices for the lower (VLO,
C       JLO) and upper (VHI, JHI) levels corresponding to the transition
C       selected.
C
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      LOGICAL*1 EXIST
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/POP/NV(0:10,0:20),NTOT,CONCA,CONCBC,CONCM,JCOLL(3)
      $SUB=$PRSUB('LASER  ')
  200 WRITE(IOUT,*) 'What transition is involved in the laser'
      WRITE(IOUT,*) 'excitation?  (Input 4 integers, separated by a'
      WRITE(IOUT,*)  comma corresponding to v,j ---> v', j" E.g.,'
      WRITE(IOUT,*) 'v=0, J=0 ---> v=3, J=2 would be input as 0 0 3 2)'
      WRITE(IOUT,*) 'Default is 0,0 ---> 1,0 transition.'
      READ(INP,*,END=900) VLO,JLO,    VHI,JHI
      GO TO 901
  900 IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 200
      ELSE
        CALL DFALTI(VLO,0)
        CALL DFALTI(JLO,0)
        CALL DFALTI(VHI,1)
        CALL DFALTI(JHI,0)
      ENDIF
  901 WRITE(ISAVE,*) VLO,JLO,VHI,JHI
      IFLAG=0
C     Check to se  that VLO and VHI are within the proper range.
      IF(.NOT.EXI.,(VLO,JLO,VHI,JHI)) THEN
        WRITE(IOUT,*) 'Transition levels out of bounds ',VLO,JLO,
     &  VHI,JHI
          IFLAG = IFLAG + 1
          IF (IFLAG.LE.1) GO TO 200
          WRITE(IOUT,*) 'Terminating in LASER due to v or J bounds.'
          CALL IBTERM(VHI,$SUB)
      ENDIF
  101 WRITE(IOUT,*) 'What fraction of the initial population'
      WRITE(IOUT,*) 'in the lower level has been excited to the upper'
      WRITE(IOUT,*) 'level?'
      READ(INP,*,END=902) FRACEX
      GO TO 903
  902 IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 101
      ELSE
        CALL DFALTD(FRACEX,0.5D0)
      ENDIF
  903 WRITE(ISAVE,*) FRACEX
C Check to see that fraction is valid.
      IF(FRACEX.LE.0.0.OR.FRACEX.GT.1.0) THEN
        WRITE(IOUT,*) 'Fraction', FRACEX, 'must be between 0 and 1.'
        WRITE(IOUT,*) 'Respecify fraction within bounds.'
        IFLAG = IFLAG + 1
        IF(IFLAG.LE.1) GO TO 101
        WRITE(IOUT,*) 'Program terminating in LASER.'
        CALL ABTERM(FRACEX,$SUB)
      ELSE
        CALL BOLTZ
        NVEXC       = NV(VLO,JLO)*FRACEX
        NV(VHI,JHI) = NV(VHI,JHI) + NVEXC
        NV(VLO,JLO) = NV(VLO,JLO) - NVEXC
      ENDIF
    RETURN
    END
```

```
      SUBROUTINE DELTA
C This routine is an idealiced representation of a very restrictive
C nascent population.   It will set
C all populations to zero, and the fraction in the desired level
C will be equal to the total population.
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      LOGICAL*1 EXIST
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/POP/NV(0:10,0:20),NTOT,CONCA,CONCBC,CONCM,JCOLL(3)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
      $SUB=$PRSUB('DELTA  ')
      IFLAG=0
  200 WRITE(IOUT,*)  ' Situate the delta function at which v,J level?
      WRITE(IOUT,*)    <Default: 1 1>'
      READ(INP,*,END=900) ANV,ANJ
      GO TO 901
  900 IF(IOUT.EQ.ISUP)  THEN
        INP=5
        IOUT=6
        GO TO 200
      ELSE
        CALL DFALTI(ANV,1)
        CALL DFALTI(ANJ,1)
      ENDIF
  901 WRITE(ISAVE,*) ANV,ANJ
C Check to see that this level is within the proper range.
C (To use 'EXIST' function, look for 0,0->anv,anj transition.)
      IF(.NOT.EXIST(0,0,ANV,ANJ)) THEN
        WRITE(IOUT,*)  ' Specified level out of bounds   ANV,ANJ,
     &    Value must be between 0 and  ,MAXV
        IFLAG=IFLAG+1
        IF (IFLAG.LE.1) GO TO 200
        WRITE(IOUT,*)  ' Terminating in DELTA due to v bounds.'
        CALL IBTERM(ANV,$SUB)
      ENDIF
      DO 100 V=0,MAXV
        DO 100 J=0,MAXJ(V)
          NV(V,J)=0.0D0
  100 CONTINUE
      NV(ANV,ANJ)=1.0D0
      RETURN
      END
```

```
      SUBROUTINE JBOLTZ
C  Given an (V,NV) vector, this routine will calculate values for
C  the J-levels for each vibrational level, assuming a Boltzmann
C  distribution of the J-levels.  For the moment, we are assuming
C  the rotational temperature will be coincident with the
C  experimental temperature.
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/PHYSCT/PLANCK,FBOLTZ,RGAS,SCEXP1,CLIGHT,PI,I4EVER
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
      COMMON/POP/NV(0:10,0:20),NTOT,CONCA,CONCBC,CONCM,JCOLL(3)
      $SUB=$PRSUB('JBOLTZ  ')
      BCONST = -1.0D0/(KBOLTZ*TEMP)
      WRITE(IOUT,*)  ' ******* J-level BOLTZMANN DISTRIBUTIONS ******
      WRITE(IOUT,7)
    7 FORMAT(10X,  'V',  10X,  'J',  12X,  'Elev',  15X,  'N(v)')
      DO 100 V=0,MAXV
        NV0 = NV(V,0)
        DO 100 J=0,MAXJ(V)
C degeneracy factor for the population level is from Harplus & Porter
C page 474.
C Since EDEL is per-molecule, the Boltzmann constant is used.
          NV(V,J) = DBLE(2*J + 1)*NV0*DEXP(BCONST*EDEL(V,0,V,J))
          IF(NV(V,J).GT.0.0D0) WRITE(IOUT,*)  V,J ELEV(V,J),NV(V,J)
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE FINIT(ICTR1)
C* Note: this subroutine is currently a mix of o.d and new code
C* and probably does not work!  When the time comes to get the
C* Eigenvalue solution part working this code will have to be reworked..
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      LOGICAL*1 EXIST
      COMMON/FVIB/FUPUP(0:10,0:20),KUPD(0:10,0:20),KUPO(0:10,0:20),
     1  FDOWNU(0:10,0:20),FDOWND(0:10,0:20),FDOWNO(0:10,0:20),
     2  FOUP(0:10,0:20),FODOWN(0:10,0:20),
     3  FLUMP(200,200)
      COMMON/FVIB3/FUU3(3,0:10,0:20),FUD3(3,0:10,0:20),
     1  KDU3(3,0:10,0:20),KDD3(3,0:10,0:20),FDO3(3,0:10,0:20),
     2  FOU3(3,0:10,0:20),KOD3(3,0:10,0:20),FUO3(3,0:10,0:20),
      COMMON/MASTR/FVJBIM(0:10,0:20)
      COMMON/POP/NV(0:10,0:20),NTOT,CONC(3),JCOLL(3),
C                          ...for CONCA CONCBC CONCM respectively
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
C ICOLL denotes which collision partner is involved: A, BC or M
      $SUB=$PRSUB('FINIT  ')
      MAXFLU=200
      ISPEC=3
C                 ...the maximum order of the matrix FLUMP
      DO 100 INITRW = 1,MAXFLU
        DO 100 INITCL = 1,MAXFLU
          FLUMP(INITRW,INITCL) = 0.0D0
  100 CONTINUE
      ICTR1 = 0
      DO 200 V1 = 0,MAXV
         DO 200 J1 = 0,MAXJ(V1)
            ICTR1 = ICTR1 + 1
            IF(ICTR1.GE.MAXFLU) GO TO 201
            ICTR2 = 0
            DO 300 V2 = 0,MAXV
              DO 300 J2 = 0,MAXJ(V2)
                ICTR2 = ICTR2 + 1
                IF(ICTR2.GE.MAXFLU) GO TO 301
                V=V2
                J=J2
                FLUMP(ICTR1,ICTR2)=0.0D0
                DNVVJ=0.0D0
                DO 400 IC=1,ISPEC
                  IF(EXIST(V-1,J,V,J)) THEN
                    DNVVJ= CONC(IC)*KDO3(IC,V,J)  + DNVVJ
                    WRITE(6,*) IC,V,J =  IC,V,J
                    WRITE(6,*) KDO3    =  KDO3(IC,V,J)
                    WRITE(6,*) DNVVJ   =  DNVVJ
                  ENDIF
                  IF(EXIST(V-1,J+2,V,J)) THEN
                    DNVVJ= CONC(IC)*KUO3(IC,V,J)  + DNVVJ
                  ENDIF
                  IF(EXIST(V-1,J-2,V,J)) THEN
                    DNVVJ = CONC(IC)*KDD3(IC,V,J)+ DNVVJ
                  ENDIF
                  IF(EXIST(V,J-2,V,J)) THEN
                    DNVVJ= CONC(IC)*KOD3(IC,V,J)  + DNVVJ
                  ENDIF
                  IF(EXIST(V+1,J,V,J)) THEN
                    WRITE(6,*) '2nd if-exist) DNVVJ =',DNVVJ
                    WRITE(6,*) IC,V,J =  IC,V,J
                    WRITE(6,*)'KUO3   =',KU(3(IC,V,J)
                    DNVVJ= CONC(IC)*KUO3(IC,V,J)+DNVVJ
                  ENDIF
                  IF(EXIST(V+1,J+2,V,J)) THEN
                    DNVVJ= CONC(IC)*KUU3(IC,V,J)  + DNVVJ
                  ENDIF
                  IF(EXIST(V+1,J-2,V,J)) THEN
                    DNVVJ= CONC(IC)*KUD3(IC,V,J)+DNVVJ
                  ENDIF
                  IF(EXIST(V,J+2,V,J)) THEN
                    DNVVJ= CONC(IC)*KOU3(IC,V,J)  + DNVVJ
                  ENDIF
  400           CONTINUE
                DNVVJ=CONC(JCOLL(1))*FVJBIM(V,J)  +DNVVJ
C                 ...CONC(1) since reactions will only occur with A
                FLUMP(ICTR1,ICTR2)=DNVVJ
                WRITE(6,*)'ICTR1,ICTR2, FLUMP() =  ',ICTR1,ICTR2,
     &                FLUMP(ICTR1,ICTR2)
  300         CONTINUE
  301         CONTINUE
  200   CONTINUE
  201   WRITE(IOUT,*)   'Eigenvalue matrix is of order ',ICTR1
        WRITE(IOUT,*) ' Maximum order of matrix allowed is ',MAXFLU
        IF(IDEBUG.EQ.1) THEN
          IF(ICTR1.NE.ICTR2) WRITE(6,*) 'Internal ICTR error in FINIT.
          CALL IBTERM(ICTR1,$NAME)
        ENDIF
        RETURN
        END
```

```
      SUBROUTINE SOLVER
C This routine is at the heart of time integration.  It calls IMSL
C routine DGEAR which will perform the integration of the o.d.e.'s
C (which are contained in routine MASTER as required by DGEAR).
C
C Some comments on the parameters required by the call to DGEAR:
C IVLVL is the number of o.d.e's to be solved.
C MASTER is the subroutine which evaluates the o.d.e.'s.  It is
C        called as 'CALL MASTER(IVLVL,T,NVPAC,DNVPAC)'
C JACOB is the subroutine which evaluates the Jacobian matrix for
C       the o.d.e.'s (In our case the Jacobian is 1 so this is
C       essentially a dummy routine).
C &&&&Is DGEAR ignoring the Jacobian or is it evaluating it by
C other means???
C T is the independant variable  time.
C H is the step-size used in the first call; then regulated by
C      DGEAR as necessary to maintain the required tolerance (see
C      TOL).
C TEND is the value of T at which the integration will cease.
C TOL -- relative error bound used on the first call.
C    N.B. -- we found it very impt. to have a small H and TOL.
C METH -- basic method to be used by DGEAR.  Selected '2' for
C         solns of stiff differential eq'ns.
C MITER -- selection of the method of iteration.  MITER = 0 means
C          that a dummy JACOB can be used.
C INDEX -- set to 1 on input, to indicate that this is the first
C          call to DGEAR.
C IWK -- integer array of size IVLVL.  Not used when MITER = 0.
C WK -- in this instance  WK must have dimension 10*IVLVL + 1.
C IER -- output error code from DGEAR.
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      COMMON/IC/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(5:23)
      COMMON/PHYSCT/PLANCK,KBOLTZ,RGAS,SCEXP1,CLIGHT,PI,I4EVER
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
      COMMON/POP/NV(0:10,0:20),NTOT,CONCA,CONCBC,CONCM,JCOLL(3)
      DIMENSION IWK(1111),WK(12221),NVPAC(1111)
C       ....IWK dim. must be IVLVL.  WK dim. is 4*IVLVL+NMETH+NMITER
C     where, for our case, NMETH = 6*IVLVL and NMITER=IVLVL
      EXTERNAL MASTER,JACOB
      $SUB=$PRSUB('SOLVER ')
C                          Variables required by DGEAR
      INDEX=1
      MITER=2
      METH=2
C           ...these values for METH and MITER were selected
C           according to IMSL recommendation for stiff diff. eqn.
C           (MITER=2 would be even better, but the WK dim. grows
C           to IVLVL*(IVLVL+1), so will use the less-accurate
C           MITER=3.)
      TOL=1.0D-8
      T=0.0D00
      H=1.0D-10
      DNDT=0.0D00
C
C  Initialize variables for the writing loop  so that the first
C  10 values are written to files (using routine DUMP),
C  then every tenth point, for the next 100 calls to DGEAR,
C  then every hundredth point  up to the extent of the loop
C  (variable I4EVER, which is currently set to 10000).
C
      INC1 = 10000
      INC2 = 10
      INC3 = 100
      LAST1 = I4EVER
      LAST2 = 110
C
  901 WRITE(IOUT,*)  'At what extent of reaction should the calcula
     &  tion be stopped? <Default: 0.1>'
      READ(INP,*,END=902) FRACT
      GO TO 903
  902 IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 901
      ELSE
        CALL DFAULT(FRACT 0.1D0)
      ENDIF
  903 WRITE(ISAVE,*) FRACT
      FRCPCS=MIN(CONCA,CONCBC)/MAX(CONCA CONCBC)
      IF(CONCA.LT.CONCBC) THEN
        WRITE(IOUT,*)  'BC is in excess'
        NEND = CONCA
      ELSE
        WRITE(IOUT,*)  'BC is not in excess'
        NEND = CONCBC
      ENDIF
      WRITE(IOUT,*)  ' Ratio of concentration reagents:   FRCPCS
      WRITE(IOUT,*)  'Specified extent of reaction desired:   FRACT
      IVLVL = 0
      DO 100 V = 0,MAXV
         DO 100 J = 0,MAXJ(V)
            IVLVL = IVLVL + 1
            NVPAC(IVLVL) = NV(V,J)
  100 CONTINUE
      DO 150 V = 0  MAXV
        WRITE(VUNIT(V) '702)
        WRITE(VUNIT(V+12),702)
  701   FORMAT(3X  'V',3X,' J  12X, 'Time  12X,' NV(V,J)  12X 'NLUMP')
  702   FORMAT(3X,' V',3X,' J  12X 'Time  12X    DNV(V,J)  12X, DNLUMP')
  150 CONTINUE
      WRITE(21,*)    Time        ExtentRxn       dN/dT          R_phenom'
     &          Sum of y_v*k_v
      CALL DUMP(IVLVL,T,NVPAC,NEND)
      TEND = T + H
      CALL DGEAR(IVLVL,MASTER,JACOB,T,H,NVPAC,TEND,TOL,METH,MITER,
     &           INDEX,IWK,WK,IER)
      IF(IER.GT.128) GO TO 30
      CALL DUMP(IVLVL,T,NVPAC,NEND)
C
```

```
905   WRITE(IOUT,*)' Input the preferred time step size.
      WRITE(IOUT,*)'<Carriage return: time integrator routine decides.>
      READ(INP,* END=906) H
      INDEX=0
      GO TO 907
906   IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 905
      ELSE
C Dummy call to DFALTD to make sure we can read from INP the next time
        CALL DFALTD(H,H)
      ENDIF
907   WRITE(ISAVE,*) H
      DO 100 ICOUNT = 1,I4EVER
        TEND = DBLE(ICOUNT)/0.10000D0
        TEND = T + H
C Warning: index=3 causes dgear to work much more slowly.  Don't use
C unless you really want to see the result after each step.
C       INDEX=3
C                 ...will make DGEAR return after one step...
        CALL DGEAR(IVLVL,MASTER,JACOB,T,H,NVPAC,TEND,TOL,METH,MITER,
     &             INDEX,IWF,WF,IER)
        IF(IER.GT.128) GO TO 30
C       IF ( (ICOUNT.LT.LAST1 .AND. MOD(ICOUNT,INC1).EQ.0) THEN
C       IF ( (ICOUNT.LT.LAST1 .AND. MOD(ICOUNT,INC1).EQ.0) .OR.
C    &       (ICOUNT.LT.LAST2 .AND. MOD(ICOUNT,INC2).EQ.0) .OR.
C    &       (ICOUNT.GE.LAST2 .AND. MOD(ICOUNT,INC3).EQ.0) ) THEN
          CALL DUMP(IVLVL,T,NVPAC,NEND)
          IF((CONCBC-NTOT)/NEND.GE.FRACT) GO TO 210
C       ENDIF
100   CONTINUE
210   WRITE(IOUT,*) ' SOLVER ends after ',ICOUNT-1,' iterations.'
      WRITE(IOUT,*) ' Max. no. of iterations allowed = ',I4EVER
      WRITE(IOUT,*) ' Suggested TEND value=',T
      RETURN
30    CONTINUE
      WRITE(IOUT,*) ' Problems in DGEAR. IER = ',IER
      RETURN
      END
```

```
      SUBROUTINE MASTER (IVLVL,T,NVPAC,DNVPAC)
C
C The arguments of MASTER are set by IMSL specifications.
C (see DGEAR.)  MASTER evaluates the r.h.s.'s to be integrated
C by DGEAR.
C
      IMPLICIT DOUBLE PRECISION (B-H,P,R,T,X,Y,Z)
      IMPLICIT INTEGER (A,C-G,L-N,V)
      IMPLICIT CHARACTER*6 (S)
      LOGICAL*1 EXIST
      COMMON/MVIEW/MVU(3,0:10,0:20),MUU(3,0:10,0:20),
     1  MZU(3,0:10,0:20),MZD(3,0:10,0:20),MU(3,0:10,0:20),
     2  MV(3,0:10,0:20),MZD(3,0:10,0:20),MUU(3,0:10,0:20),
      COMMON/MASTR/KVUBIM(0:10,0:20)
      COMMON/PCP/NV(0:10,0:20),NTOT,CONCA,CONCBC,CONFM,SCALE(3),
      COMMON/COUNT/MAXV,MAXJ(0:20),MAXNV,  INFTR,  ICOLL
      DIMENSION DNV(0:10,0:20),
     &          NVPAC(IVLVL),DNVPAC(IVLVL),   CONC(3),
      EQUIVALENCE (CONC(1),CONCA)
      $SUB=PPSUB('MASTER')

C Since DGEAR requires a population vector but we want to
C work with a V,J population array we must unpack the array supplied
C by DGEAR (NV) at the beginning of MASTER and pack the array required
C by DGEAR at the end of MASTER (DNV)
      IVLVL = 0
      NTOT  = 0.D0
      DO 100 V = 0,MAXV
        DO 100 J = 0,MAXJ(V)
          IVLVL = IVLVL + 1
          NV(V,J)  = NVPAC(IVLVL)
          NTOT=NV(V,J)+NTOT
100   CONTINUE
      NVA = CONCA - (CONCBC - NTOT)
      DO 200 V=0,MAXV
        DO 200 J=0,MAXJ(V)
        DNVVJ=0.0D0
        NVVJ=NV(V,J)
        DO 300 IC= 1,ICOLL
          IF(EXIST(V-1,J,V,J)) THEN
            DNVVJ=NV(V-1,J)*KU03(IC,V-1,J)+DNVVJ
            DNVVJ=NVVJ*KD03(IC,V,J)  + DNVVJ
          ENDIF
          IF(EXIST(V-1,J+1,V,J)) THEN
            DNVVJ= NV(V-1,J+1)*KU03(IC,V-1,J+1)+DNVVJ
            DNVVJ= NVVJ*KU03(IC,V,J)  + DNVVJ
          ENDIF
          IF(EXIST(V-1,J-2,V,J)) THEN
            DNVVJ=NV(V-1,J-2)*KU03(IC,V-1,J-2)+DNVVJ
            DNVVJ  = NVVJ*KD03(IC,V,J)+ DNVVJ
          ENDIF
          IF(EXIST(V,J-2,V,J)) THEN
            DNVVJ= NV(V,J-2)*KU03(IC,V,J-2)  + DNVVJ
            DNVVJ= NVVJ*KD03(IC,V,J)  + DNVVJ
          ENDIF
          IF(EXIST(V+1,J,V,J)) THEN
            DNVVJ=NV(V+1,J)*KD03(IC,V+1,J)  + DNVVJ
            DNVVJ= NVVJ*KU03(IC,V,J)+DNVVJ
          ENDIF
          IF(EXIST(V+1,J+2,V,J)) THEN
            DNVVJ=NV(V+1,J+2)*KD03(IC,V+1,J+2)+DNVVJ
            DNVVJ= NVVJ*KU03(IC,V,J)  + DNVVJ
          ENDIF
          IF(EXIST(V+1,J-2,V,J)) THEN
            DNVVJ= NV(V+1,J-2)*KD03(IC,V+1,J-2)+DNVVJ
            DNVVJ= NVVJ*KU03(IC,V,J)+DNVVJ
          ENDIF
          IF(EXIST(V,J+2,V,J)) THEN
            DNVVJ= NV(V,J+2)*KD03(IC,V,J+2)  + DNVVJ
            DNVVJ= NVVJ*KU03(IC,V,J)  + DNVVJ
          ENDIF
          DNVVJ = DNVVJ*CONC(3)*COLL(IC)

300     CONTINUE
        DNVVJ=NVVJ*NVA*KVJBIM(V,J)  + DNVVJ
        DNV(V,J)=DNVVJ
200   CONTINUE
      IVLVL = 0
C Now we must pack the V,J matrix of derivatives into a vector.
      DO 400 V = 0,MAXV
        DO 400 J = 0,MAXJ(V)
          IVLVL = IVLVL + 1
          DNVPAC(IVLVL) = DNV(V,J)
400   CONTINUE
C     WRITE(6,*) 'NVPAC: ',(NVPAC(I),I=1,MIN(IVLVL,10))
C     WRITE(6,*) 'DNVPAC: ',(DNVPAC(I),I=1,MIN(IVLVL,10))
      RETURN
      END
```

```
      SUBROUTINE JACOB(IVLVL,T,NVPAC,PD)
C
C This (dummy) subroutine supplies the Jacobian matrix (if req'd)
C for the DGEAR subroutine.  Arguments are as specified in IMSL
C documentation for DGEAR.
C
      IMPLICIT CHARACTER*6 (S)
      INTEGER IVLVL
      DOUBLE PRECISION NVPAC     /1/,PD(IVLVL,IVLVL),T
      $SUB=PPSUB('JACOB')
      RETURN
      END
```

```
      SUBROUTINE DUMP(IVLVL,T,NVPAC,NENT)
C  It is not obvious that the sum of DNV(V,J) over the V and J
C  levels gives dN/dT.  The following example shows that this is
C  the case:
C  Given a 3-level system where 42 molecules/sec leave the
C  first level  4 to react  and 38 to go to the second level.
C  25 molecules/sec leave the third level--13 react and 15 fall
C  back to the second level.
C  Therefore    53 molecules/sec enter the second level and
C  10 from the third  4 from the second--or
C  (42-53-25) must be leaving the system.
C  That is  DNV(1)+DNV(2)+DNV(3) = DNDT.
C
      IMPLICIT DOUBLE PRECISION (B-H,P,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      COMMON/POP/NV(0:10,0:20),NTOT,CONCA,CONCBC,CONCM,JJ(11,3)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INKV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
      COMMON/MASTR/KVJBIM(0:20,0:20)
      DIMENSION DNV(0:10,0:20),NVPAC(231),DNVPAC(231)
      $SUB=$PRSUB('DUMP  ')
C Master will calculate the DNVPAC vector  given the NVPAC vector
      CALL MASTER(IVLVL,T,NVPAC,DNVPAC)
C      now unpack the DNVPAC  NVPAC vectors.
      IVLVL = 0
      DO 100 V = 0,MAXV
         DO 100 J = 0,MAXJ(V)
            IVLVL = IVLVL + 1
            DNV(V,J) = DNVPAC(IVLVL)
            NV(V,J)  = NVPAC(IVLVL)
100   CONTINUE
C     NTOT = 0.0D0
      DNDT = 0.0D0
      VVKV = 0.0D0
      DO 200 V = 0, MAXV
        NLUMP=0.0D0
        DO 201 J=0,MAXJ(V)
C          NTOT  = NV(V,J) + NTOT
           NLUMP=NV(V,J)+NLUMP
           VVKV=VVKV+NV(V,J)*KVJBIM(V,J)
C                          ... lumped' over J values
           WRITE (VUNIT(V),701) V, J, T, NV(V,J), NLUMP
 701       FORMAT(2I5,3(2X,D16.10))
 201    CONTINUE
        DNLUMP=0.0
        DO 202 J=0,MAXJ(V)
          DNDT = DNV(V,J) + DNDT
          DNLUMP=DNV(V,J)+DNLUMP
          WRITE(VUNIT(V+12),701) V, J, T  DNV(V,J),DNLUMP
 202    CONTINUE
 200  CONTINUE
      VVKV = VVKV/NTOT
      RVA = 70NCA - (CONCBC - NTOT)
      RPHEN = -DNDT/(NTOT*RVA)
      XTENT = (CONCBC-NTOT)/NEND
      WRITE(31,702) T,XTENT,  DNDT,  RPHEN,  VVKV
 702     FORMAT(5(2X,D10.4))
      WRITE(6,*) 'Time, Total Population:    ',T,NTOT
      RETURN
      END
```

```
      SUBROUTINE EIGEN (ICTR1)
      IMPLICIT DOUBLE PRECISION (B-H,P,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      COMPLEX*8 W(200),Z(200,200)
      DIMENSION  WF(40400)
C        ...this dim. based on MAXKLU=200: dim.=(MAXKLU+2)*MAXKLU
      COMMON/KVIB/KFUP(0:10,0:20), KFDN(0:10,0:20), KFUP0(0:10,0:20),
     1  KFDWN0(0:10,0:20), KFDWN0(0:10,0:20), KFDWN0(0:10,0:20),
     2  KGUP(0:10,0:20),KGDOWN(0:10,0:20),
     3  KLUMP(200,200)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INKV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
C
C EIGEN.  WF and Z are dummy arguments and are in the argument list
C above to allow the two arrays in question to have variable
C dimensions - otherwise the arrays would be inefficiently large.
C
      $SUB=$PRSUB('EIGEN  ')
      MAXKLU=200
      IF(IDEBUG.EQ.1) THEN
        WRITE (6,*)   order of KLUMP =  ,ICTR1
        ICTMIN=MIN(ICTR1,5)
        WRITE(6,*) 'KLUMP = ',((KLUMP (I,J), I=1, ICTMIN), J=1, ICTMIN)
      ENDIF
      CALL EIGRF(KLUMP,ICTR1,MAXKLU,2,W,Z,MAXKLU,WK,IER)
C ...pls see IMSLDLIB documentation for explanation of EIGRF arguments
      IF (IER.GT.128) THEN
        WRITE(IOUT,*)  Error in IMSL routine EIGRF.'
        CALL IBTERM(IER,$SUB)
      ENDIF
      WRITE(IOUT,*) ' Eigenvalues of the master equation are
     &  (complex numbers):'
      DO 10 I=1,ICTR1
        WRITE(IOUT,*) I, W(I)
 10   CONTINUE
      WRITE(IOUT,*)   The eigenvector matrix according to
     &  ' (row, column) in complex notation is:
      LROW = MIN(ICTR1,10)
      DO 200 M = 1,LROW
        WRITE(IOUT,*) '  ==================================='
        DO 200 L = 1,LROW
          WRITE(IOUT,*) M,L,Z(M,L)
 200  CONTINUE
      RETURN
      END
```

```
      SUBROUTINE TABOUT
C Tables created are suitable for input to a plotting routine
C or are TeX files suitable for incorporation into documents.
      IMPLICIT DOUBLE PRECISION (B-H,P,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      DIMENSION X(102),NV(0:10,0:20)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INKV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
      COMMON/PHYSCT/PLANCK,BOLTZ,PGAS,SCEXP1,PI,I4EVER
      $SUB=$PRSUB('TABOUT')
C For the moment the following is a subset
101   IFLAG=0
900   WRITE(IOUT,*)   For which V, J level do you wish to plot:
      WRITE(IOUT,*)  <default is 0,0 level.>
      READ(INP,*,END=901) VV,JJ
      GO TO 902
901   IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 900
      ELSE
        CALL DFALTI(VV,0)
        CALL DFALTI(JJ,0)
      ENDIF
902   WRITE(ISAVE,*) VV,JJ
      IF(VV.LT.0.OR.VV.GT.10) THEN
        WRITE(IOUT,*)  ' V level incorrectly specified: V must lie
     &    between 0 and 10 inclusive.'
        IFLAG=IFLAG+1
        IF(IFLAG.LE.1) GO TO 101
          CALL IBTERM(VV,$SUB)
      ENDIF
      IFLAG=0
      IF(JJ.LT.0.OR.JJ.GT.100) THEN
        WRITE(IOUT,*)  ' J level incorrectly specified: J must lie
     &    between 0 and 100 inclusive.'
        IFLAG=IFLAG+1
        IF(IFLAG.LE.1) GO TO 101
          CALL IBTERM(JJ,$SUB)
      ENDIF
      DO 200 I=1,I4EVER
        READ(VUNIT(VV),* ,END=201) X
        NV(VV,JJ)=X(JJ+2)
        WRITE(IOUT,*)  I,NV(VV,JJ)
200   CONTINUE
201   WRITE(IOUT,*)  ' If you want output for more V levels  type  1 '
      READ(INP,*,END=903)  IANS
      GO TO 904
903   IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
       GO TO 201
       ELSE
        CALL DFALTI(IANS,0)
      ENDIF
904   WRITE(ISAVE,*) IANS
      IF(IANS.EQ.1) GO TO 101
      RETURN
      END
```

```
      SUBROUTINE PLTOUT
C DISSPLA is the plotting program used.  Modifications needed to use
C a different plotting routine should be minor.
C The routine will eventually have the capability to output
C plots directly to the plotting device or to create a file suitable
C for incorporation into a larger document.
      IMPLICIT DOUBLE PRECISION (B-H,P,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INKV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
C For the moment a dummy routine.
      $SUB=$PRSUB('PLTOUT')
      WRITE(IOUT,*) ' PLTOUT has been called.
      RETURN
      END
```

```
      SUBROUTINE PRNTVJ(INT,DPREC)
      IMPLICIT DOUBLE PRECISION (B-H,P,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
C assumed size...does this save on memory?
      DIMENSION DPREC(0:10,*)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INKV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
C For the moment a dummy routine.
      $SUB=$PRSUB('PRNTVJ')
      RETURN
      END
```

```
      FUNCTION EDEL(I,J,K,L)
      IMPLICIT DOUBLE PRECISION (B-H,P,N-U,W-Z)
      IMPLICIT INTEGER (A,I-M,V)
      IMPLICIT CHARACTER*6 (S)
      COMMON/ENERGY/ELEV(0:10,0:10),  TEMP
C     $SUB=$PRSUB('EDEL  ')
      EDEL = ELEV(K,L) - ELEV(I,J)
      RETURN
      END
```

```
      FUNCTION $PRSUB(SNAME)
      IMPLICIT CHARACTER*6 ($)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INCV,VONLY,VUNIT(0:23)
      $SUB= $PRSUB
      IF (IDEBUG.EQ.1) WRITE(IOUT,701) $NAME
  701 FORMAT(' *** Routine ',A6,' called.')
      $PRSUB = $NAME
      RETURN
      END
```

```
      SUBROUTINE OPNFIL
      IMPLICIT INTEGER (A,C-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INCV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV
C  OPEN files on units 10 ---> 20 reserved for NV(V)
C                      40 ---> 50 reserved for DIM(V)
C                      21, 51     reserved for NTOT and ENTOT  resp.
C  This subroutine will eventually have machine-readable files
C  instead of formatted files (which is actually TABCUT's function)
C  but for ease of debugging we have kept these files being written
C  in an inefficient but humanly-readable way.
C
C  File must be opened for direct access.
C  RECL must be specified for direct access.
C  Records must be written direct access in order to be
C  read direct access.  Syntax of the READ cmd is READ(un,fmt,REC=rec)
C  First entry in each record is TIME; remaining entries are
C   pop. (der. pop.).  Each file has information about one
C   v level and its J sub-level/p.  Record length is not
C   assuming double precision (8 bytes).
C
      $SUB=$PRSUB( OPNFIL )
      DO 10 V = 0, MAXV
        WRITE(VUNIT(V),*)'     v     J       time                n(v,J)'
C       OPEN(VUNIT(V),FORM='UNFORMATTED' ACCESS='DIRECT',
C     &       RECL=(MAXJ(V)+1)*8)
        WRITE(VUNIT(V+12),*)'     v     J       time                dn(v,J)'
C       OPEN(VUNIT(V+12),FORM='UNFORMATTED' ACCESS='DIRECT',
C     &       RECL=(MAXJ(V)+1)*8)
   10 CONTINUE
      WRITE(8,*) '   time          total population '
      WRITE(9,*) '   time          rate of decrease of total population'
      RETURN
      END
```

```
      SUBROUTINE SIU(INTYPE,VALUE,UNIT)
C  **** FOR ALL CALLS TO SIU, MAKE SURE ITYPE IS CONSISTENT!
C  This subroutine converts units specified for the following
C  types to SI MKS units, according to the tens-digit of the
C  variable INTYPE:
C        (1) energy
C        (2) rate constants
C        (3) length  inverse length
C        (4) mass
C        (5) temperature
C        (6) population
C        (7) pressure
C        (8) concentration
C
C  For example  'CALL SIU(13,10.0,'CM')' means that the type  1
C  (energy, since integer(13/10)=1) has been selected.  The value
C  of 10.0 cm**-1 has been passed to subroutine SIU, which will
C  convert it to MKS units (namely  kJ/mol).
C
C  Conversion factors are based on W.H. Flygare(1978)
C
      IMPLICIT DOUBLE PRECISION(A-H,J-Z)
      IMPLICIT INTEGER*4 (I)
      IMPLICIT CHARACTER*6 ($)
      CHARACTER*2 UNIT
      INTEGER VUNIT,VONLY
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INCV,VONLY,VUNIT(0:23)
      PARAMETER (AVOGAD=6.02205D23)
      PARAMETER (EVTOJO=1.60219D-19*AVOGAD,  CMTOJO=1.98648D-23*AVOGAD,
     1           KCTOJO=6.95246D-21*AVOGAD,  CATOJO=6.95246D-24*AVOGAD,
     2           KJTOJO=1.0D-03              HATOJO=0.43598ISD-17*AVOGAD
     3           ERTOJO=1.0D-07)
      PARAMETER (MITOSE=1.0D0/60.0D0,        HRTOSE=1.0D0/3600.0D0)
      PARAMETER (BOTOME=5.29177D6D-11,       AGTOME=1.0D-10,
     1           NMTOME=1.0D-09,             CMTOME=1.0D-02,
     2           PMTOME=1.0D-12)
C  Values considered are for grams/mole  *not* per molecule.
      PARAMETER (AUTOKG=1.0D-03,GMTOKG=1.0D-03)
      PARAMETER (CETOKE=273.15D-0)
      PARAMETER (TOTOPA=133.289              ,  KPTOPA=1.0D+3,
     1           ATTOPA=1.013D+5,            LBTOPA=6891.16,
     2           BATOPA=1.0D5,               MMTOPA=TOTOPA)
C  Suggestion for future improvement:  for the symbol AB read in
C  for the units, this could in fact be AB, ab, Ab or aB.
C  We should construct a CASE function or subroutine, which
C  will automatically convert AB to upper case.
C  Then, the IF stmts may be shortened.
      $SUB=$PRSUB('SIU    ')
      ITYPE=    (INTYPE/10)

C  The reason that the number representing the type of conversion
C  needed must be integer-divided by 10
C  is that this number serves to store more than one
C  type of information: not only the kind of quantity  (e.g., energy
C  rate constant, etc.) but the physical
C  meaning associated with that number for purposes of reading in
C  values in READVJ and READIN.  For example there is more than one type
C  of rate constant (quantity type 2) that is read in but each must have
C  the same units-conversion applied and therefore the code associated
C  with those values (IREAD) is numbered 21, 22, 23, etc.
C  This has the drawback that the program will have to be changed
C  substantially if, for example, there were more than 10 types of rate
C  constants between which we wished to distinguish.
C  The advantage of this approach is that the program has consistency
C  in the codes used by subroutines READIN, READVJ, SIU, NOVICE
C   PRNTVJ, and routines that call these.
C
      IFLAG=0
  101 IF(ITYPE.EQ.1) THEN
        IF(UNIT.EQ.'EV'.OR.UNIT.EQ.'ev') THEN
          VALUE=VALUE*EVTOJO
          RETURN
        ELSEIF(UNIT.EQ.'CM'.OR.UNIT.EQ.'cm') THEN
          VALUE=VALUE*CMTOJO
          RETURN
        ELSEIF(UNIT.EQ.'JO'.OR.UNIT.EQ.'jo') THEN
          RETURN
        ELSEIF(UNIT.EQ.'KC'.Ok.UNIT.EQ.'kc') THEN
          VALUE=VALUE*KCTOJO
          RETURN
        ELSEIF(UNIT.EQ.'CA'.OR.UNIT.EQ. ca') THEN
          VALUE=VALUE*CATOJO
          RETURN
        ELSEIF(UNIT.EQ.'KJ'.OR.UNIT.EQ.'kj') THEN
          VALUE=VALUE*KJTOJO
          RETURN
        ELSEIF(UNIT.EQ.'HA'.OR.UNIT.EQ.'ha') THEN
          VALUE=VALUE*HATOJO
          RETURN
        ELSEIF(UNIT.EQ.'ER'.OR.UNIT.EQ.'er') THEN
          VALUE=VALUE*ERTOJO
          RETURN
        ELSE
          WRITE(IOUT,*) 'Units input for energy cannot be con',
     1     'verted.  Only EV, CM, KC, CA, KJ, JO, HA, ER ',
     2     'allowed.'
          GO TO 100
        ENDIF
      ELSEIF(ITYPE.EQ.2)THEN
        IF(UNIT.EQ.'SE'.OR.UNIT.EQ.'se') THEN
          RETURN
        ELSEIF(UNIT.EQ.'MI'.OR.UNIT.EQ.'mi') THEN
          VALUE=VALUE*MITOSE
          RETURN
        ELSEIF(UNIT.EQ.'HR'.OR.UNIT.EQ.'hr') THEN
          VALUE=VALUE*HRTOSE
        ELSE
          WRITE(IOUT,*) 'Units for rate constant cannot be con',
     1     'verted.  Only SE, MI or HR allowed.'
          GO TO 100
        ENDIF
      ELSEIF(ITYPE.EQ.3) THEN
```

```
      IF(UNIT.EQ.'BO'.OR.UNIT.EQ.'bo ') THEN
        VALUE=VALUE*BOTOME
        RETURN
      ELSEIF(UNIT.EQ.'AG'.OR.UNIT.EQ.'ag ') THEN
        VALUE=VALUE*AGTOME
        RETURN
      ELSEIF(UNIT.EQ.'NM'.OR.UNIT.EQ.'nm ') THEN
        VALUE=VALUE*NMTOME
        RETURN
      ELSEIF(UNIT.EQ.'CM'.OR.UNIT.EQ.'cm ') THEN
        VALUE=VALUE*CMTOME
        RETURN
      ELSEIF(UNIT.EQ.'PM'.OR.UNIT.EQ.'pm ') THEN
        VALUE=VALUE*PMTOME
        RETURN
      ELSEIF(UNIT.EQ.'ME'.OR.UNIT.EQ.'me ') THEN
        RETURN
      ELSE
        WRITE(IOUT,*) 'Units for distance cannot be converted.
     1     '  Only BO, ME, AG, NM, CM, PM allowed.'
        GO TO 100
      ENDIF
    ELSEIF( ITYPE.EQ.4) THEN
      IF(UNIT.EQ.'AU'.OR.UNIT.EQ.'au ') THEN
        VALUE=VALUE*AUTOKG
        RETURN
      ELSEIF(UNIT.EQ.'KG'.OR.UNIT.EQ.'kg ') THEN
        RETURN
      ELSEIF(UNIT.EQ.'GM'.OR.UNIT.EQ.'gm ') THEN
        VALUE=VALUE*GMTOKG
        RETURN
      ELSE
        WRITE(IOUT,*) 'Units for mass cannot be converted.',
     1       ' Only AU, KG, GM allowed.
        GO TO 100
      ENDIF
    ELSEIF(ITYPE.EQ.5) THEN
      IF(UNIT.EQ.'FA'.OR.UNIT.EQ.'fa ') THEN
        VALUE=(VALUE-32.0)*5.0/9.0 + CETOKE
        RETURN
      ELSEIF(UNIT.EQ.'CE'.OR.UNIT.EQ.'ce ') THEN
        VALUE=VALUE + CETOKE
        RETURN
      ELSEIF(UNIT.EQ.'KE'.OR.UNIT.EQ.'ke ') THEN
        RETURN
      ELSE
        WRITE(IOUT,*)  'Units for temperature cannot be converted.'
     1       ' Only FA, CE, KE allowed.'
        GO TO 100
      ENDIF
    ELSEIF(  ITYPE.EQ.6) THEN
      IF(UNIT.EQ.'MO'.OR.UNIT.EQ.'mo') THEN
        RETURN
      ELSEIF(UNIT.EQ.'NO'.OR.UNIT.EQ.'no ') THEN
        VALUE=VALUE /AVOGAD
        RETURN
      ELSE
        WRITE(IOUT,*) 'Units for population cannot be converted.
     1     '  Only MO (moles) or NO (absolute number) allowed.'
        GO TO 100
      ENDIF
    ELSEIF(  ITYPE.EQ.7) THEN
      IF(UNIT.EQ.'PA'.OR.UNIT.EQ.'pa') THEN
        RETURN
C **** conversion for torr to SI units?
      ELSEIF(UNIT.EQ.'TO'.OR.UNIT.EQ.'to') THEN
        VALUE=VALUE*TOTOPA
        RETURN
      ELSEIF(UNIT.EQ.'AT'.OR.UNIT.EQ.'at ') THEN
        VALUE=VALUE*ATTOPA
        RETURN
      ELSEIF(UNIT.EQ.'KP'.OR.UNIT.EQ.'kp ') THEN
        VALUE=VALUE*KPTOPA
        RETURN
      ELSE
        WRITE(IOUT,*) 'Units for pressure cannot be converted.
     1       ' Only PA (PAscal), KP (KiloPascal), AT (ATmosphere),
     2       ' or TO (TOrr) allowed.
        GO TO 100
      ENDIF
    ELSEIF(  ITYPE.EQ.8) THEN
      IF(UNIT.EQ.'MO'.OR.UNIT.EQ.'mo') THEN
        RETURN
C **** additional concentration units??
      ELSE
        WRITE(IOUT,*) 'Units for concentration cannot be converted.
     1     '  Only MO (moles) allowed.'
        GO TO 100
      ENDIF
    ELSE
      WRITE(IOUT,*) 'Internal error in SIU.'
      WRITE(IOUT,*) 'SI unit conversion not performed.
      WRITE(IOUT,*) 'Type is inappropriate.  ITYPE = ' ITYPE
      RETURN
    ENDIF
C*** spurious?cjc88/3/1  RETURN
  100 WRITE(IOUT,*) 'Please respecify unit.  ' UNIT
      READ(INP,*,END=901) UNIT
      GO TO 902
  901 IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 100
      ELSE
C The defaults are hard to specify here, because such a variety
C  of units is available.
        CALL DFALTD(VALUE,1.0D0)
        IF(ITYPE.EQ.1) CALL DFALTC(UNIT,'DE')
        IF(ITYPE.EQ.2) CALL DFALTC(UNIT,'SE')
        IF(ITYPE.EQ.3) CALL DFALTC(UNIT,'AG')
        IF(ITYPE.EQ.4) CALL DFALTC(UNIT,'GM')
        IF(ITYPE.EQ.5) CALL DFALTC(UNIT,'CE')
        IF(ITYPE.EQ.6) CALL DFALTC(UNIT,'MO')
        IF(ITYPE.EQ.7) CALL DFALTC(UNIT,'PA')
        IF(ITYPE.EQ.8) CALL DFALTC(UNIT,'MO')
      ENDIF
  902 WRITE(IDAVE,*) VALUE, UNIT
      WRITE(IOUT,*) VALUE UNIT
      IFLAG=IFLAG+1
      IF(IFLAG.LE.2) GO TO 901
      WRITE(IOUT,*)  Error in respecifying.  Program terminating.  ISUB
      STOP 7001
      END
```

```
      SUBROUTINE NOVICE(INTYPE)
C This subroutine prompts for units specified for the following
C types of SI MKS units, according to the tens-digit of ITYPE:
C       (1) energy
C       (2) rate constants
C       (3) distance, inverse distance
C       (4) mass
C       (5) temperature
C       (6) population
C       (7) pressure
C       (8) concentration
C
C For example,  CALL NOVICE(12)  means that the first  kind of
C unit (since int(12/10)=1 ) has been selected.
C
      IMPLICIT DOUBLE PRECISION(A-H,J-Z)
      IMPLICIT CHARACTER*6 (S)
      INTEGER*4 VUNIT, VONLY
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV VONLY VUNIT(6:22)
      ISUB='PRSUB(  NOVICE')
      ITYPE=    (INTYPE/10)
      IF(ITYPE.EQ.1) THEN
        WRITE(IOUT,*)  ' Units which are acceptable for energy are:
          WRITE(IOUT,*)  'EV or ev - electron volts'
          WRITE(IOUT,*)  'CM or cm - wavenumbers'
          WRITE(IOUT,*)  'KC or kc - kilocalories/mole
          WRITE(IOUT,*)  'CA or ca - calories/mole'
          WRITE(IOUT,*)  'KJ or kj - kilojoules/mole'
          WRITE(IOUT,*)  'JO or jo - joules/mole
          WRITE(IOUT,*)  'HA or ha - hartree (atomic units)
          WRITE(IOUT,*)  'ER or er - erg/mole
      ELSEIF(ITYPE.EQ.2)THEN
          WRITE(IOUT,*)  'Units for rate constant allowed are:
          WRITE(IOUT,*)  'SE or se - moles/sec'
          WRITE(IOUT,*)  'MI or mi - moles/minute
          WRITE(IOUT,*)  'HR or hr - moles/hour'
      ELSEIF(ITYPE.EQ.3)  THEN
          WRITE(IOUT,*)  'Units for distance or inverse distance are:'
          WRITE(IOUT,*)  'ME or me - meters'
          WRITE(IOUT,*)  'BO or bo - bohrs (atomic units)'
          WRITE(IOUT,*)  'AG or ag - angstroms'
          WRITE(IOUT,*)  'NM or nm - nanometers'
          WRITE(IOUT,*)  'CM or cm - centimeters'
          WRITE(IOUT,*)  'PM or pm - picometers'
      ELSEIF( ITYPE.EQ.4) THEN
          WRITE(IOUT,*)  'Units for mass allowed are:'
          WRITE(IOUT,*)  'AU or au - atomic mass units'
          WRITE(IOUT,*)  'KG or kg - kilograms'
          WRITE(IOUT,*)  'GM or gm - grams'
      ELSEIF(ITYPE.EQ.5) THEN
          WRITE(IOUT,*)  'Units for temperature allowed are:
          WRITE(IOUT,*)  'FA or fa - Fahrenheit'
          WRITE(IOUT,*)  'CE or ce - Celsius'
          WRITE(IOUT,*)  'KE or ke - Kelvin'
      ELSEIF(  ITYPE.EQ.6) THEN
          WRITE(IOUT,*)  'Units for population are:
          WRITE(IOUT,*)  'MO or mo - moles'
          WRITE(IOUT,*)  'NO or no - number of molecules'
      ELSEIF(  ITYPE.EQ.7) THEN
          WRITE(IOUT,*)  'Units for pressure are:'
          WRITE(IOUT,*)  'PA or pa - pascal'
          WRITE(IOUT,*)  'TO or to - torr (cm Hg)'
          WRITE(IOUT,*)  'AT or at - atmosphere'
          WRITE(IOUT,*)  'KP or kp - kilopascal'
      ELSEIF(  ITYPE.EQ.8) THEN
          WRITE(IOUT,*)  'Units for concentration are:'
          WRITE(IOUT,*)  'MO or mo - molar'
      ELSE
        WRITE(IOUT,*)  'Internal error.'
        WRITE(IOUT,*)  'Type is inappropriate.  ITYPE = ' ITYPE
        CALL ISTERM(ITYPE,ISUB)
        RETURN
      ENDIF
      WRITE(IOUT,*)
      RETURN
      END
```

```
      SUBROUTINE READIN(IREAD,DATA)
      IMPLICIT DOUBLE PRECISION (B-H,R,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      LOGICAL*1 $YVAL(0:20)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IPRBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,INPTS,ICOLL
      DIMENSION YVALUE(0:20),DATA(0:20,0:20)
      CHARACTER*2 UNIT
      CHARACTER*70 $HEADR
C
C  This subroutine is flagged, according to what kind of values
C   will be read in:
C
C    IREAD                        UNIT FOR READING IN FILE
C       11 -- v, E(v,J=0)                1
C       21 -- v, kup(v,0)                2
C       24 -- v, kbim(v,0)               3
C       61 -- v, nv(v,0)                 4
C
C  Please see the general comment in the main program regarding
C  the IREAD numbering convention.
C
C  Any data file consisting of triples (e.g. v,J, E(v,J)) is read in
C  through subroutine READVJ.
C
      $SUB=$PRSUB('READIN')
      IFIT=0
      IFIRST=0
C  Files for read-in data are assigned to units as noted
      IF(IREAD.EQ.11) INDAT=1
      IF(IREAD.EQ.21) INDAT=2
      IF(IREAD.EQ.24) INDAT=3
      IF(IREAD.EQ.61) INDAT=4
C Is MAXV already known?  If not known, it will still be zero,
C  and within this subroutine, we will have to determine MAXV by a
C  standard comparison technique.  IFIRST acts as a flag,
C  to signal that MAXV must be found.
      IF(MAXV.EQ.0) IFIRST=1
      DO 100 J=0,20
        YVALUE(J)=0.0D0
        $YVAL(J)=.FALSE.
  100 CONTINUE
      IF(IREAD.EQ.11) THEN
  902   WRITE(IOUT,*) ' What is the unit for energy in the file?'
        WRITE(IOUT,*) ' <Default:  "kj">'
        IF(INOV.EQ.1) CALL NOVICE(IREAD)
        READ(INP,*,END=900) UNIT
        GO TO 901
  900   IF(IOUT.EQ.ISUP) THEN
          INP=5
          IOUT=6
          GO TO 902
        ELSE
          CALL DFALTC(UNIT,'KJ')
        ENDIF
  901   WRITE(ISAVE,*) UNIT
        $HEADR='     V              E(V)     '
      ELSE IF(IREAD.EQ.21) THEN
  903   WRITE(IOUT,*) ' What is the unit for k(v-->v+1) rate '
     &   'constants in the file?'
        WRITE(IOUT,*)  ' <Def.:  "se">'
        IF(INOV.EQ.1) CALL NOVICE(IREAD)
        READ(INP,*,END=9041 UNIT
        GO TO 905
  904   IF(IOUT.EQ.ISUP) THEN
          INP=5
          IOUT=6
          GO TO 903
        ELSE
          CALL DFALTC(UNIT,'SE')
        ENDIF
  905   WRITE(ISAVE,*) UNIT
        $HEADR='     V              K(V-->V+1)'
      ELSE IF(IREAD.EQ.24) THEN
  906   WRITE(IOUT,*) ' What is the unit for kbim(v) in the file?'
        WRITE(IOUT,*) ' <Def.:  "se">'
        IF(INOV.EQ.1) CALL NOVICE(IREAD)
        READ(INP,*,END=907) UNIT
        GO TO 9''
  907   IF(IOUT.EQ.ISUP) THEN
          INP=5
          IOUT=6
          GO TO 906
        ELSE
          CALL DFALTC(UNIT,'SE')
        ENDIF
  908   WRITE(ISAVE,*) UNIT
        $HEADR='     V              KBIM(V)'
      ELSE IF(IREAD.EQ.61) THEN
  909   WRITE(IOUT,*) ' What is the unit used for population?'
        WRITE(IOUT,*) ' <Def.:  "mc  ">'
        IF(INOV.EQ.1) CALL NOVICE(IREAD)
        READ(INP,*,END=910) UNIT
        GO TO 911
  910   IF(IOUT.EQ.ISUP) THEN
          INP=5
          IOUT=6
          GO TO 909
        ELSE
          CALL DFALTC(UNIT,'mc')
        ENDIF
  911   WRITE(ISAVE,*) UNIT
        $HEADR='     V              NV(V)'
      ELSE
        CALL IBTERM(IREAD,$SUB)
      ENDIF
C  As the data are read in, they are converted into s.i. units.
C  Data conversion in SIU depends on whether data are energy (IREAD/10
C  = 1) or rate constants(IREAD/10 = 2) or population (IREAD/10 = 6).
C
C  The first value of each pair must by definition be an integer.
C  This loop will determine the size of the array and automatically
C  arrange the points in the array according to the value of V.
C
C  The array $yval is intended simply to denote which locations in
C  the array are filled(.true.)  and which are not (.false.).
        VINMAX=0
        DO 200 I=0,20
          READ(INDAT,*,END=300) VIN,DATAIN
          IF($YVAL(VIN)) THEN
            WRITE(6,*) 'Duplicate in array at location  ',VIN,
     &        '.  Most recent value supplied will be used.'
          ELSE
            $YVAL(VIN)=.TRUE.
          ENDIF
          IF(VIN.GT.VINMAX) VINMAX=VIN
          CALL SIU(IREAD,DATAIN,UNIT)
          YVALUE(VIN) = DATAIN
  200   CONTINUE
C
C    There are IDATA x,y pairs in the
C  array, and the counters go from 0 to IDATA-1.
C
  300 WRITE(IOUT,*) I,'  data pairs read in.'
        If this is the first call to READIN (i.e. MAXV has not been
        set beyond zero), then the max. v value read in will become
        MAXV.  Otherwise, check to see if the max. v read in
        corresponds to the MAXV the user has previously specified.
      IF(IFIRST.EQ.1) THEN
        MAXV=VINMAX
      ELSE IF(MAXV.GT.VINMAX) THEN
        WRITE(IOUT,*) '  File read in does not go as high as max. v '
     &                'value expected:  ',MAXV,VINMAX
        WRITE(IOUT,*) '  Remaining values will be extrapolated.'
      ELSE IF(MAXV.LT.VINMAX) THEN
        WRITE(IOUT,*) '  File read in goes beyond the max. v '
     &                'value expected:  ',MAXV,VINMAX
      ENDIF
C
C  Now, we must check to see if any v levels are missing, and then
C   interpolate to obtain the missing value by calling an
C   IMSL subroutine, which is accessed through FITTR1.
C   (The '1' in the subroutine names refers to the one-dimensional
C   cubic spline function.)
C
      DO 500 I=0, MAXV
        IF (.NOT.$YVAL(I)) IFIT = 1
  500 CONTINUE
C  There are two types of one-dimensional fitting subroutines:
C   (1) a smoothing fitter ICSSCV
C   (2) an interpolating fitter ICSCCU
C  The routine called depends on how closely the user wants the
C  fitting function to "stick" to the data.
C  The most accurate data should interpolate; more questionable
C  points may have an improved fit by allowing the fit to smooth the
C  data.  (To be decided by user.)
      IFLAG=0
      IF (IFIT.NE.0) THEN
  930   WRITE(IOUT,*)'Some points are missing from data file.'
        WRITE(IOUT,*)'Would you like the data to be smoothed (1) ,
     &   ' or interpolated (0)* ?
        READ(INP,*,END=931) ANSMO
        GO TO 932
  931   IF(IOUT.EQ.ISUP) THEN
          INP=5
          IOUT=6
          GO TO 930
        ELSE
          CALL DFALTI(ANSMO,0)
        ENDIF
  932   WRITE(ISAVE,*) ANSMO
        CALL FITTR1($YVAL,YVALUE,ANSMO)
      ELSE
  933   WRITE(IOUT,*)'No points are missing from the data     .'
        WRITE(IOUT,*)'Would you like the data to be smoothed anyway?'
        WRITE(IOUT,*)' 1 = yes; 0 = no* '
        READ(INP,*,END=934) ANSMO
        GO TO 935
  934   IF(IOUT.EQ.ISUP) THEN
          INP=5
          IOUT=6
          GO TO 933
        ELSE
          CALL DFALTI(ANSMO,0)
        ENDIF
  935   WRITE(ISAVE,*) ANSMO
        IF(ANSMO.EQ.1) CALL FITTR1($YVAL,YVALUE,1)
      ENDIF
      WRITE(IOUT,*) $HEADR
      DO 600 I=0,MAXV
        DATA(I,0)=YVALUE(I)
        WRITE(IOUT,*) I,DATA(I,0)
  600 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE FITTR1(SYVAL,YVALUE,ANSMO)
      IMPLICIT DOUBLE PRECISION (B-H,K,N-O,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      LOGICAL*1 $YVAL(0:20)
      PARAMETER (IYSIZE=20)
      DIMENSION XIN(0:20),YIN(0:20),XOUT(0:20),YOUT(0:20),
     & YVALUE(0:20),SPLCOF(0:20,3),WK(31108)
C&& We need to document calls for wk size.
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,INPTS,ICOLL
C  There are two types of one-dimensional fitting subroutines:
C    (1) a smoothing fitter ICSSCV
C    (2) an interpolating fitter ICSCCU
C  The routine called depends on how closely the user wants the
C  fitting function to "stick" to the data.
      $SUB = 'FITTR1'
      ICOUNT=0
      IFIT=0
      IJOB=1
      IERR=0
C Build the 'in' arrays  which will be
C  used to create the spline coefficients.
      DO 100 I=0,IYSIZE
        IF($YVAL(I)) THEN
          XIN(ICOUNT)=DBLE(I)
          YIN(ICOUNT)= YVALUE(I)
          ICOUNT=ICOUNT+1
        ENDIF
        XOUT(I)=DBLE(I)
  100 CONTINUE
      IF(ICOUNT.GE.4.AND.ANSMO.EQ.1) THEN
        CALL ICSSCV(XIN,YIN,ICOUNT,YOUT,SPLCOF,IYSIZE+1,IJOB,WK,IERR)
      ELSE IF(ICOUNT.GE.2 .AND. ANSMO .EQ.0) THEN
        CALL ICSCCU(XIN,YIN,ICOUNT,SPLCOF,IYSIZE+1,IERR)
      ELSE
        WRITE(IOUT,*) 'Too few points for fitting.'
      ENDIF
      IF(IERR.GT.128) CALL IBTERM(IERR,$SUB)
C  Now, use the spline coefficients to find the points needed.
C  (We will recalculate the y-values of even those points
C  whose input values are known, since this is easier than sorting
C  the haves from the have-nots.)
C  For now, we evaluate the whole array.  If this proves to be other
C than harmless (extrapolation goes wild, for example) then the
C 2nd-last parameter in the call to icsevu will have to be changed
C to limit the extent of the extrapolation.
      CALL ICSEVU(XIN,YIN,ICOUNT,SPLCOF,IYSIZE+1,XOUT,YVALUE,IYSIZE+1
     &            IERR)
      IF(IERR.GT.128) CALL IBTERM(IERR,$SUB)
      RETURN
      END
```

```
      SUBROUTINE READVJ(IREAD, DATA)
      IMPLICIT DOUBLE PRECISION (B-H,P,R-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      LOGICAL*1 $XYVAL(0:10,0:20)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,INPTS,ICOLL
      DIMENSION XYVAL(0:10,0:20), DATA(0:10,0:20), JINMAX(0:10)
      CHARACTER*2 UNIT
      CHARACTER*70 $HEADR
      $SUB='PRSUB('READVJ')
      IFIT=0
      IFIRST = 0
C
C This subroutine is similar to READIN, except that it reads in
C  a three-dimensional array.
C Please see the general comment in the main program regarding
C the IREAD numbering convention.
C
C   IREAD                        UNIT FOR READING IN FILE
C     12 -- v, J, E(v,J)                1
C     25 -- v, J, kbim(v,J)             3
C     62 -- v, J, N(v,J)                4
C
C  Files for read-in data are assigned to units as noted
C
      IF(IREAD.EQ.12) INDAT=1
      IF(IREAD.EQ.25) INDAT=3
      IF(IREAD.EQ.62) INDAT=4
C Determine if MAXV has been set or if the current call to this
C routine should set it AND the MAXJ array.
      IF (MAXV.EQ.0) IFIRST = 1
C
C redimension here
      DO 100 V=0,10
        DO 100 J=0,20
          XYVAL(V,J)=0.0D0
          $XYVAL(V,J) = .FALSE.
 100  CONTINUE
      IF(IREAD.EQ.12) THEN
 900    WRITE(IOUT,*)  ' What is the unit for energy in the file?
        WRITE(IOUT,*)  ' <default: ''kj'' >'
        IF(INOV.EQ.1) CALL NOVICE(IREAD)
        READ(INP,*,END=901) UNIT
        GO TO 902
 901    IF(IOUT.EQ.ISUP) THEN
          INP=5
          IOUT=6
          GO TO 900
        ELSE
          CALL DFALTC(UNIT,'kj')
        ENDIF
 902    WRITE(ISAVE,*) UNIT
        $HEADR = '    V     J        E(V,J)
      ELSEIF(IREAD.EQ.25) THEN
 910    WRITE(IOUT,*)  ' What are
     &                 'the units for the rate constants in the file?
        WRITE(IOUT,*)  ' <default: ''se'' >'
        IF(INOV.EQ.1) CALL NOVICE(IREAD)
        READ(INP,*,END=911) UNIT
        GO TO 912
 911    IF(IOUT.EQ.ISUP) THEN
          INP=5
          IOUT=6
          GO TO 910
        ELSE
          CALL DFALTC(UNIT,'se')
        ENDIF
 912    WRITE(ISAVE,*) UNIT
        $HEADR = '    V     J        kbim(v,J)
      ELSEIF(IREAD.EQ.62) THEN
 920    WRITE(IOUT,*)  ' What are
     &                 'the units for population in the file?
        WRITE(IOUT,*)  ' <default: ''no'' >'
        IF(INOV.EQ.1) CALL NOVICE(IREAD)
        READ(INP,*,END=921) UNIT
        GO TO 922
 921    IF(IOUT.EQ.ISUP) THEN
          INP=5
          IOUT=6
          GO TO 920
        ELSE
          CALL DFALTC(UNIT,'no')
        ENDIF
 922    WRITE(ISAVE,*) UNIT
        $HEADR = '    V     J        N(v,J)
      ELSE
        CALL IBTERM(IREAD,$SUB)
      ENDIF
C  As the data are read in, they are converted into S.I. units.
C  Data conversion in SIU depends on whether data are energy (IREAD/10
C  = 2) or rate constants(IREAD/10 = 3) or population (IREAD/10 = 6).
C
C  The first two values of each triplet must by definition be integers.
C  This loop will determine the size of the array and automatically
C  arrange the points in the array according to the value of v and J.
C
C  The array $xyval is intended simply to denote which locations in
C  the array are filled(.true.), and which are not (.false.).
      VINMAX = 0
      DO 150 I = 0, 10
        JINMAX(I) = 0
 150  CONTINUE
      DO 200 I=0 1110
        READ(INDAT,*,END=300) VIN, JIN, DATAIN
        IF($XYVAL(VIN, JIN)) THEN
          WRITE(6,*)  'Duplicate in array at location ', VIN,', ', JIN
     &          ,'  Most recent value supplied will be used.'
        ELSE
          $XYVAL(VIN  JIN)=.TRUE.
        ENDIF
        IF (VIN.GT.VINMAX) THEN
```

```
            VINMAX = VIN
            IF (JIN.GT.JINMAX(VIN)) JINMAX(VIN) = JIN
          ENDIF
          CALL SIU(IREAD,DATAIN,UNIT)
          XYVAL(VIN,JIN) = DATAIN
200     CONTINUE
300     WRITE(IOUT,*) I,' data triplets read in.'
        IF (IFIRST.EQ.1) THEN
          MAXV = VINMAX
          DO 250 I = 0, 10
            MAXJ(I) = JINMAX(I)
250       CONTINUE
        ELSE
          IF (MAXV.GT.VINMAX) WRITE(IOUT,*)' Interpolation coming up.
     &     ' V values between ',MAXV,' and ',VINMAX,' are missing.'
          IF (MAXV.LT.VINMAX) WRITE(IOUT,*)' Warning...some input V
     &      ' values are beyond the maximum (',MAXV
     &      ,') for this run.  They will be ignored.'
          DO 360 IV = 0, MAXV
            DO 260 IJ = 0, MAXJ(IV)
              IF (MAXJ(IV).GT.JINMAX(IV)) WRITE(IOUT,*)
     &         ' Interpolation coming up.  J values between ',MAXJ(IV)
     &         ,' and ',JINMAX(IV),' are missing.'
              IF (MAXJ(IV).LT.JINMAX(IV)) WRITE(IOUT,*)
     &         ' Warning...some input J value are beyond the maximum '
     &         ,'(',MAXJ(IV),') for this run.  They will be ignored.'
260         CONTINUE
360       CONTINUE
        ENDIF
C
C  Now  we must check to see if any V J levels are missing, and then
C   interpolate to obtain the missing value by calling an
C   IMSL subroutine, which is accessed through FITTR2.
C   (The '2' in the subroutine name refers to the two-dimensional
C   cubic spline function.)
C
        DO 500 IV=0, MAXV
          DO 500 IJ=0, MAXJ(IV)
            IF (.NOT.SXYVAL(IV,IJ)) IFIT = 1
 500    CONTINUE
C  There are two types of two-dimensional fitting subroutines:
C    (1) a smoothing fitter ? analogous to ICSSCV
C    (2) an interpolating fitter ? analogous to ICSCCU
C  The routine called depends on how closely the user wants the
C  fitting function to 'stick' to the data.
C  The most accurate data should interpolate; more questionable
C  points may have an improved fit by allowing the fit to smooth the
C  data.  (To be decided by user.)
        IFLAG=0
        IF (IFIT.NE.0) THEN
930       WRITE(IOUT,*) Some points are missing from data file.'
          WRITE(IOUT,*)'Would you like the data to be smoothed (1)'
     &      ' or interpolated (0)' ?'
          READ(INP,*,END=931) ANSMO
          GO TO 932
 931      IF(IOUT.EQ.ISUP) THEN
            INP=5
            IOUT=6
            GO TO 930
          ELSE
            CALL DFALTI(ANSMO,0)
          ENDIF
 932      WRITE(ISAVE,*) ANSMO
          CALL FITTR2(SXYVAL,XYVAL,ANSMO)
        ELSE
 933      WRITE(IOUT,*) No points are missing from the data file.'
          WRITE(IOUT,*)'Would you like the data to be smoothed anyway?
          WRITE(IOUT,*)' 1 = yes; 0 = no'
          READ(INP,*,END=934) ANSMO
          GO TO 935
 934      IF(IOUT.EQ.ISUP) THEN
            INP=5
            IOUT=6
            GO TO 933
          ELSE
            CALL DFALTI(ANSMO,0)
          ENDIF
 935      WRITE(ISAVE,*) ANSMO
          IF(ANSMO.EQ.1) CALL FITTR2(SXYVAL,XYVAL,1)
        ENDIF
        WRITE(IOUT, *) SHEADR
        DO 600 IV=0, MAXV
          DO 600 IJ=0,MAXJ(IV)
            DATA(IV,IJ)=XYVAL(IV,IJ)
            WRITE(IOUT,*) IV, IJ, DATA(IV,IJ)
600     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE FITTR2(SXYVAL,XYVAL,ANSMO)
      IMPLICIT DOUBLE PRECISION (B-H,P-N-U,W-Z)
      IMPLICIT INTEGER (A-I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      LOGICAL*1 SXYVAL(0:10,0:20)
      PARAMETER (IXSIZE=10, IYSIZE=20)
      DIMENSION XIN(0:231),YIN(0:231),XINT(0:10),YINT(0:20)
     & ,ZIN(0:231),ZINT(0:10,0:20)
     & ,XYVAL(0:10,0:20),IWK(35552),WK(6666)
C&& document the iwk and wk array sizes.
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN, INPTS, ICOLL
      SSUB = 'FITTR2'
      ICINP=0
C                 counter for no. of points with known values
      ICINT=0
      IERR=0
      LENGTH=(IXSIZE+1)*(IYSIZE+1)
C Build the  in  arrays, XIN, YIN and ZIN  which will be
C  used to create the spline coefficients.
      DO 100 I=0,IXSIZE
      XINT(I)=DBLE(I)
      DO 100 J=0,IYSIZE
        IF(I.EQ.0) YINT(J)=DBLE(J)
        IF(SXYVAL(I,J)) THEN
          XIN(ICINP)=DBLE(I)
          YIN(ICINP)=DBLE(J)
          ZIN(ICINP)= XYVAL(I,J)
          ICINP=ICINP+1
        ENDIF
 100  CONTINUE
      IYMAX=0
C                       find max. of all the JMAX
      DO 300 V=0, MAXV
        IF(MAXJ(V).GT.IYMAX) IYMAX=MAXJ(V)
300   CONTINUE
C
C  (We will recalculate the y-values of even those points
C  whose input values are known, since this is easier than sorting
C  the haves from the have-nots.)
C  For now  we evaluate the whole array.  If this proves to be other
C than harmless (extrapolation goes wild, for example) then we will
C have to limit the extent of the extrapolation.
      IF(ICINP.GE.4) THEN
        CALL IQHSCV(XIN,YIN,ZIN,LENGTH,XINT,MAXV+1,YINT,IYMAX+1,
     &              ZINT,IXSIZE+1,IWK,WK,IER)
      ELSE
        WRITE(IOUT,*) 'Too few points for fitting.'
      ENDIF
C                       write out the matrix found by interpolation
      WRITE(6,*)
      WRITE(6,99)
99    FORMAT(12X,'V=0',7X,'V=1',7X,'V=2',7X,'V=3',7X,'V=4',7X,'V=5'
     &       7X,'V=6',7X,'V=7',7X,'V=8',7X,'V=9',7X,'V=10')
      WRITE(6,*)
      DO 400 J=0,IYMAX
        WRITE(6,98) J,(ZINT(L,J),L=0,MAXV)
        WRITE(6,98) J,(XYVAL(L,J),L=0,MAXV)
98      FORMAT(2X, I3,2X,11E10.4)
        WRITE(6,*)
400   CONTINUE
      IF(IERR.GT.128) CALL IETERM(IERR,SUB)
      RETURN
      END
```

```
      SUBROUTINE DFALLD(VAR,VALUE)
      DOUBLE PRECISION VAR, VALUE
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      VAR = VALUE
      CLOSE (INP)
      RETURN
      END
```

```
      SUBROUTINE DFALTC(VAR,VALUE)
      CHARACTER*1  VAR, VALUE
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      VAR = VALUE
      CLOSE (INP)
      RETURN
      END
```

```
      SUBROUTINE DFALTI(VAR,VALUE)
      INTEGER  VAR, VALUE
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      VAR = VALUE
      CLOSE (INP)
      RETURN
      END
```

```
      SUBROUTINE IBTERM(IANS,SNAME)
      IMPLICIT INTEGER (I)
      IMPLICIT CHARACTER*6 (S)
C A program termination for incorrect integer input.
      $SUB=$PRSUB('IBTERM')
      WRITE(6,*)  Specification for value in subroutine ',SNAME,
     &    ' is invalid.    IANS
      WRITE(6,*)  Program terminating in ',SNAME,'.'
      STOP  3601
      END

      SUBROUTINE ABTERM(ANS,SNAME)
      IMPLICIT CHARACTER*6 (S)
C A program termination for incorrect real number input.
      DOUBLE PRECISION ANS
      $SUB=$PRSUB('ABTERM')
      WRITE(6,*)  Specification for value in subroutine ',SNAME,
     &    ' is invalid.    ANS
      WRITE(6,*)  Program terminating in ',SNAME,'.'
      STOP  3701
      END
```

```
      SUBROUTINE DUMP(IVLVL,T,NVPAC,NEND,XTENT)
C  It is not obvious that the sum of DNV(V,J) over the V and J
C  levels gives dN/dT.  The following example shows that this is
C   the case:
C   Given a 3-level system where 42 molecules/sec leave the
C   first level, 4 to react, and 38 to go to the second level.
C   25 molecules/sec leave the third level--10 react and 15 fall
C   back to the second level.
C   Therefore    53 molecules/sec enter the second level and
C   10 from the third, 4 from the second--or
C   (42-53+25) must be leaving the system.
C   That is, DNV(1)+DNV(2)+DNV(3) = DNDT.
C
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      REAL SDUMMY
      COMMON/GEAR/DUMMY(48),SDUMMY(4),IDUMMY(38,
      COMMON/POP/NV(0:10,0:20),NTOT,CONCA,CONCBC,CONCM,JCOLL(3)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
      COMMON/MASTR/KVJBIM(0:10,0:20)
      COMION/UNDERF/IUNDER
      DIMENSION DNV(0:10,0:20),NVPAC(231),DNVPAC(231)
C     $SUB=$PRSUB('DUMP')
      CALL INCCT
C Master will calculate the DNVPAC vector, given the NVPAC vector
      CALL MASTER(IVLVL,T,NVPAC,DNVPAC)
C      now unpack the DNVPAC  NVPAC vectors.
      IVLVL = 0
      DO 100 V = 0,MAXV
         DO 100 J = 0,MAXJ(V)
            IVLVL = IVLVL + 1
            DNV(V,J)  = DNVPAC(IVLVL)
            NV(V,J)   = NVPAC(IVLVL)
100   CONTINUE
C     NTOT = 0.0D0
      DNDT = 0.0D0
      YVKV = 0.0D0
      DO 200 V = 0, MAXV
        NLUMP=0.0D0
        DO 201 J=0,MAXJ(V)
C         NTOT  = NV(V,J) + NTOT
          NLUMP=NV(V,J)+NLUMP
          YVKV=YVKV+NV(V,J)*KVJBIM(V,J)
C                          ...'lumped' over J values
C         WRITE(VUNIT(V),701) V, J, T, NV(V,J),NLUMP
 701      FORMAT(2I5,3(2X,D16.10))
 201    CONTINUE
        DNLUMP=0.0
        DO 202 J=0,MAXJ(V)
          DNDT = DNV(V,J) + DNDT
          DNLUMP=DNV(V,J)+DNLUMP
C         WRITE(VUNIT(V+12),701) V, J, T, DNV(V,J),DNLUMP
 202    CONTINUE
 200  CONTINUE
      YVKV = YVKV/NTOT
      NVA = CONCA + CONCBC + NTOT
      KPHEN = -DNDT/NTOT/NVA
C     WRITE(6,'(1P,7(2X,D10.3),0P)')YVKV,KPHEN,DNV(0,0),DNV(1,0)
C    1  DNV(2,0),DNV(3,0),DNV(4,0)
C xtent is amount reacted/amount available to react...
      XTENT = (CONCBC-NTOT)/NEND
      WRITE(21,702) XTENT, KPHEN, DUMMY(8), IDUMMY(6),
     1 IDUMMY(7)  IUNDER
 702      FORMAT(1P,3(D14.7),3(I10))
      WRITE(22,703) NV(0,0)/NTOT, N,(1,0)/NTOT
     1NV(2,0)/NTOT, NV(3,0)/NTOT  NV(4,0)/NTOT
 703  FORMAT(1P,D20.13,4(D15.8))
C      IF(IRESET.EQ.1)  IRESET = 2
C      IF(XTENT.GT.RESET.AND.IRESET.EQ.0)  THEN
C      IRESET = 1
C      WRITE(6,*) 'Resetting tol...'
C      ENDIF
C      IF(IRESET.EQ.2   IRESET = 4
C      IF(XTENT.GT.1.E-1.AND.IRESET.EQ.2)  THEN
C      IRESET = 3
C      WRITE(6,*)  'Resetting tol...'
C      ENDIF
C     IF(IRESET.EQ.5)  IRESET = 6
C     IF(XTENT.GT.1.E-3.AND.IRESET.EQ.4)  THEN
C     IRESET = 5
C     WRITE(6,*) 'Resetting tol...'
C     ENDIF
C     WRITE(6,*) 'Time, Total Population: ', T,NTOT
      RETURN
      END
```

```
      SUBROUTINE INIT
C This subprogram initializes the physical constants used.
C  The constants are given for SI units.
C% NF Document default changeability.
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      REAL*8 ARAD,AMASS
      DOUBLE PRECISION AVOGAD
      CHARACTER*36 $SPGEN(3)
      COMMON/SPCIES/ISPEC,$SPEC(6,3),
      COMMON/PHYSCT/CLANCK,PBOLTZ,RGAS,SCEXP1,PLIGHT,PI,I4EVEP
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
      COMMON/POP/NV(0:10,0:20),NTOT,CONCA,CONCBC,CONCM,JCOLL(3),
      COMMON/ENERGY/ELEV(0:11,0:10),TEMP
      COMMON/MASS/EMASS,CMASS,AMASS,FELMAS,RRAD,CRAD,ARAD,RE
      PARAMETER (AVOGAD=6.022045D23)
      EQUIVALENCE($SPGEN(1),$SPEC(1,1))
C  The long strings which are specific for the various kinds of
C  collision are written into a *36 character string, which is
C  then effectively broken into 6 - *6 character strings by means
C  of equivalencing with an array $SPEC of dimension 6.
C  The string-length is chosen because CHARACTER*6 is conventionally
C  used in other subroutines.

C  ERRSET is called because underflow errors may occur in the program.
C  ERRSET is an IBM system program.  Other adaptations of this
C  program to different systems will require a call to a different
C underflow-adjustment system routine.  If no such subroutine is
C  available, the user may have to modify the program.
C  Potential trouble spots are:
C                 ------
C
C  &&(Chris  Vic will determine this by running the pgm without
C  && ERRSET being called.)
C
C      CALL ERRSET (208, 0, -1, 1, 1)
C      MAXUN = 11
C      INP   = 5
C      IDEBUG= 1
C                 ...debug toggled 'on'  for the time being
C      IOUT  = 6
C            IOUT and IDEBUG must be set  for $PRSUB to work
C      $SUB=$PRSUB('INIT')
C
C  If this program has been run once, the answers will be saved
C  on unit ISAVE so that a restart option is available.
C  To avoid meaningless prompting on the terminal, the
C  output will be suppressed by writing it to unit ISUP.
C      ISUP=99
C      ISAVE=98
C      IFLAG=0
C      WRITE(IOUT,*) ' Defaults are explicitly noted, or are
C     &  'indicated by a '*' on the menus.'
C      WRITE(IOUT,*)  Defaults may be obtained by entering only a
C     &   carriage return after the prompt.
C      WRITE(IOUT,*)
C904   CONTINUE
C 904 WRITE(IOUT,*) 'Is this a restart? (*Yes = 1, No = 0)'
C      ANRES = 1
C      GO TO 912
C 911 CALL DFALTI(ANRES,1)
C912   IF(ANRES.LT.0.OR.ANRES.GT.1) THEN
C        WRITE(IOUT,*)  Error in restart specification ',ANRES
C     &                                  ' Choose 0 or 1'
C        IFLAG=IFLAG+1
C        IF(IFLAG.LE.1) GO TO 904
C        CALL IBTERM(ANRES,$SUB)
C      ENDIF
C      IF (ANRES.EQ.1) THEN
C        INP=99
C                 ....to avoid colliding with filedata for v-levels
C        IOUT=ISUP
C      ENDIF
       PLANCK=6.63618D-34
       KBOLTZ=1.38066D-23
C      KBOLTZ=1.3805D-23
C  K obtained from Rev. Mod. Phys. vol. 27, p. 363 (1955).
       RGAS=KBOLTZ * AVOGAD
       CLIGHT=2.997925D-08
       I4EVEP=10000000
       SCEXP1=1.0D0
       PI=3.141592653589793
       EMASS=0.0D0
       CMASS=0.0D0
       AMASS=0.0D0
       RELMAS=0.0D0
       RRAD=0.0D0
       CRAD=0.0D0
       ARAD=0.0D0
       RE=0.0D0
C The maximum number of output units is set by MAXUN.  For
C  V-levels ranging from 0 to 10, this means 11 output files
C  each of which will contain population information  for
C  appropriate J sub-levels. The rate-of-population-change
C  information is contained on 11 more output files.
C  2 additional files will monitor total population
C  and total rate-of-population-change.
C                                          unit                VUNIT index
C        NV data for V=0....10       units 10....20             0--10
C        NV data summed (NTOT)             21                    11
C        DNV data for v=0...10       units 40....50             12--22
C        DNDT   data                 unit 51                    23
C
       MAXV=0
       DO 20 V=0,10
         MAXJ(V)=0
         VUNIT(V)=10+V
         VUNIT(V+12)=40+V
  20   CONTINUE
C       The matrix ELEV must have one extra level, so that the
```

```
C       energy difference can be calculated in EDEL for a given level.
        DO 10 V = 0 10
          DO 10 J= 0 20
            ELEV(V J)=0.0
   10   CONTINUE
C NV initialized  value unimportant  normalized later.
        NV(0 0) = 1.0
C
C       NOTES ON DEFAULTS:
C
C  The default value is either explicitly denoted as such
C  or it is indicated with a  *  in front of the menu number.
C
C  If the input is from the terminal  a carriage return will
C  signify that the default value is desired.
C  If the input is from a file  then reset the input unit
C  to that of the terminal  so that the user can pick up
C  from where s/he left off...
C
C  The following default subroutines are called:
C         DFALTD - for double precision numbers
C         DFALTI - for integer numbers
C         DFALTC - for character input.
C  If the user wishes to change the default value  only the
C  prompting default reminder  and the second argument in the
C  IFALTn call must be changed.
C
  900  WRITE(IOUT,*)  ' Type a ''1 '' if prompting for units is
     &  ' required. Any other number--no prompting. <Default: 0>'
C         (The routine for prompting for units is NOVICE.)
       READ(INP,*,END=901) INOV
       GO TO 902
  901  IF(IOUT.EQ.ISUP) THEN
         INP=5
         IOUT=6
         GO TO 900
       ELSE
         CALL DFALTI(INOV,0)
       ENDIF
  902  WRITE(ISAVE *) INOV
       JCOLL(1)=0
       JCOLL(2)=0
       JCOLL(3)=0
       $SPGEN(1)=' For collisions between A and BC: '
       $SPGEN(2)=' For collisions between BC and BC: '
       $SPGEN(3)=' For collisions between M and BC: '
       IF(INOV.NE.1) RETURN
       WRITE(IOUT,*)' This program accepts many kinds of units.'
       WRITE(IOUT,*)' The user must know which two-letter codes refer'
       WRITE(IOUT,*)' to which units. e.g., EV means electron volts.'
       WRITE(IOUT,*)' The letter code for inverse units is identical to'
       WRITE(IOUT,*)' that of the non-inverted units.  The user must'
       WRITE(IOUT,*)' input values and units  with the units enclosed
      1  ' in single quotes.'
      2  ' and separated by commas  e.g. "4.0  'cm' ."'
       WRITE(IOUT,*) ' '
       RETURN
       END
```

```
       SUBROUTINE FC1SCL(ANF01)
C  In the calculation of the rate coefficient for v---->v+1 scaling
C  the value from which all other values are scaled must first be
C  obtained.  This routine prompts for k(0,0-->1,0)
C  and  if rotational levels exist (i.e.,  vonly=0) it prompts
C  for k(0,0-->1,2)  and k(0,2-->1,0).
C  When ANF01 is 1  Landau-Teller scaling is desired.
C  If the argument ANF01 is 2  use an alternate form of scaling.
       IMPLICIT DOUBLE PRECISION (A-H F,N-O W-Z)
       IMPLICIT INTEGER (A I-J,L-M V)
       IMPLICIT CHARACTER*8 (S)
       CHARACTER*3 UNIT
       COMMON/SPCIES/ISPEC ISPEC(6 3)
       COMMON/FV1B/FUPUP(0:10 0:20) FUPD(0:10 0:20)  FUP(0:10 0:20)
      1  FDOWNP(0:10 0:20) FDOWND(0:10 0:20) FDOWN(0:10 0:20)
      2  FUP(0:10 0:20) FDOWN(0:10 0:20)
       COMMON/IO/INP,IOUT,ISAVE ISUP IDEBUG INDAT,INOV VONLY VUNIT(0:23)
       COMMON/ENERGY/ELEV(0:10,0:20) TEMP
       COMMON/PHYSCT/PLANCK RBOLTZ RGAS SCEXP1 CLIGHT PI,INEVER
       COMMON/COUNT/MAXV,MAXJ(0:10) MAXUN  INPTS  IOLL
       ISUB=$PRSUB('FC1SCL')
       WRITE(IOUT,701) (ISPEC(L ISPEC)  L=1 6)
  701  FORMAT(1X,6A6)
  900  WRITE(IOUT,*)  Input the units to be used for v --> v+1
     &  '                rate constants: <default: ''cm3/s'' >'
       IF(INOV.EQ.1) CALL NOVICE(30)
       READ(INP,*,END=901) UNIT
       GO TO 902
  901  IF(IOUT.EQ.ISUP) THEN
         INP=5
         IOUT=6
         GO TO 900
       ELSE
         CALL DFALTC(UNIT, 3)
       ENDIF
  902  WRITE(ISAVE *) UNIT
C            ... dummy call to SIU checks for valid unit
       CALL SIU(20,0,50 UNIT)
C  Originally we prompted for k01 but this turned out to be inconvenient
C  (the literature usually cites k10).  so now we prompt for k10 and
C  immediately convert to k01 so that the remainder of the code is not
C  affected.  Users may wish to prompt for k01...
  903  WRITE(IOUT,*)  'Input k for v=1 J=0 -> v=0 J=0 transition:'
       WRITE(IOUT,*)  <Default: 1.44E7>'
       READ(INP,*,END=904) K1000
       K1000 = GETK10(ISPEC-1)
       GO TO 905
  904  IF(IOUT.EQ.ISUP)  THEN
         INP=5
         IOUT=6
         GO TO 903
       ELSE
         CALL DFALTD(K1000 1.44D7)
       ENDIF
  905  WRITE(ISAVE,*) K1000
       CONST=-1.0/(RGAS*TEMP)
       FUPD(0,0) = K1000*DEXP(EDEL(0,0 1 0)*CONST)
C      IF (IDEBUG.EQ.1, WRITE(6 *)  FupD=  '  FUPD(0 0)
       CALL SIU(20,KUPD(0,0),UNIT)
       IF (VONLY.EQ.0)  THEN
  906    WRITE(IOUT,*) 'Input k for v=1 J=2 -> v=0 J=0 transition:'
         WRITE(IOUT,*) <Default: 1.8D7>
         READ(INP,*,END=907) K1200
         GO TO 908
  907    IF(IOUT.EQ.ISUP)  THEN
           INP=5
           IOUT=6
           GO TO 906
         ELSE
           CALL DFALTD(K1200 1.8D7)
         ENDIF
  908    WRITE(ISAVE *) K1200
         FUPUP(0,0) = K1200*DEXP(EDEL(0 0 1 2)*CONST)
         CALL SIU(20,KUPUP(0,0), UNIT)
  909    WRITE(IOUT,*)  Input k for v=1 J=0 -> v=0 J=1 transition:'
         WRITE(IOUT,*)  <Default: 1.2D7>
         READ(INP,*,END=910) K1002
         GO TO 911
  910    IF(IOUT.EQ.ISUP) THEN
           INP=5
           IOUT=6
           GO TO 909
         ELSE
           CALL DFALTD(K1002,1.2D7)
         ENDIF
  911    WRITE(ISAVE *) K1002
         FUPD(0,0) = K1002*DEXP(EDEL(0 0 1 0)*CONST)
         CALL SIU(20,KUPD(0,0) UNIT)
C         Must find KUPUP(0 all J) and KUPD(0,all J) and FUP(0 all J)
C       --this will be done using a modified KROTUP routine.
         CALL KROT2
       ENDIF
       IF(ANF01.EQ.1) THEN
         CALL FC1LT
       ELSEIF(ANF01.EQ.2) THEN
         CALL FC1ALT
       ELSE
         CALL IBTERM(ANF01 ISUB)
       ENDIF
       RETURN
       END
```

```
      SUBROUTINE TMPTUR
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      IMPLICIT INTEGER (I)
      IMPLICIT CHARACTER*6 ($)
      CHARACTER*2 UNIT1
      INTEGER VUNIT, VONLY
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      ISUB='PPSUB' 'TMPTUR'
115   WRITE(IOUT,*)  ' Input the temperature. <Default: 300   Ke  >
      IF(INOV.EQ.2) CALL NOVICE(50)
      READ(INP,*,END=903)  TEMP,UNIT1
      TEMP = GETTMP()
      GO TO 904
903   IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 115
      ELSE
        CALL DFALTD('TEMP',300.0D0)
        CALL DFALTC(UNIT1,'KE')
      ENDIF
904   WRITE(ISAVE,*) TEMP,UNIT1
      CALL SIU(50,TEMP,UNIT1)
      RETURN
      END
```

```
      SUBROUTINE CONCN
      IMPLICIT DOUBLE PRECISION A-H,O-Z
      IMPLICIT INTEGER (I)
      IMPLICIT CHARACTER*6 ($)
      CHARACTER*2 UNIT1
      INTEGER VUNIT, VONLY, JCOLL
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/PSP/NV(0:10,0:20),NTOT,CONCA,CONCBC,CONCM,JCOLL(3)
      ISUB='ERSUB' CONCN
110   WRITE(IOUT,*)   ' What is the unit of concentration used?'
      WRITE(IOUT,*)   '<Default:  no    molar>'
      IF(INOV.EQ.2) CALL NOVICE(60)
      READ(INP,*,END=900)  UNIT1
      GO TO 901
900   IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 110
      ELSE
        CALL DFALTC(UNIT1,'NO')
      ENDIF
901   WRITE(ISAVE,*) UNIT1
C       Dummy call to SIU, just to check if units for concentration
C       are kosher.
      CALL SIU(80,1.0D0,UNIT1)
111   WRITE(IOUT,*)   ' What is the concentration of species A?
      WRITE(IOUT,*)   '<Default: 1.0D-6>
      READ(INP,*,END=902)  CONCA
      GO TO 903
902   IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
CAAA what happened to IFLAG...?
        GO TO 111
      ELSE
        CALL DFALTD('CONCA',1.0D-6)
      ENDIF
903   WRITE(ISAVE,*) CONCA
      CALL SIU(60,CONCA,UNIT1)
112   WRITE(IOUT,*)  ' What is the concentration of species BC?'
      WRITE(IOUT,*)  '<Default: 1.0D-3>'
      READ(INP,*,END=904)  CONCB
      GO TO 905
904   IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 112
      ELSE
        CALL DFALTD('CONCBC',1.0D-3)
      ENDIF
905   WRITE(ISAVE,*) CONCBC
      CALL SIU(60,CONCBC,UNIT1)
113   WRITE(IOUT,*)   'What is the concentration of relaxant gas?'
      WRITE(IOUT,*)   '<Default: 1.0D-1>'
      READ(INP,*,END=906)  CONCM
      GO TO 907
906   IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 113
      ELSE
        CALL DFALTD('CONCM',1.0D-1)
      ENDIF
907   WRITE(ISAVE,*) CONCM
      CALL SIU(60,CONCM,UNIT1)
C  The three gases present in the default concentrations give a
C  combined pressure of about 60 torr.
      RETURN
      END
      PROGRAM BSJC
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      COMMON/BSTOWS/X   TMP,  K10(3), FPH(500),   ATENT(500),  LOCFCT,  JCOLL(
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN,  INPTS,  ICOLL
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/SPCIES/ISPEC,$SPEC(6,3)
      COMMON/POP/NV(0:10,0:20),NTOT,CONC(3),JCOLL(3)
      PRINT*,   Version 064  of BIMSIM, 20 May    1990'
      PRINT*,  '
      CALL DC30
      STOP 1006
      END
C This program simulates the kinetics of bimolecular reactions.
C
C   Rotational levels included.
C
C     Variables:
C       The assumptions made for the allowed v, j--> v',j' transitions
C       are: a change of 0 or 1 or -1 in the vibrational level;
C       and a change of 0 or 2 or -2 in the rotational levels.
C
C KUPUP, KUPD, KUP0 --
C         These two dimensional arrays contain the energy transfer rate
C         constants in the 'up' direction.  The convention is as
C         follows: KUPUP(v,J) => the transition v,J -> v+1,J+2
C                  KUPD(v,J)  => the transition v,J -> v+1,J-2
C                  KUP0(v,J)  => the transition v,J -> v+1,J
C  KDOWNU  KDOWND, KDOWN0 --
C          Similarly,these two-dimensional arrays contain the energy
C          transfer rate constants in the 'down' direction.
C         The convention is as follows:
C                  KDOWNU(v,J) => the transition v,J  -> v+1,J+2
C                  KUPD(v,J)  => the transition v,J -> v+1,J-2
C                  KUP0(v,J)  => the transition v,J -> v+1,J
C The J-->J-2 transition rate coefficients can be determined
C  by microscopic reversibility (along with v-->v-1 rates).
C
C The following relationships exist through microscopic reversibility:
C                  KUPUP(v,J) <------> KDOWND(v,J)
C                  KUPD(v,J)  <------> KDOWNU(v,J)
```

```
C                 KUP(V,J)  <-- --> KDOWN(V,J)
C                 KUP(V,J)  <------ KDOWN(V,J)
C
C  Note that these matrices occupy only a subset of the allocated
C  space.  For example
C  KDOWN(0:10,0:20) actually only has values over (1:10,2:100)
C  and KUP(0:10,0:20) would only have values over (0:9,0:98).
C
C  KROT -- When the vibrational level does not change, but
C          the rotational level changes  to 2 or -2.
C  KVCBIM -- The bimolecular reaction rate coefficient for
C            each particular v,J level.
C  DESDEL -- (in SOLVER) The desired "delta" or fractional decrement
C            in population before a value is output for a given
C            v level.
C
C   IOUT -- an integer  holds the unit on to which prompts will go.  If
C input is from the terminal then prompts go to the terminal.  If input
C is coming from a file (including a file saved from a previous run)
C then prompts are redirected to a dummy file and are not sent to the
C terminal.
C
C  Values are classified according to what kind of quantity they
C  represent (energy, mass, temperature, etc.).
C  To determine what "kind" a value is, divide its IREAD code by 10.
C  The general divisions are:
C       (1) energy
C       (2) rate constants
C       (3) distance, inverse distance
C       (4) mass
C       (5) temperature
C       (6) population
C
C The reason that the number representing the type of conversion
C needed must be integer-divided by 10
C is that this number serves to store more than one
C type of information: not only the kind of quantity (e.g.  energy,
C rate constant, etc.) but the physical
C meaning associated with that number for purposes of reading in
C values in READVJ and READIN.  For example there is more than one type
C of rate constant (quantity type 2) that is read in but each must have
C the same units-conversion applied and therefore the code associated
C with those values (IREAD) is numbered 21, 22, 23, etc.
C This has the drawback that the program will have to be changed
C substantially if, for example, there were more than 10 types of rate
C constants between which we wished to distinguish.
C The advantage of this approach is that the program has consistency
C in the codes used by subroutines READIN, READVJ, SIU, NOVICE.
C
C  This program converts units specified for the following
C  types to SI units:
C
C       (1) energy
C            10 - "generic" energy
C            11 - E(V), disregarding J
C            12 - E(v,J)
C            13 - E(J), for a set v
C       (2) rate constants
C            20 -  generic  rate constants
C            21 - k(v), disregarding J
C            22 - k(v,J)
C            23 - k(J), for a set v
C            24 - kbim(v), disregarding J
C            25 - kbim(v,J)
C       (3) distance, inverse distance
C            30 - distance
C            39 - inverse distance
C       (4) mass
C            40 - mass
C       (5) temperature
C            50 - temperature
C       (6) population
C            60 - "generic" population
C            61 - n(V), disregarding J
C            62 - n(v,J)
C            63 - n(J), for a set v
C
C  Conversion factors are based on W.H. Flygare(1978)
C
C  If data are being read in from particular files, the user must
C   be sure to define the locations on which the files are located
C   according to the following assignments (from subroutine READIN):
C
C   IREAD                         UNIT FOR READING IN FILE
C   (from subroutine READIN):
C      11 -- v, E(v,J=0)               1
C      21 -- v, kup(v,0)               2
C      24 -- v, kbim(v,0)              3
C      61 -- v, nv(v,0)                4
C      13 -- J, E(v=0,J)               7
C
```

```
      SUBROUTINE BIMSIM(TOL,HINIT, RESET, TFIRST,TMULT,IEP)
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDEBUG,INDAT,INOV,VONLY,VUNIT(0:23)
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXCH, INPTS, ICOLL
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/SPECIES/ISPEC,SSPEC(6,3)
      COMMON/POP/NV(0:10,0:20),NTOT,CONC(3),JCOLL(3)
C       ...  for concentrations CONCA,CONCBC,CONCM, respectively
C
C  IOUT and IDEBUG must have values when SPRSUB is called but
C  their values
C  are assigned proper in subroutine INIT.
      IDEBUG = 0
      IOUT   = 6
      ISUP (SPRSUB) BIMSIM )
C                           Change comment stmt with version number.
      PRINT*,'   Version 064  of BIMSIM  30 May    1990'
      PRINT*
      CALL INIT
      CALL RXNSYS
      IF(INOV.EQ.1) CALL ASSUME
      CALL TMPTUR
C  The temperature should be known first, in case the user specifies
C  partial pressure in the CONC subroutine.
      CALL CONCN
      CALL PIKMET(ANMETH)
C         For each species in the reaction, the rate coefficients
C         must be found.
C
C   The vibrotational structure of diatomic BC is of interest
C   but not that of anything such as the relaxant gas M which
C   collides with BC, and thus PIKGNU, where the potential energy
C   curve is specified, is not within the DO 100 loop.
C
      CALL PIKGNU(ANGNU)
C
C   Now that the ELEV array for BC has been calculated and filled in
C   from (0,0) to (maxv,maxJ), we can consider the effect of collisions
C   on BC.
C
      ICOLL=0
      DO 100 ISPEC=1,3
C                         ...the rate coefficients depend on what SPECies
C                         collide with BC:  A, BC itself or M.
 901    WRITE(IOUT,*) ' '
        WRITE(IOUT,*) '*******************************************'
        WRITE(IOUT,*)
        WRITE(IOUT,701) (SSPEC(L,ISPEC), L=1,6)
 701    FORMAT(1X,6A6)
        WRITE(IOUT,*)'Type 1 to continue with these collision partners.'
        WRITE(IOUT,*) '(Otherwise, type another number.)  <default = 1>'
        READ(INP,*,END=902) ANCOLL
        GO TO 903
 902    IF(IOUT.EQ.ISUP) THEN
          INP=5
          IOUT=6
          GO TO 901
        ELSE
          CALL DFALTI(ANCOLL,1)
        ENDIF
 903    WRITE(ISAVE,*) ANCOLL
        IF(ANCOLL.NE.1) THEN
          GO TO 100
        ELSEIF(IGETCP().EQ.1.AND.ISPEC.EQ.3) THEN
        READ(INP,*)
        READ(INP,*)
          GO TO 100
        ELSE
          ICOLL=ICOLL+1
          JCOLL(ICOLL)=ISPEC
C The values of the array JCOLL are set up according to the following
C collision partners with BC:
C         1  -  A + BC
C         2  -  BC + BC
C         3  -  relaxant M + BC
C    e.g.  if we only want to consider A + BC and M + BC cases
C           then JCOLL(1) = 1 and JCOLL(2) = 3.
C The JCOLL array is initialized to zero  so that the total number of
C collision types may be determined by finding the position of
C the last non-zero value.
        ENDIF
C
C Calculation of  KOUP vector for each v level...
C  where  KOUP(v,m) means k(v,m)-->k(v,m+2)
C   (only called if ro-vib. solution wanted)
C
        IF(VONLY.EQ.0) CALL PIKKOU(ANKOUP)
C The J-->J-2 transition rate coefficients can be determined
C  by microscopic reversibility (along with v-->v-1 rates).
C
        IF (ANGNU.NE.1) THEN
C                           ANGNU=1 means that K01 values are determined
          CALL PIKK01(ANK01)
        ELSE
          CALL K01SCL(1)
C                     .... arg. 1 means Landau-Teller scaling to be done
        ENDIF
        CALL CHECKR
C MICREV must be called before COLLCT:
        CALL MICREV
        CALL COLLCT
 100  CONTINUE
      CALL PIKKRX(ANKMRX)
C  After the user has specified the method of obtaining the
C  microscopic rate coefficient in this subroutine, and the krxn(v,J)
C  are found in the various subroutines PIKKRX branches to,
C  then the krxn should be examined to see which values are
C  not reasonable for the system of interest.
C  For a bimolecular reaction, this would imply that those krxn
C  which exceed the collision frequency possible, according
```

```
C  to the temperature and pressure of the system.
C  Thus, the subroutine DIFFUS is called.
C  In this subroutine, the collision frequency is calculated
C  and any krxn(v,J) which exceeds it is set equal to it.
C  If the user is modelling a unimolecular reaction, or an ideal
C  reaction which is not limited by the rate of diffusion, then
C  'no upper limit' in the subroutine DIFFUS should be specified.
      IF(ANMETH.NE.2) THEN
        CALL PIKPOP(ANPOP)
        CALL DIFFUS
        CALL SOLVER(TOL,HINIT, RESET, TFIRST,TMULT,IER)
        RETURN
      ENDIF
      IF(ANMETH.NE.1) THEN
        CALL FINIT(ICTR1)
        CALL EIGEN (ICTR1)
      ENDIF
 907  WRITE(IOUT,*) Type 1 if you want tabulated output. <Default: 1>
      READ(INP,*,END=905) ANTAB
      GO TO 906
 905  IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 907
      ELSE
        CALL DFALTI(ANTAB,1)
      ENDIF
 906  WRITE(ISAVE,*) ANTAB
      IF (ANTAB.EQ.1) CALL TABOUT
 908  WRITE(IOUT,*)'Type 1 if you want output plotted <Def.: 1>'
      READ(INP,*,END=909) ANPLOT
      GO TO 910
 909  IF(IOUT.EQ.ISUP) THEN
        INP=5
        IOUT=6
        GO TO 908
      ELSE
        CALL DFALTI(ANPLOT,0)
      ENDIF
 910  WRITE(ISAVE,*) ANPLOT
      IF (ANPLOT.EQ.1) CALL PLTOUT
      RETURN
      END
```

```
      SUBROUTINE DO10
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      COMMON/BS30WS/X, TMP, K10(2),KPH(500), XTENT(500), LOOPCT, ICOLL
      COMMON/BSDATA/HUSED(500), IQUSED(500), ISTEP(500), IUNDAR(500)
     1Y0(500),Y1(500),Y2(500),Y3(500),Y4(500)
      COMMON/POP/NV(0:10,0:20),NTOT,CONCA,CONCBC,CONCM,JCOLL(3)
      COMMON/UNDERF/IUNDER
      EXTERNAL TRAKUF
      CALL ERRSET (208, 0, -1, 1, TRAKUF 207)
      CALL TBLTOP
      OPEN(4,FORM='UNFORMATTED')
      INNER = 0
 10     READ(1,*,END=20) KHTTMP, CONCA, CONCBC, CONCM, TMP,
     1   ICOLL, K10(1), K10(2), TOL, HINIT, RESET, TFIRST, TMULT
      SCALEC = 1.D0
      CONCA = CONCA*SCALEC
      CONCBC= CONCBC*SCALEC
      CONCM = CONCM*SCALEC
      IF(ICOLL.EQ.1) THEN
       X = CONCA/CONCBC
      ELSE
       X = CONCA/CONCM
      ENDIF
        DO 30 M=0,99
        IF(M.EQ.1.OR.M.EQ.2.OR.M.EQ.4.OR.M.EQ.5) GOTO 30
        REWIND(M)
 30     CONTINUE
        LOOPCT = 0
        INNER = INNER + 1
        IUNDER = 0
        ISTEP(1) = 0
        WRITE(6,'(1X,I2,1X,4(E8.1,1X),I1,2X,2(E8.1))')
     1          INNER,CONCA,CONCBC,CONCM, TMP,ICOLL,K10
      WRITE(6,*)HINIT, RESET, TFIRST
        CALL BIMSTM(TOL,HINIT, RESET, TFIRST,TMULT,IER)
        CALL OUTPUT(KHTTMP,IER)
        GOTO 10
 20     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE TBLTOP
      WRITE(2,'(2(A10,1X),4(A7,1X),A8,1X,A3)')
     $   'k_min/k_eq','HT results', 'ConcA','ConcBC',
     $   'ConcM','Temp','Collider','IER'
      RETURN
      END
```

```
      SUBROUTINE TBLOUT(KHT,IER)
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      COMMON/BS30WS/X, TMP, K10(2),KPH(500), XTENT(500), LOOPCT, ICOLL
      COMMON/POP/NV(0:10,0:20),NTOT,CONCA,CONCBC,CONCM,JCOLL(3)
      COMMON/BSDATA/HUSED(500), IQUSED(500), ISTEP(500), IUNDAR(500)
     1Y0(500),Y1(500),Y2(500),Y3(500),Y4(500)
      CHARACTER*3 COLLIDR
      IF(ICOLL.EQ.1) THEN
        COLLIDR = 'HCl'
      ELSE
        COLLIDR = 'He'
      ENDIF
      ITMP = TMP
      CALL USMNMX(KPH,LOOPCT,1,KMIN, KMAX)
      KRAT = KMIN/KPH(1)
      ISTEPS = ISTEP(LOOPCT)
      KDIFF = (KRAT-KHT)/KHT
      WRITE(6,*)ISTEPS,KDIFF,KDIFF/DBLE(ISTEPS)
      WRITE(2,'(1P,2(E10.4,1X),3(E7.1,2X),0P,I7,1X,A8,1X,I3)')
     $   KRAT,KHT,CONCA, CONCBC, CONCM,ITMP,COLLIDR,IER
      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION GETCA()
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      COMMON/BS30WS/X, TMP, K10(2),KPH(500), XTENT(500), LOOPCT, ICOLL
      COMMON/POP/NV(0:10,0:20),NTOT,CONCA,CONCBC,CONCM,JCOLL(3)
      GETCA = CONCA
      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION GETCBC()
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      COMMON/BS30WS/X, TMP, K10(2),KPH(500), XTENT(500), LOOPCT, ICOLL
      COMMON/POP/NV(0:10,0:20),NTOT,CONCA,CONCBC,CONCM,JCOLL(3)
      WRITE(6,*) 'in getcbc', CONCBC
      GETCBC = CONCBC
      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION GETCM()
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      COMMON/BS30WS/X, TMP, K10(2),KPH(500), XTENT(500), LOOPCT, ICOLL
      COMMON/POP/NV(0:10,0:20),NTOT,CONCA,CONCBC,CONCM,JCOLL(3)
      GETCM = CONCM
      RETURN
      END
```

```
      FUNCTION IGETCP()
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      COMMON/BS30WS/X, TMP, K10(2),KPH(500), XTENT(500), LOOPCT, ICOLL
      IGETCP = ICOLL
      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION GETTMP()
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      COMMON/BS30WS/X, TMP, K10(2),KPH(500), XTENT(500), LOOPCT, ICOLL
      GETTMP = TMP
      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION GETK10(ISPEC)
      IMPLICIT DOUBLE PRECISION (B-H,K,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 (S)
      COMMON/BS30WS/X, TMP, K10(2),KPH(500), XTENT(500), LOOPCT, ICOLL
      GETK10 = K10(ISPEC)
      RETURN
      END
```

```
      SUBROUTINE OUTPUT(KHT,IER)
      REAL*8 KHT
      CALL GETDAT
      CALL TBLOUT(KHT,IER)
      CALL PLTDAT
      RETURN
      END
```

```
      SUBROUTINE GETDAT
      IMPLICIT DOUBLE PRECISION (B-H,P,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      COMMON/BS3CWS/X,  TMP,  P(10),FPH(500),  XTENT(500),  LOOPCT,  ICOLL
      COMMON/BSDATA/HUSED(500),  IQUSED(500),  ISTEP(500),  IUNDAR(500),
     1Y0(500), Y1(500), Y2(500), Y3(500), Y4(500)
      CHARACTER*80 STRING
      REWIND(21)
      REWIND(23)
      READ(21,'(A80)') STRING
C     WRITE(6,*) STRING, LOOPCT
      DO 10 I = 1,LOOPCT
      READ(21,   (3(D14.7),3(2I10)), 1   XTENT(I),FPH(I),
     1   HUSED(I),  IQUSED(I),  ISTEP(I), IUNDAR(I)
      READ(22   (D20.13,4(D15.8))')  Y0(I), Y1(I), Y2(I),Y3(I), Y4(I)
C     XTENT(I)  = LOG(XTENT(I))
 10   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE PLTDAT
      IMPLICIT DOUBLE PRECISION (B-H,P,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      COMMON/BS3CWS/X,  TMP,  P(10),FPH(500),  XTENT(500),  LOOPCT,  ICOLL
      COMMON/BSDATA/HUSED(500),  IQUSED(500),  ISTEP(500),  IUNDAR(500),
     1Y0(500), Y1(500), Y2(500), Y3(500), Y4(500)
      COMMON/POP/NV(0:10,0:20),NTOT,CONCA,CONCBC,CONCM,JCOLL(3)
      CALL USMMMX(FPH,LOOPCT,1,FMIN,  FMAX)
      EPS = 1.D-5
C     DO 11 I=1,LOOPCT
C       IF((FPH(I)-FMIN)/FPH(1).LT.EPS) WRITE(6,*)'YES ',XTENT(I)
C11     IF((FPH(I)-FMIN)/FPH(1).GE.EPS) WRITE(6,*)'NO ',XTENT(I)
      WRITE(4)LOOPCT,ICOLL,TMP,X
      WRITE(4)(XTENT(I),I=1,LOOPCT)
      WRITE(4)(FPH(I),I=1,LOOPCT)
      WRITE(4)(HUSED(I),I=1,LOOPCT)
      WRITE(4)(IQUSED(I),I=1,LOOPCT)
      WRITE(4)(ISTEP(I),I=1,LOOPCT)
      WRITE(4)(IUNDAR(I),I=1,LOOPCT)
      WRITE(5)(Y0(I),I=1,LOOPCT)
      WRITE(5)(Y1(I),I=1,LOOPCT)
      WRITE(5)(Y2(I),I=1,LOOPCT)
      WRITE(5)(Y3(I),I=1,LOOPCT)
      WRITE(5)(Y4(I),I=1,LOOPCT)
      RETURN
      END
```

```
      SUBROUTINE INCCT
      IMPLICIT DOUBLE PRECISION (B-H,P,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      COMMON/BS3CWS/X,  TMP,  P(10),FPH(500),  XTENT(500),  LOOPCT,  ICOLL
      LOOPCT = LOOPCT + 1
      IF(LOOPCT.GT.500) THEN
      WRITE(6,*)'oops...arrays are full'
      STOP
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE DIFFUS
C The krxn(V,J) should be examined to see which values are
C not reasonable for the system of interest.
C For a bimolecular reaction  this would imply that those krxn
C which exceed the collision frequency possible  according
C to the temperature and pressure of the system.
C should have an upper limit which is equal to or less
C than the rate of diffusion. The rate-limiting step is
C no longer the speed of chemical conversion  but the
C speed at which one molecule can meet another molecule.
C In this subroutine  the collision frequency is calculated
C and any krxn(V,J) which exceeds it is set equal to it.
C If the user is modelling a unimolecular reaction  or an ideal
C reaction which is not limited by the rate of diffusion  then
C 'no upper limit' in the subroutine DIFFUS should be specified.
      IMPLICIT  UBLE PRECISION (B-H,P,N-U,W-Z)
      IMPLICIT INTEGER (A,I-J,L-M,V)
      IMPLICIT CHARACTER*6 ($)
      REAL*8 AVOGAD  AMASS,  ARAD
      PARAMETER  (AVOGAD=6.02205D23)
      COMMON/IO/INP,IOUT,ISAVE,ISUP,IDERUG,INDAT,INOV,VONLY,VUNIT(0:23),
      COMMON/COUNT/MAXV,MAXJ(0:10),MAXUN   INPTS   ICOLL
      COMMON/MASTR/KVJBIM(0:10,0:20)
      COMMON/ENERGY/ELEV(0:10,0:20),TEMP
      COMMON/PHYSCT/PLANCP,KBOLTZ,RGAS,SCEKP1,CLIGHT,PI,I4EVER
      COMMON/MASS/BMASS,CMASS,AMASS,REDMAS,BRAD,CRAD,ARAD,RE
      COMMON/POP/NV(0:10,0:20),NTOT,CONCA,CONCBC,CONCM,JCOLL(3)
      $SUB=$PRSUB('DIFFUS')
      IFLAG = 0
102   WRITE(IOUT,*)'Should the microscopic rate constants
     &' have an upper limit equal to that of the maximum collision ',
     &'frequency possible?
      WRITE(IOUT,*)'          1) yes, establish this as an upper limit'
      WRITE(IOUT,*)'          2) the upper limit is proportional to ',
     &'  the collision frequency'
      WRITE(IOUT,*)'          3) no upper limit (e.g. unimolecular rxn)'
      READ(INP,*,END=903) ANDIFF
      GOTO 904
903   IF(IOUT.EQ.ISUP) THEN
        INP  = 5
        IOUT = 6
        GOTO 102
      ELSE
        CALL DFALTI(ANDIFF,1)
      ENDIF
904   WRITE(ISAVE,*) ANDIFF
      IF (ANDIFF.EQ.3) THEN
       GO TO 104
      ELSEIF (ANDIFF.NE.1.AND.ANDIFF.NE.2) THEN
       WRITE(IOUT,*)' Error in upper limit specification',ANDIFF
       WRITE(IOUT,*)' Specify 1, 2 or 3.'
       IFLAG = IFLAG + 1
       IF (IFLAG.LE.1) GOTO 102
       CALL XBTERM(ANDIFF,$SUB)
      ENDIF

C The rate of collision  Z  may be expressed in a form analogous to
C the rate of reaction  with a diffusion coefficient  k_diff:
C                        Z = k_diff (A) (BC)
C where (A) and (BC) are the concentrations of the species involved
C in collision.
C For each VJ level
C we want to compare
C k_diff with the K_VJbim coefficient input to the
C program.  Since only collisional processes are considered in this
C simulation  no K_VJbim should exceed k_diff.
C The formula for Z is from J.N. Levine, Physical Chemistry
C (Mcgraw-Hill  1978), p. 411:
C    Z = pi*(r_A + r_BC)**2*((8*R*T/pi)*(1/M_A + 1/M_BC))**(1/2)
C      *(N_A/V)*(N_BC/V)
C Therefore  k_diff = Z/([A]*[BC]*N_Av)
C where the additional N_AV in the denominator comes in because we are
C considering the number of 'moles' of collisions rather than the
C number of collisions.
C    or       k_diff = pi*(r_A + r_BC)**2*((8*RT/pi)
C                      *RECiprocal-Reduced-Mass)**(1/2)*N_Av
C This will be used to estimate the upper limit for
C the bimolecular rate coefficient.
C The molecular diameter is estimated as the sum of atomic
C radii  whose values are passed through common block MASS.
C The atomic masses and radii are found in subroutine RXNSYS.
C Since we are concerned only with reactive collisions (A+BC)
C we ignore M-BC collisions in determining k_diff.
      RBC = (BRAD+CRAD)
      BCMASS=BMASS+CMASS
      RECRM = (1.0D0/AMASS + 1.0D0/BCMASS)*AVOGAD
C AMASS and BCMASS come in on a 'per mole' basis....we need them
C on a 'per molecule' basis  so we divide the mass by N_AV  which
C is equivalent to multiplying the reciprocal by N_AV as above.
      KDIFF=PI*(ARAD+RBC)**2*DSQRT(8.0D0*RGAS*TEMP/PI)*RECRM)*AVOGAD
      WRITE(IOUT,700) KDIFF
700   FORMAT(' Collision coefficient (k_diff) is ',D12.4,' sec-1')
      IF(ANDIFF.EQ.2) THEN
105     WRITE(IOUT,700)'What is the ratio of the upper limit to the ',
     &       ' collision coefficient? <Default: 1.0>'
        READ(INP,* END=908) RATIO
        GOTO 909
908     IF(IOUT.EQ.ISUP) THEN
          INP  = 5
          IOUT = 6
          GOTO 105
        ELSE
          CALL DFALTD(RATIO,1.0D0)
        ENDIF
909     KDIFF=RATIO*KDIFF
      ENDIF
      DO 100 V=0, MAXV
        DO 100 J=0,MAXJ(V)
          IF(KVJBIM(V,J).GT.KDIFF) THEN
            WRITE(6,701) V,J,KVJBIM(V,J),KDIFF
            KVJBIM(V,J)=KDIFF
701         FORMAT(' Kvjbim(',2I3,') exceeds the collision ',
```

```
     &                'coefficient: ',D12.4,' > ',D12.4,/,' Value reset.');
          ENDIF
  100  CONTINUE
  104  RETURN
       END

       SUBROUTINE TRAPUF(I1,I2,D1,I3)
       COMMON/UNDERF/IUNDER
       REAL*16 D1
       IUNDER = IUNDER + 1
       I1 = 0
C      WRITE(6,*) underflow: ',IUNDER
       RETURN
       END
```

Bibliography

[1] J. A. Kerr. volume 18 of *Comprehensive Chemical Kinetics*, page 39. Elsevier, Amsterdam, 1976.

[2] R. B. Bernstein. *Chemical Dynamics via Molecular Beams and Laser Techniques* and references therein. Oxford University Press, Oxford, 1991.

[3] W. Stiller. *Arrhenius Equation and Non-Equilibrium Kinetics.* Teubner Verlagsegesellschaft, Leipzig, 1989.

[4] J. Troe. *J. Chem. Phys.*, **1977**, *66*, 4745.

[5] C. Bowes, N. Mina, and H. Teitelbaum. *J. Chem. Soc. Faraday Trans.*, **1991**, *87*, 229.

[6] R. Porter, L. Sims, D. Thompson, and L. Raff. *J. Chem. Phys.*, **1973**, *58*, 2855.

[7] M. Finkelman. PhD thesis, University of Toronto, 1976.

[8] T. C. Clark, J. E. Dove, and M. Finkelman. *Acta Astronaut.*, **1979**, *6*, 961.

[9] C. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice-Hall, Englewood Cliffs, N.J., 1971.

[10] D. Gutkowicz-Krusin. *Physica A*, **1979**, *97*, 425.

[11] C. Lim. PhD thesis, University of Minnesota, 1984.

[12] C. Lim and D. G. Truhlar. *J. Phys. Chem.*, **1986**, *90*, 2616.

[13] H. Teitelbaum. *J. Chem. Soc. Faraday Trans. 2*, **1988**, *84*, 1889.

[14] H. Teitelbaum. *J. Phys. Chem.*, **1990**, *94*, 3328.

[15] R. D. Present and B. M. Morris. *J. Chem. Phys.*, **1968**, *50*, 151.

[16] B. Widom. *J. Chem. Phys.*, **1971**, *55*, 44.

[17] B. Widom. *J. Chem. Phys.*, **1974**, *61*, 672.

[18] N. Snider and J. Ross. *J. Chem. Phys.*, **1966**, *44*, 1087.

[19] B. Shizgal. *J. Chem. Phys.*, **1972**, *57*, 3915.

[20] L. Poulsen. *J. Chem. Phys.*, **1970**, *53*, 1987.

[21] L. Shampine and C. W. Gear. *SIAM Review*, **1979**, *21*, 1.

[22] G. D. Byrne and A. C. Hindmarsh. *ACM Trans. on Math. Software*, **1975**, *1*, 71.

[23] C. W. Gear. *Communications of the ACM*, **1971**, *14*, 176.

[24] J. R. Rice. *Numerical Methods, Software and Analysis.* McGraw-Hill, New York, 1983.

[25] H. Teitelbaum. Private communication.

[26] H. Melenk, H. Moller, and W. Neun. *Impact of Computing in Sci. and Eng.*, **1989**, *1*, 138.