K/ETO-140

Technical Analysis and Operations Division

SOLGAS Refined

A Computerized Thermodynamic Equilibrium Calculation Tool

L. D. Trowbridge
J. M. Leitnaker

November 1993

Prepared by the
Oak Ridge K-25 Site
Oak Ridge, Tennessee 37831-7101
managed by
MARTIN MARIETTA UTILITY SERVICES, INC.
for the
U.S. ENRICHMENT CORPORATION
under contract USECHQ-93-C-0001

ABSTRACT

SOLGAS, an early computer program for calculating equilibrium in a chemical system, has been made more user-friendly, and several "bells and whistles" have been added. The necessity to include elemental species has been eliminated. The input of large numbers of starting conditions has been automated. A revised format for entering data simplifies and reduces chances for error. Calculated errors by SOLGAS are flagged, and several programming errors are corrected. Auxiliary programs are available to assemble and partially automate plotting of large amounts of data. Thermodynamic input data can be changed "on line." The program can be operated with or without a co-processor. Copies of the program, suitable for the IBM-PC or compatibles with at least 384 bytes of low RAM, are available from the authors.

TABLE OF CONTENTS

FIGURES

I. INTRODUCTION

A. An Overview

1. **SOLGAS defined.** SOLGAS is a computer program capable of calculating equilibrium quantities of chemical species in multicomponent systems. This program has a long history, and we have recently made modifications to make it more useful to us in our application. We hope it will be useful to others in their applications.

2. **Minimum reading requirements.** Our goal in writing this document is to make the use of the computer program SOLGAS as easy and as "user-friendly" as possible. Many of those whom we hope will be interested in this program need only to read the section on preparing files, II.A. 2. and 3., in order to start using SOLGAS. Then, after creating the appropriate files, they will be able to run SOLGAS and perform useful calculations. There are, however, some additional "bells and whistles" that even these users might wish to know about.

The user who just wants to make a series of relatively simple, perhaps "ballpark type" calculations for himself or herself may not want to wade through a lot of wordy descriptions of precise thermodynamic terminology. On the other hand, the program can make calculations as precise as the input data will permit. We have tried to accommodate the user who wants that sort of ability.

Everything else that appears in the rest of Section I can be skipped by most people that we envision might use this document. On the other hand, some students might want reasons for using the program; statements of these appear subsequently. We also owe a debt to the originators of the program we have modified. This acknowledgement is included. Some users may want to know precisely what units and conventions have been included; we have tried to accommodate them. And also, to those users of previous versions of SOLGAS, we have tried to indicate the changes that are intended to be improvements.

3. **System requirements.** The program can be operated on an IBM-compatible personal computer having at least 384 K bytes of random access memory. The monitor can be any variety. We have tried the program on DOS 3.2, 3.3, 5.0, and 6.0; it works with all these versions. If a co-processor is present, SOLGAS will use it, but its presence is not necessary.

B. Calculations SOLGAS can perform

1. **Species and amounts present at equilibrium.** SOLGAS can calculate the species and the amounts present in a complicated chemical system at equilibrium. A simple system, such as hydrogen, oxygen, and water, could be solved by any freshman student of physical chemistry. But a system containing five or more elements and a greater number of compounds is a different story. The modern computer, properly programmed, can give the answer rather easily.

However, the program is only as good as the data input. Thermodynamics is always limited by the ability of the user to supply data in the needed accuracy for species of importance. This is a caution which must always be given to users of thermodynamic calculations.

Furthermore, while the calculations can give the answer at equilibrium, thermodynamics has no ability to calculate the time necessary to achieve equilibrium. This inability and the need for data are the two serious limitations of thermodynamics.

2. **Species and amounts present at pseudoequilibrium.** Some reactions which are possible on a thermodynamic basis are not observed in practice, or are not observed on the time scale of the observation. For example, hydrogen can fuse to form helium, and helium can fuse to form carbon thermodynamically. But these nuclear reactions do not occur on earth at a measurable rate, and one does not include the possibility in studies of chemical reactions. Likewise, there are chemical reactions of interest in which the formation of the stable form of a species or compound is not possible within the timeframe during which the reaction is being studied. By leaving the unobservable species out of the list of possible end products, one can study a pseudoequilibrium. An example of this type of calculation is given in Appendix E.

3. **Amount of heat released in a reaction.** In almost every chemical reaction, there will be a release or absorption of heat. SOLGAS calculates and reports the amount of this heat.

4. **Possible temperature rise.** As presently written, SOLGAS needs two temperatures from the user. The starting temperature is the one at which, it is imagined, the particular starting components of the reaction are fed into the imagined reactor. The reaction temperature is the temperature at which the reactor is held. (In the programming, these are referred to as T_INIT and T_FINAL.) SOLGAS calculates the heat necessary to raise the input materials (the reactants) to the final temperature, calling this the preheat. It then calculates the heat of reaction at the final temperature. If the sum of the preheat and the heat of reaction at the final temperature were zero, then the final temperature would be the maximum temperature rise that could, in principle, be achieved under the other restrictions on the system (See Appendix D). Note that losses to the environment would prevent this temperature from actually being achieved, and only an idealized situation has been calculated.

5. **Gas pressure or volume change.** SOLGAS performs its calculations assuming either constant pressure or constant volume. (The default is a one bar constant pressure calculation.) When one of these two is selected as constant, the other is calculated assuming ideal gas behavior. Only the gas phase volume is considered (i.e., not the volume of condensed phases). Further, SOLGAS does not "know" that the temperature might rise because of the heat of reaction, so the pressure it will calculate will be for a constant temperature process at the user-specified final temperature.

6. **Aid in adjustment of thermodynamic data.** Frequently thermodynamic data, particularly those from different sources, do not correctly predict physical phenomena. For example, thermodynamic data for water should predict that it will have a vapor pressure of one atm. at 373.15 K. That is, one may be very certain that water boils at 100°C. If the data do not predict this, then some portion of the data is wrong and should be changed if this type of calculation is important.

Unfortunately, it is not always clear which data should be changed. For example, heat of formation, entropy, and heat capacity data on both liquid and gaseous water are required in order to calculate the vapor pressure of liquid water. The user must use judgement and perhaps, auxiliary data to decide how to make the desired calculation. Usually one tries to establish the data which are most reliable by referring to calculations in other systems frequently, and adjust the least reliable data to bring the calculation into agreement with observation. However, it must be recognized that this process is a judgement call, rather than an experimental observation.

Perhaps these last two paragraphs will emphasize to the user that obtaining data from diverse sources -- and even from singular sources -- must be done carefully, recognizing one can only be sure that observable quantities will be correctly predicted by trying the appropriate calculation (See Sec. III.I).

C. Derivation of program

The heart of the calculational program is a minimization technique for finding the least Gibbs free energy of a multicomponent system which was developed by Eriksson[1-3] and programmed for computer calculation. Spear and Besmann[4] obtained a copy of the program and made changes to permit use of either a constant volume or a constant pressure, and furthermore, provided an easier way of varying input parameters. Spear and Liau[5] at the Pennsylvania State University further developed the program for use on the IBM-compatible PC and made that program available to the present authors.

D. Computations of the program

1. **Mathematical formalism.** The actual calculation is as follows: $\Delta H^\circ_{f,298}$ and S°_{298}, the enthalpy of formation of a species in its standard reference state and the entropy of the species in its standard reference state, respectively, are inputs, as are C°_p heat capacity values as a function of temperature for all species of interest. The present program calculates for each species

$$\text{"HT"} = [\Delta H^\circ_{f,298} + (H_T - H_{298})] \tag{1}$$

and also

$$S_T = S^\circ_{298} + (S^\circ_T - S^\circ_{298}) \tag{2}$$

for each species. It then calculates, again for each species, a dimensionless quantity, G/(RT), via

$$G/(RT) = \text{"HT"}/(RT) - S^\circ_T/R. \tag{3}$$

It is this quantity, which is clearly not the standard free energy of formation of the material, that SOLGAS minimizes in this version. This treatment eliminates the need to define an elemental standard state for each element involved in the calculation of equilibrium. However, it cannot eliminate the need to include a source of each element in the calculation.

Eriksson's treatment, which we have not modified significantly, then seeks a minimum of the dimensionless (G/RT) in the system in which the amounts of each possible species are the variables. Eriksson set up a matrix of quantities related to the unknown amounts of each product species which he solved by Gaussian elimination.

2. **Units and conventions.** All input into the program is in SI units. Temperatures are in Kelvin. Compositions are in moles or gram-atoms. Enthalpies are the standard enthalpies of formation, ΔH°_f, at 298.15 K, of the material, and the units are joules-mol^{-1}. By convention, the enthalpy of formation of an element in its standard reference state would be identically zero at all temperatures. The entropies (not entropy changes) of the several materials are the values at 298.15 K and the units are joules-mol^{-1}-deg^{-1}. The heat capacities are to be given as a function of temperature, if possible, and the units are the same as those of the entropies. More than one equation might be necessary to describe the heat capacity of a substance, in which case a temperature and enthalpy of transition, in SI units, will need to be inserted. It will be obvious that the heat capacity equation(s) should be valid in a range including both the initial and final temperature of the system. The value of the gas constant, R, is taken as 8.3145107 J-deg^{-1}-mol^{-1}, the value recommended by CODATA[6], which leads to a value of 0.083145107 l-bar-deg^{-1}-mol^{-1} (The molar volume at this implied standard state pressure of 1 bar (.1 MPa) is 22.7111 at 273.15 K). Note that the R value is slightly inconsistent with the NBS[7] value, and both are slightly inconsistent with the JANAF[8] tables.). The boiling point of a substance is taken as the temperature, in Kelvin, at which the partial pressure of the substance is 0.101325 MPa, exactly.

"Species" will be referred to in two ways: as the various chemical substances to be examined as potential products during the minimization calculation, and as substances being used as starting materials. To differentiate between these two uses, we will always refer to the latter materials as "input species," which are a subset of "species."

There are three specific temperatures with which the user will come in contact. The first is the reference temperature for the thermodynamic data, 298.15 K, which is embedded into both the program and the thermodynamic data, and is unlikely to ever change. The other two temperatures are the starting temperature, T_INIT in the programming, and the reaction temperature, T_FINAL in the programming. The starting (or initial) temperature is input with the rest of the thermodynamic data, and is the temperature at which one imagines the input species are fed into a reaction chamber. The reaction (or final) temperature is the temperature at which reaction takes place, which may be either higher or lower than the initial temperature.

3. **Sources of error.** Errors in the input data will obviously result in errors in the output. By using a standard data source formulation (described later as an option), one can eliminate an always troubling source of error: inadvertent inclusion of transposed or miscopied numbers. SOLGAS can have other problems in obtaining a correct solution, and it is these that we intend to address in this section.

There are three indications to the user that a correct answer has not been obtained: (1) The program prints on the screen "Equilibrium has not been achieved"; (2) a message that there is a violation of mass balance; and (3) an underflow error is announced. Occasionally, SOLGAS announces that "the small Y values may not be correct." Usually, the latter is of small concern. We discuss the source of errors and some corrective measures in the following text.

SOLGAS can be prevented from reaching an answer, as Eriksson originally noted, if a singular matrix results from the input data. For example, when a simple system is treated in which two constituents react completely to form a compound, such a case occurs. Eriksson noted that the solution is to add a small amount of one of the constituents, or to add a small amount of another component that will slightly unbalance the stoichiometric reaction. In some cases, the failure to arrive at a solution can be overcome by using the option that allows the user to estimate the amounts of the equilibrium species (One must have, of course, knowledge from previous calculations or chemical intuition, perhaps from some results which bracket the troublesome calculation).

Another case in which no answer is reached involves the lack of a gas pressure in the system. Most frequently, a small amount of an inert gas, e.g., 0.001 moles of Ar, in the input materials will be sufficient to overcome this problem without affecting significantly any of the output. However, if a condensed phase solution is present, this restriction is lifted.

In some cases, SOLGAS reaches an apparent solution and prints an answer which does not agree with calculations very near to it in composition; sometimes this apparent solution is clearly wrong. During intermediate calculations of a particular system, SOLGAS will set the composition of some species identically equal to zero if the calculated composition of that species gets low enough. It appears that in certain cases, SOLGAS has done this at too early a stage in the solution. Once a composition has been set to zero, SOLGAS apparently does not go back to re-examine its potential for appearance. Thus, the system being examined is no longer the system entered through the input, but rather a reduced subset of the system. This problem originates in a section of code which we have not modified, and often manifests itself in the form of a potential floating point underflow error in an exponential calculation. The calculation is not attempted, but instead the exponential is set to zero and a flag is set which adds a message to the output table that such a condition was detected, warning the user to carefully check the output in that particular run.

The mass balance error, mentioned previously, has been flagged by summing the mass of each element in the final mix of products and comparing this result with the initial input. The user reads, "Warning: Mass balance not preserved in this run. Examine results carefully!" The reader also receives notice of the extent of non-balance.

We have not detected any errors which are not flagged by the messages listed above. Avoiding these errors can sometimes be done, after the fact, by changing the initial composition somewhat and by manually entering estimates for the results (i.e., using menu option "2" rather than relying on the computer's estimate) or in some cases, changing the order of species putting the least important species last in the list of species to examine. Sometimes the desired answer can only be obtained by interpolating between satisfactorily computed answers. Fortunately, these occasions constitute less than 1% of the calculations performed.

4. Restrictions. Only ten elements are allowed as input to SOLGAS, and only 99 species can be considered as potential products. Furthermore, only 20 input species can be considered. The maximum number of mixtures is 20, and the minimum number of gaseous species is one. If the user attempts to violate one of these restrictions while entering data, SOLGAS stops and prints a message indicating the nature of the violation. The user must then adjust his data file in order to use SOLGAS. One limit that isn't checked but is imposed on the data input, is that the maximum length of any single line in the input file is 500 characters.

If the user wants a material considered as a component of the equilibrium mixture, there must be a source of the elemental constituents of that material. The converse is not true; however, a material for which there is no source in the input materials can remain in the species list but will not have a contribution in the output list.

The program assumes that the solids occupy no volume. In specifying a constant volume, one can therefore consider only the volume of the gas of interest.

5. **Output.** SOLGAS reports in tabular form the species considered in the system, the initial and final amounts of each, and the equilibrium (final) pressure of each gaseous species. It also reports the heat required to raise the input species from a stated starting temperature to the stated reaction temperature, the heat of reaction, and the algebraic sum of the two heat values. An example of the output table is given in Appendix B.II.B. If condensed phases were available, but not present in the final product, SOLGAS may list the free energy change necessary to make the next most stable species appear. This can be useful in adjusting thermodynamic data.

The report for an individual run is clear, convenient, and useful. However, for significant numbers of runs, a different tabulation would be more convenient. A more convenient tabulation is described in Section V.

E. Modifications to SOLGAS

SOLGAS as documented here is considerably altered from the version we originally obtained. A brief description of changes we have made is stated in the following text.

1. **Readability.** We have changed the interactive portion of SOLGAS so that the user responds to more familiar terms. For example, rather than offering to change "NPKT" or "B", we use "number of points to calculate," and "quantities of starting materials." Thus it is hoped that the casual user will find the program easier to use.

2. **Elemental specification eliminated.** In the formulation of the original SOLGAS, it was necessary to enter elemental species explicitly in their reference state. In many of the reactions we have studied, the concentration of elements in the final product was negligible which sometimes led to computational difficulties. Furthermore, in some cases, it is desirable to actually exclude the elements from consideration. These cases arise when kinetically, the elemental state is not accessible during the time of interest. One can study a reaction's thermodynamic pseudoequilibrium by eliminating the elemental state of one or more elements from consideration (See Appendix E). We eliminated the need for specifying the elements.

3. **Specifying degrees of freedom: DOF.FOR.** The original SOLGAS used the elements as a way of calculating the number of thermodynamic degrees of freedom of a system, a quantity necessary to calculate thermodynamic equilibrium. Almost invisible to the user is a subroutine in SOLGAS, DOF.FOR, which calculates the number of components in the system. For example, one might be studying the hydration of $CaCl_2$, in which one would have $CaCl_2$, along with the hydrated solid, liquid, and gaseous water. Obviously, there would be four elements, but at ambient temperatures, only two components. The user only observes the result of this calculation in those cases in which the number of components is smaller than the number of elements. In such a case, the screen displays the result of the calculation which gives the assignment of species' components. Other than this screen display, the user has no knowledge of the change in component assignment—and maybe no need to know.

4. **Alternative thermodynamic calculations.** We have eliminated the ability to specify the species' thermodynamic data in the form of ΔG° as a function of temperature. ΔH°_{298}, which we use, can be derived from ΔG°; S°_{298}, also used, is virtually always available or can be easily estimated. Earlier SOLGAS versions also had the ability to calculate equilibrium by means of enthalpies of formation and free energy functions. This feature is not available in the present version.

5. **Heat calculations.** The option to eliminate the heat calculations is no longer available; one can always ignore the results.

6. **Automated inputs.** The present version of SOLGAS permits one to easily enter a large number of input species quantities, along with variations in temperature and either pressure or volume, without laboriously entering these "by hand." This choice has value particularly in those ranges of a system in which significant changes of an equilibrium species occur with relative small changes in input conditions. The procedure, described later in detail, is to construct a second input data file in a spreadsheet in which one specifies varying starting conditions for the desired series of SOLGAS runs. (See Section III.G.)

7. **Revised format for data inputs.** A significant change has been made in the format by which data is entered. All data for one species is entered on a single line. These data entries include a name, chemical composition, and thermodynamic data, such as enthalpies, entropies, heat capacities and any phase transitions. The impetus for this change is four fold: (a) Gathering all the information about a compound into one entity (e.g., a line or record) makes the file more readable, (b) subsequent modification of the data file is straight forward, (c) putting data about species into a library would be facilitated in this way, and, (d) once a species is correctly entered into a library file, errors in copying or transposing should be largely eliminated. (See Section II.A.2 and Appendix F.)

8. **Revised format for data outputs.** Printing out activities of gases have been eliminated. The present formulation of SOLGAS does not use activity coefficients, and the activities and the ideal gas pressures, which are printed, are identical.

9. **Calculation of species composition.** Species composition is now entered in a more natural way. Instead of specifying that water has two gr-atoms of H and one of O, one writes the formula H2O, and SOLGAS calculates the appropriate composition. Such things as CCl4 (naturally one must use the letter "el" rather than the numeral "one" to indicate carbon

tetrachloride) and Fe0.945O (being careful to use the letter "oh" rather than a zero) seem more easily used.

10. **Error corrections.** A number of errors in SOLGAS have been corrected. One was a procedure which incorrectly calculated the entropy of transition for a phase, with the result that any data set containing a species with a phase transition would yield an incorrect answer above the transition temperature of that species. A second was that, as originally programmed, phase transitions were assumed to be above 298.15 K. Such transitions may now be at any temperature.

11. **Program portability.** Some modifications have been made to allow compilation on other computers. New COMMON storage areas were added to preserve local variables' values between calls to subroutines. SOLGAS was written on the assumption that these values were saved, and in fact, Ryan/McFarland FORTRAN and Microsoft FORTRAN did save these values, although the R/M manual says they will not be saved. A time-honored "trick" by which text is read into numeric arrays was removed. This necessitated creation of separate COMMON for the new CHARACTER variables, since characters cannot be mixed with numerics in COMMON. SOLGAS has been compiled using several compilers: R/M FORTRAN and MS FORTRAN for IBM-compatible PC's, and VAX/VMS FORTRAN on VAX 8600. A few modifications were necessary to get the PC version operational on the VAX, but identical results were obtained on all systems.

12. **Error trapping.** File existence checks were added to trap some common errors related to use of duplicate or misspelled file names. A large number of protections were built into the data entry routine against various types of invalid data files.

13. **Inconsistent array bounds.** These were rendered consistent, and a limit on the number of starting materials (of 20) was imposed. Earlier, various arrays allowed 10,20, and 99 starting materials, and some out-of-bounds data storage occurred as a result.

14. **Auxiliary programs to treat output data.** A program named "MASSAGE" has been written to convert a file containing many output data tables from SOLGAS and rearrange them into a spreadsheet-compatible file which can be sorted, pruned, selected, and plotted. This program eliminates a large amount of laborious hand tabulations with the attendant chance for error (See Section V). Another program, ADIABAT, calculates the adiabatic temperature rise when species are reacted (See Section IV).

15. **Changing thermodynamic data "on line".** In order to adjust thermodynamic data to make these agree with physical observations, an option has been added to make the adjustment while in SOLGAS, rather than exiting the program, changing the file, and re-entering SOLGAS.

II. GETTING READY TO USE SOLGAS

A. The thermodynamic data file

1. **Reason for the format.** To run SOLGAS, one must first prepare a file containing the thermodynamic data for the several species under consideration in the problem at hand. This is done with any text editor, word processor, or spreadsheet that can create a file in ASCII (i.e., plain text) form. The original SOLGAS data input was on 80-column data cards. Use of SOLGAS on the IBM-compatible computer required an input file for each run, and the file paralleled the input cards for the "mainframe computer" originally used. Each separate problem required a data file and, although often an old file could be changed somewhat to produce a new file, a considerable amount of manual bookkeeping was still required of the user, all with the attendant chance of error. Because of the restricted length of the input card, names and composition indications were on one card, enthalpies and entropies were on another, heat capacities yet on another, and all had to be arranged in a specified order for the computer.

The input file that was required on data cards, and then translated into an ASCII file for the personal computer, has been simplified. Because the requirement of a fixed line length has been lifted, one can rather easily put all the data relating to one compound on a single line. Furthermore, the computer can be used to compute compositions from the usual chemical formula of a compound and do other bookkeeping chores formerly required of the user. One is limited, as mentioned previously, to 500 characters in a single line, though it is rare, in our experience, to need such a long line. Not all word processors allow this length of line. Both EDLIN, the early DOS line editor, and EDIT, the newer DOS editor, only allow 256 characters on a line. WordPerfect 4.0 allows 250 characters. WordPerfect 5.1 formats its line length based on inches; 16.5 in. is the maximum length of line, and with 10 "x's" per inch, one can have approximately 165 characters on a line. Word Star can handle more than 500. Most spreadsheets could readily handle 500 characters, although one would pay a severe penalty in space used to save a moderately long entity with widths in the vicinity of 500 characters.

2. **Description of the file.** An example file is shown in Fig. 1, and will be used as a means of explaining the general requirements. Such a file can most easily be prepared by means of a spreadsheet but can also be prepared in any text editor which allows saving in an ASCII file, as will be shown subsequently. Incidentally, some spreadsheets/text editors may "do you a favor" when they save your file. Quattro Pro, Lotus 1-2-3, and EXCEL are covered in this document[13]; if your file isn't read correctly from your spreadsheet or text editor, check to see if the file produced actually corresponds to the description in this section. For example, we found some programs inserted tab characters where we had expected spaces.

Fig. 1. Example of a SOLGAS data file.

```
TITLE Test SOLGAS data file
    This is the sort of thing that one might use to investigate the reaction of chlorine with
    fluorine.

T=298.15

    *G      Ar      Ar         0       155     20.8    ;
    *G      Cl2     Cl2        0       223     33.9    ;
     G      Cl      Cl      121302     165     21.8    ;
    *G      F2      F2         0       203     31.3    ;
     G      ClF     ClF     -50292     218     32.1    ;
     G      ClF3    ClF3   -158866     282     63.8    ;
     G      F       F        79390     159     22.7    ;
```

i. Title line

"TITLE" must appear at the beginning of the first line; the next 80 characters on that line will appear as the title in SOLGAS output files. Any number of remarks lines (which will be ignored by SOLGAS), may follow the title line.

ii. Starting temperature line

"T=" or "T =" must begin the next "working" line; the value following it is the starting temperature of the mix of input species in kelvin. (For the convenience of EXCEL users, "T,=" is also accepted.)

iii. Data lines

All the remaining lines are data for the chemical species to be considered in the thermodynamics problem. The data for each species appears on a single, perhaps very long, line. The format of the line is somewhat free-form, with data fields separated by a comma or space (nothing else is valid). The data need not be lined up in columns as shown in the example of Fig. 1, although that sort of organization may make the data set more readable to the user.

The "*" as the first character of the line indicates that this species is a starting material, an input species. Lack of an asterisk indicates that the species is not part of the starting mix, but will be considered in the calculation as a possible product.

A "G," "M," or "P" must be the next letter encountered (after the asterisk, if it is present). However, blank spaces are allowed between the asterisk and the "G," "M," or "P." These letters designate the material as "Gas," "Mixture," (which could be a liquid or solid solution) or "Pure" condensed phase material, either liquid or solid. If a mixture is present, you must designate what you are referring to by following the "M" with a one or two digit number. (Only 20 mixtures are allowed.) "M1","M,1", "M 1", "M01" are all allowed formats in the data file.

Species must appear in the following order (which is the same as in previous versions of SOLGAS):

(1) all gases.
(2) all mixtures, in the mixture order, that is, all species in "M1" before any in "M2," and so forth.
(3) pure substances.

The next field is the species name. Only ten characters of a name will be retained by SOLGAS, although one could use a longer name if desired. Obviously, spaces or commas would not be allowed in the name. In Fig. 1, we have unimaginatively used chemical formulae for the names, but this is not necessary.

Next is the formula field, which requires a bit more explanation. SOLGAS uses the formula to calculate the composition much as a chemist normally would. Thus, the formula must be entered in a simple format, without spaces or commas. The format is: element-subscript-element-subscript... etc. “Element” means the one or two-letter chemical symbol for that element. The first letter must be capitalized; the second, if any, must be lower case. For example, for chlorine, one must use “Cl,” not “CL” or “cl.” Subscripts must be positive numbers, and decimal points are allowed, but signs and an exponential format are not. If the subscript would be “1,” it may be omitted. An element must appear no more than once in a formula, and parentheses or other structure-designating additions are not allowed. Some valid formulas are: H2O, C3F8, and Ca.8Mg0.2CO3; some invalid ones are CF3CF2CF3 or Ca0.9995Mg5E-4CO3. There is a limit of 30 characters to describe the formula.

The next two fields are, in order, ΔH°_f and S°. The ΔH°_f value is the standard enthalpy of formation of the species, in $J\text{-}mol^{-1}$, at 298.15 K. The S° value is the entropy of the species, in $J\text{-}mol^{-1}\text{-}deg^{-1}$, at 298.15 K in its standard, reference state.

One next inputs constants of a heat capacity equation for the species, again in $J\text{-}mol^{-1}\text{-}deg^{-1}$. Up to five parameters, a, b, c, d, and e, in order, may be used, according to the C_p equation

$$C_p = a + bT + cT^2 + dT^{-2} + eT^{-3}. \qquad (1)$$

For species of interest at high temperatures, the data most often available are enthalpies in tabular form. It may be necessary for the user to fit the data into an appropriate equation, and differentiate with respect to T (K) to produce an appropriate C_p equation (since C_p is defined as the derivative of enthalpy with respect to temperature at constant pressure). A value for "a" must be provided, but other parameters will be read as "zero" if not explicitly entered. For example, a more usual multi-term equation would be

$$C_p = a + bT + dT^{-2}. \qquad (2)$$

If a value for "d" were entered in the file, a "zero" for the parameter "c" would be required, but nothing would be required for "e.". In using Equation 1, we are following the previous versions of SOLGAS.

Multiple C_p equations are allowed. These may be needed to fit different regions, either because of a phase change or simply as a device to splice equations for different regions to obtain a better fit to data. If more than one equation is needed, an additional set of numbers is needed after the then-required "e" variable in the previous set. The values required for the second and subsequent sets are shown in the correct order:

(1) the enthalpy of transition, in J-mol^{-1}, at the

(2) temperature of transition, in kelvin, and

(3) C_p for the new region, consisting of a required "a" parameter, followed by up to the four additional parameters of Equation 1.

Instead of using multiple C_p equations, one could use multiple species. In many cases, one would have to derive an appropriate ΔH°_f at 298.15 K (by integrating $C_p{\cdot}dT$) and an S° at 298.15 K (by integrating $[C_p/T]{\cdot}dT$).

End of data line:

Finally, a ";" indicates that all data is complete on that line. Following the ";" one may enter any desired remarks, which will be ignored by SOLGAS. For example, one might enter notes on the source of data for that species.

Most plausible violations of the above formats will result in an error message indicating the type of format error and listing the offending data line. Data lines are read until an end of file is encountered. The limits imposed by SOLGAS as to the maximum number of elements (10), mixtures (20), and species (99), the minimum number of gases (1), and also the limitation on the number of characters in a line, 500, apply.

3. Preparation of the file. At least three methods may be used to create the data file described in II.A.2; by use of a spreadsheet, from a database package, or in a word processing package.

When constructing a file, remember that SOLGAS looks for either a comma or a space as a delimiter between fields, and looks for the ";" as an indication that the data for that species is complete. The particular method of construction is immaterial.

B. An optional file to vary starting conditions

1. **Purpose of the file.** In some cases, a user will wish to compute changes in output as a function of some input parameter. (The reader may wish to postpone reading this section until after actually using SOLGAS, or at least until after having read Section III, "Using SOLGAS.") Particularly in complicated cases, a plot of such data is much more immediately revealing than columns of numbers. Such plots can be readily generated from columns of numbers in spreadsheets. Two features have been designed to facilitate such plots. The first feature is the additional input file to vary starting conditions, which is described here. The second is a computer program (appropriately called MASSAGE) which "massages" lengthy output produced by SOLGAS into a form which is easily accepted by a spreadsheet. In the spreadsheet, the data can be pruned and selected to show the results to best advantage. (See Section V.)

2. **Description of the file.** The file is simply an ASCII file in which each row of data defines the starting conditions for a single SOLGAS calculation. The first number on a data line is the final temperature of the run. The second number contains the (constant) system pressure, entered in bar, or the (constant) gas volume, entered as a negative number in liters. The remaining numbers contain the quantities, in moles, of the starting species. This quantity data must appear in the same order as the input species are listed in the associated SOLGAS input data file. For example, the Fig. 1 data file would require the order to be "Ar," "Cl_2," "F_2."

3. **Operation of the file.** When the user selects CHOICE 12 (the option in the interactive portion of SOLGAS which allows the use of such a file), SOLGAS reads lines from the control file one at a time, solves the problem for the temperature and starting quantities specified, and automatically saves the output table to the previously designated output file. Control data lines are read and processed until an end-of-file is reached, at which time the program returns to the main menu in SOLGAS. The successive output tables may be scanned individually or treated with the program called MASSAGE. (See Section V.)

4. **Preparation of the file.** The file can be prepared in two different ways: either (a) in a spreadsheet or (b) in a word processor or text editor. Most frequently, only one or two of the input parameters will be varied during a run, although there is no necessity for this. Filling a column of constant values is particularly easy in a spreadsheet, as is incrementing a value by a constant amount. Hence, there is a predilection on the part of the authors for preparing these control or parameter files via a spreadsheet. One then saves the file in an ASCII format, either space or comma delimited.

5. **An example file.** Fig. 2 is a parameter control file which will be used in Appendix B. The first column, as mentioned above, is the temperature; a quick glance reveals that this is the only parameter changing. The second column in the pressure in bar, and the user will recall that 1.01325 bar is defined as 1 atm. The associated data file (see Appendix B, Fig. B.1) contains two input species, Ar and $H_2O(l)$, in that order. The next column, thus, contains the moles of Ar input in each problem. The last column contains the moles of water.

Fig. 2. An example parameter control file. (To be used in Appendix B.)

```
373.0        1.013        1E-6        1.0
373.2        1.013        1E-6        1.0
373.4        1.013        1E-6        1.0
373.6        1.013        1E-6        1.0
373.8        1.013        1E-6        1.0
374.0        1.013        1E-6        1.0
```

III. USING SOLGAS

A. Introduction

SOLGAS is designed to be "user friendly," and, to the extent that the authors have succeeded in accomplishing this goal, the user won't need much of this section. This section will take the reader through the questions asked by SOLGAS and will give suggested answers along with the consequences of choices that might be made. In addition, a number of comments will be made about the information to be displayed on the screen.

B. Definitions: our use of names for input and output files.

The authors have used specific extensions to file names to indicate what the file is designed for. Since the program operates on the IBM-compatible personal computer, the rules of naming will be familiar to nearly all PC users - no more than eight alphanumeric characters in the file name and no more than three in the extension (see the disk operating system instructions for invalid characters in the file name). For the thermodynamic data input file, we will always use the extension .DAT. MY_FILE.DAT is used here as an example. The extension .CTL is used for the parameter control file, e.g., MY_FILE.CTL. (This type of file was described in Section II. B.)

Early in any session with SOLGAS, the user is asked for the name of an output file; for such files we will use the extension .OUT, e.g., MY_FILE.OUT. (In some cases, when a variety of case-by-case variations are going to be studied, the authors have used a numerical extension for the output file, e.g., MY_FILE.001, ...002, etc.)

After using the parameter control file, MY_FILE.CTL, one would often wish to use MASSAGE to rearrange SOLGAS's results (which would be in MY_FILE.OUT) in order to make it more readable to a spreadsheet. For such files, we will use the extension ".PRN."

MASSAGE uses MY_FILE.OUT and generates another file, MY_FILE.PRN, which the user can then import into a spreadsheet.

There is no SOLGAS requirement that the extensions mentioned here must be used. They are used by the authors both to tie together a set of related files and also to use as examples for the reader.

C. Starting SOLGAS.

SOLGAS is an executable compiled file whose extension is .EXE. It runs under DOS. Hence, it is only necessary to type "SOLGAS" in either upper or lower case, followed by <RTN>, in order to start the program (The symbol <RTN> means to press the "return" or "enter" key). The authors customarily put SOLGAS.EXE and all the data and parameter files into one directory which makes interaction somewhat easier. However, this is not necessary. Paths to other disks or directories are acceptable usage for all file names.

After typing "SOLGAS" and <RTN>, the user is asked for the name of the input file. If one were following the definitions in Section III.B, one would type "MY_FILE.DAT." The next query is for an output file name, which could be "MY_FILE.OUT."

SOLGAS reads the input file, MY_FILE.DAT, and reports what it found (number of species, elements, mixtures, etc.). Occasionally, SOLGAS will find that there are fewer "linearly independent" components than elements in the input file. For example, if one were studying the boiling of water (using only liquid and gaseous water, as will be done in Appendix B), there is obviously only one component in the system, although there are two elements. In this case, SOLGAS "tells" the rest of its program that there is only a single component in the problem whose formula is H_2O. By "independent," we mean components whose quantities can be independently varied. In the case of a system containing only phases of H_2O, one cannot independently vary oxygen and hydrogen quantities.

Next, SOLGAS tells the user the starting temperature, which was included in the input data file, and then asks for the reaction temperature. A discussion of these two temperatures was given in Section I.D.2. After the reaction temperature is input, SOLGAS presents the user with a screen of interactive choices, Fig. 3. These choices are discussed in Sections D through H.

Fig. 3. Screen of interactive choices in SOLGAS

```
[1]  Computer selects a starting estimate.
[2]  Manually select a starting estimate.
[3]  Change no. of pts. to calculate, and quantities of starting materials.
[4]  Change quantities of starting material only.
[5]  Recalculate with same data, pts., and quantities.
[6]  Change the reaction temperature.
[7]  Change value of fixed pressure or volume.
[8]  Change to a new system.
[9]  Exit program.
[10] Save the last calculated result.
[11] Save the last series of results.
[12] Read run data from a parameter file.
[13] Change thermodynamic data of a species.

     Select an Option:
```

D. Quitting SOLGAS: CHOICE 9

CHOICE 9, "Exit Program," is always available at the screen of Fig. 3. Nothing further is saved from the calculations, and the computer returns to the directory from which SOLGAS was entered. In this, CHOICE 9 is similar to using <CTRL>+<Break> or <CTRL>+C.

E. Initial interactive choices

The first time the screen of 13 choices is presented, the user can reasonably make only one of three choices: CHOICE 1, CHOICE 2, or CHOICE 12. The other choices typically assume that some calculation has been made and something else is desired. The program does not "bomb" if another choice is made, but doing so seems pointless. CHOICE 12 is a special case that is described below. Choices 1 and 2 are described as follows:

CHOICE 1. This is the choice most often made. After some questions are answered by the user, SOLGAS selects a starting estimate of the equilibrium quantities of products. The calculation proceeds satisfactorily most frequently.

CHOICE 2. "Manually select a starting estimate" indicates that, in addition to starting quantities (See Section F below), the user will be asked for an estimate of the amounts of all species in the final result. Occasionally, a user-provided initial estimate of the products may prove helpful in obtaining a solution. SOLGAS uses an iterative numeric process involving Gaussian elimination to solve a matrix. Because of the way starting estimates are made in CHOICE 1, it appears the program can fail to converge to a solution. In rare cases, it may converge to a fallacious solution. (See discussion of "Sources of error" in Section I.D.3.) For example, on a few occasions during a series of runs that slowly scanned a range of compositions of initial reactants, the authors observed a run in the middle of the series that gave results that deviated significantly from those around it. In the strange run, many species' compositions had been set identically equal to zero, and reactions with strongly negatively free energy changes among remaining species didn't occur. Re-running these cases using manual input of plausible final values for each constituent yielded sensible results consistent with other runs in the series. The user must experiment to obtain a desirable result.

F. Other data following either CHOICE 1 or CHOICE 2 required by SOLGAS.

SOLGAS first asks how many points one wishes to calculate. Any positive number less than 100 is valid (By the term "point," we mean a single SOLGAS calculation). The user is then informed of the identity of the input species and is asked to indicate how the composition of the various species vary. The choices are "constant," "varies irregularly," and "varies by constant increment." For example, if four points were to be calculated, species A and B might have constant values, e.g., 1 mole; species C could vary irregularly, e.g., 0.5, 0.5, 1.0 and 1.0 moles; and species D could have a regularly varying composition, e.g., beginning at 0.25 moles with regular increases of .25 moles, ending at 1.25 moles for the final point.

If CHOICE 2 had been selected, after asking for information relating the amounts of the various input species, SOLGAS then goes on to ask for a single estimate of the final equilibrium quantity of each of the species to be considered in the problem. SOLGAS does the first calculation with this information, goes to the second calculation, and so forth without coming back for any subsequent estimate. Data from the first calculation remain in memory, and SOLGAS has the advantage of this information. However, if the user is doing a number of calculations in troublesome areas of the system, a different estimate for each point may well be warranted. In this case, the user should respond with "1" when asked how many points are to be calculated.

G. An automated data entry, CHOICE 12.

"Read run data from a parameter file" assumes SOLGAS is to select initial estimates of the output (similar to CHOICE 1), and the program takes each set of starting condition information line by line from a previously created run control file (Section II.B), calculates the results of each, and stores them in the output file (e.g., MY_FILE.OUT). The number of data lines that can be read and processed is limited only by the capacity of the storage device being used. Processing (or "massaging") of this mass of output data was mentioned briefly in Sec. II.B and will be described in more detail in Section V.

H. Saving calculated data.

SOLGAS does not automatically save the results of calculations, except in the case of CHOICE 12. Two choices are available to save data which differ in operation. These are described below.

CHOICE 10, "Save the last calculated results," will write the output results of the just completed single calculation to the output file, e.g., MY_FILE.OUT. CHOICE 10 saves data that is in RAM. If one had just calculated four points and only then chose CHOICE 10, only the last of the four points would be saved to MY_FILE.OUT because these data are still in memory. One could add the results of another calculation to the same output file if CHOICE 10 is chosen each time a calculation is completed.

CHOICE 11, "Save the last series of results," saves results of multiple-point calculations. The number of results that SOLGAS saves with this option is determined by the number by which one answered the request "Input No. of points to be calculated...," for that series of calculations will then be re-run and saved. Thus, if a user knew ahead of time he was going to run four cases, it could be more efficient to indicate that fact for the number of points and then choose CHOICE 11 at the end. If running additional cases depends on the results of the present case, then CHOICE 10 might be more efficient.

I. Other choices

CHOICE 3. "Change number of points to calculate and quantities of starting materials" is similar in behavior to CHOICE 1. However, the initial estimate of the equilibrium composition is not made in the same way. The results of the previous calculation are used for an initial estimate of the solution to the current calculation. This will be transparent to the user except in unusual cases.

CHOICE 4. "Change quantities of starting materials only" tells SOLGAS to calculate the same number of points as its most recent series, but to change the initial quantities of input species. If CHOICE 4 is selected without telling SOLGAS how many points to calculate in a previous choice, the user is told there are zero points to calculate, and SOLGAS cannot calculate an equilibrium even if data are entered.

CHOICE 5. "Recalculate with the same data, points, and quantities" is an opportunity to repeat the immediate past calculation(s). This may be useful if one has changed temperature, pressure, or volume, or even to simply see the top of a table that has scrolled off the screen. (Compare with CHOICE 3.)

CHOICE 6. "Change the reaction temperature" allows the final temperature to be changed.

CHOICE 7. "Change the value of the fixed pressure or volume" is a choice that could be made before Choices 1 or 2. SOLGAS initially assumes that calculations will be run at a constant pressure of 1 bar. Hence, one might initially wish to set the pressure at half a bar. If one wished to "operate" at a fixed volume, e.g. at 2.5 liters, one would enter -2.5; the negative sign tells the program that 2.5 is a volume in liters. Incidentally, it is probably worthwhile to note that the volume of condensed phases is assumed to be zero by SOLGAS. At moderate pressures, this causes no problem but, at high pressures, one might be misled.

CHOICE 8. "Change to a new system" essentially sends SOLGAS back to the beginning to get a new data file and a new output file. All the initial question are then asked; it is as if SOLGAS were started anew.

CHOICE 13. "Change thermodynamic data of a species" is, as suggested, an internal method of changing one or two data points for ΔH_{298} or S_{298} or the "a" value (only) of C_p in cases where the user wishes to bring thermodynamic data into agreement with physical observation (See Appendix B for a case in which one might wish to change data). If a large number of changes were to be made, it might be easier to reconstitute the entire input file, MY_FILE.DAT. At present, the user needs to keep track of data that have been changed.

When such changes are made SOLGAS "remembers" and prints the value of ΔH_{298}, S_{298}, and the "a" value of C_p for each species of the system, both on the screen and in the saved output file.

The user can try all these choices. SOLGAS has never "locked up" the computer in our experience, and trying the choices may be more effective than reading about them.

IV. Automated calculation of adiabatic temperature rise: Use of ADIABAT.

A. Description of ADIABAT.

ADIABAT.EXE is a utility program which repeatedly runs SOLGAS in order to find the adiabatic temperature change of a system. SOLGAS is designed to solve for equilibrium under isothermal conditions; in some cases, it is desirable to solve for equilibrium at adiabatic conditions (i.e., without gain or loss of energy). ADIABAT was written to fill this need.

B. Reason for use.

When reactive species are mixed, release of heat in an approximately closed system could cause the temperature to rise in a dangerous way. Even if the system is quite small, the energy that could be released in the form of heat can have important kinetic implications. Even if "only" studying the efficiency of the combustion of fuel, the ability to calculate the limiting heat available can be useful.

Usually, ADIABAT does more quickly what the user can readily do by hand. As one sees how ADIABAT is organized in the next section, one will see that the process can be accomplished a run at a time by the user; ADIABAT is just more efficient.

C. Organization of ADIABAT

ADIABAT makes use of the fact that SOLGAS calculates three important heat quantities. First, it calculates the heat necessary to raise the potential reactants of a system from the starting (or initial) temperature to the reaction (or final) temperature, called the preheat. It then calculates the heat of reaction at the reaction temperature. Finally, it algebraically combines the two calculated heats to get a final, total heat.

Now if the final total heat is zero, in principle, the reaction temperature is the highest temperature that can be reached by the reactants making up the system at the specified conditions.

Thus, in order to provide the adiabatic final temperature, /)IABAT varies the final temperature, seeking a temperature at which the total heat is within 0.02 J of zero. Of course, the difference between the final and the initial temperatures is the temperature rise.

D. Preparation of the control file

ADIABAT is the controlling program which runs SOLGAS in order to accomplish its mission. It is set up to run at a user-specified constant pressure or volume and also arranged to allow input of multiple points, since most frequently one will want the information over a range of compositions.

The structure of the control file for ADIABAT is similar to that for the CHOICE 12 control file, with the addition of one more parameter. To accomplish its task, ADIABAT varies one input variable, either the reaction temperature or the starting quantity of a single species.

Each line of the control file defines a single calculation. The entries on each line are as follows:

Control variable. The first parameter must be an integer. If its value is zero, that indicates that temperature will be varied until the "Total Heat=0" point is found. If its value is 1,2,3, etc., that indicates that starting species number 1,2,3... (as determined by the order in which it is listed in the associated data file) will be varied until the net heat is zero.

Temperature. This will be the reaction temperature for the run. (If temperature is to be varied this will be the starting point for the search.)

Pressure or Volume. A positive number indicates a run at constant pressure, a negative number at constant volume (in liters).

Quantities. The remaining values are the starting quantities, in moles, of the input species in the associated SOLGAS data file. If one of these parameters is to be varied, the number entered will simply be the starting point for the search.

Fig. 4. Example of a control file to run ADIABAT

```
1     1200   1.013   .5400   .4600   1E-7
0     1000   -44.0   .5400   .4600   1.43E-3
```

In the example in Fig. 4, the first line in the control file "tells" ADIABAT (which tells SOLGAS) that the reaction temperature is 1200 K, the pressure is 1.013 bar, and the compositions are 0.5400 moles for species A, 0.4600 moles for species B, and 1 x 10^{-7} moles for species C (Note that SOLGAS is going to get the starting temperature and thermodynamic data from MY_FILE.DAT.). The "1" at the beginning of the file tells ADIABAT that the first starting material, species A with 0.5400 moles, is the parameter to vary until the total heat at a temperature of 1200 K and a pressure of 1.013 bar is zero. Note that the order of the species A, B, and C must be identical to the specification and order of MY_FILE.DAT.

The second line of the control file in Fig. 4 has some differences. The reaction temperature is only 1000 K, the volume is fixed at 44 liters, the compositions are 0.5400, 0.4600, and 1.43 x 10^{-3}, and the "0" at the beginning indicates that the temperature is to be varied until the total heat is zero.

The control file can be prepared via any mechanism which will save the file in an ASCII format and is either space or tab delimited (comma delimited will not work).

E. Preparation to run ADIABAT

ADIABAT assumes that the following files are in the default directory: COMMAND.COM (which must be copied into the default directory from the DOS root directory), SOLGAS.EXE, ADIABAT.EXE, the control file described in Section C, and the data file (MY_FILE.DAT as used here). Data files could be specified by the DOS PATH command and located elsewhere.

F. Operation of ADIABAT

To start the program, one merely types ADIABAT followed by <RTN>, and one is asked for the thermodynamic data file, the control file and a results file. SOLGAS asks for the thermodynamic data file.

ADIABAT has a convenient feature. If one names the files carefully the typing of additional file names is eliminated. If one would type

```
ADIABAT D:\THERMO\MY_FILE <RTN>
```

ADIABAT would assume all the files were named MY_FILE, and the extensions were .DAT for the thermodynamic data file, .DRV for the ADIABAT control file, and .PRN for the output file. In the above example, ADIABAT would assume that the files were located in drive D: in subdirectory \THERMO.

G. Treatment of output

Output from ADIABAT is an ASCII file, assumed here to be named MY_FILE.PRN. Extension .PRN is useful in importing the data into a spreadsheet, since this extension will cause its file name to be listed as a choice during the importation process in both Lotus 123 and Quattro Pro.

The output is given, line by line, as:

Run No., Temperature, Pressure, Volume, Total heat, species... where each species datum consists of "name" and "starting quantity." ADIABAT will append a message "UNCONVERGED" to results lines which exceeded the iteration limit, and will list two output results from such runs (which represent its most recent attempts to estimate the solution desired). It should be noted that ADIABAT finds starting conditions which lead to a "net heat = 0" condition; if the user would like to know the final products of such conditions, it would be necessary to rerun SOLGAS at the conditions determined by SOLGAS.

For the peace of mind of the user, ADIABAT will display a status line telling which runs it has completed and the final values of a few control variables. When done with all runs, it will type "Finished" and return to DOS.

As implied above, the results can be most easily assembled in a spreadsheet where they can be plotted as desired.

H. Temporary files

Two temporary files are used, which should nearly always be invisible to the user. These are "$$temp.mac" and $$temp.out." The first is a "macro" which contains the simulated keystrokes that SOLGAS needs to process each point. The second is the output of a single SOLGAS run. These files are repeatedly created and deleted in the default directory as the run progresses and will be erased when the run is finished.

I. A hint on speed

During its operation, ADIABAT continually loads and runs SOLGAS. After it makes the first run, it varies the control variable and makes another run. The results of these two variables are used to decide how to make the next run in order to begin to converge on the final result (nearly zero total heat). Because it does the extrapolation or interpolation within ADIABAT, it has to load SOLGAS for each run. Hence, a limiting factor is the speed with which SOLGAS can be loaded, the file examined, and the data file accessed.

The speed of the process can be greatly enhanced by placing all heavily accessed working files (COMMAND.COM, SOLGAS.EXE, MY_FILE.DAT.) on a high speed or RAM drive and making that the default drive. An increase of a factor of five has been obtained by using a RAM drive versus a removable hard disk. It seems plausible that a good-sized disk cache will speed up the process more, though we have not tested this.

J. Limitations

ADIABAT's mode of interpolation can be a weakness. Once it has bracketed the answer, ADIABAT uses a newtonian method of iterative interpolation to seek the answer. It is limited to 20 iterations. In some problems, a small change in starting composition, for example, can lead to drastic changes to equilibrium compositions. In these cases, ADIABAT can fail to converge. However, it prints both sets of current calculations along with a warning saying that the system is "unconverged." Sometimes the results, though unconverged, provide useful information.

K. Common errors in ADIABAT

The following errors have occurred often enough to the authors that they are worth mentioning for the potential user:

1. Translation error.

Immediately after the user responds with the name of the SOLGAS data file, a message indicating "translation error" appears, accompanying an image of the line in the SOLGAS data file containing the materials' starting temperature. This message often means that ADIABAT "sees" an extra line in the data file beyond the final data line. The solution is to use a text editor such as EDLIN or EDIT (in DOS), or any word processor which can import and export data in an ASCII mode. One then deletes the offending line. (Using <RTN> in a word processing program after the final bit of data has been sufficient in the authors' experience).

2. Missing DOS file.

A couple of versions of this have been observed:

i. DOS error #:2 running "del $$temp.out"
ii. DOS error #:2 running "SOLGAS.EXE < $$temp.mac >nul"

The reason for the errors is that SOLGAS cannot find COMMAND.COM. The solution is to copy COMMAND.COM to the default directory and try again.

3. Bad command.

DOS spells out "Bad command or filename," which indicates either SOLGAS.EXE or COMMAND.COM is not immediately available. The solution is to copy the missing file to the default directory and restart the program.

V. Semi-Automated plotting of data: using MASSAGE

A. Reason for program

Extracting data manually from a large number of SOLGAS output tables can be a laborious task, one which readily lends itself to performance by a computer. Automation of the extraction process is the reason for the existence of MASSAGE. It takes any amount of SOLGAS output (in a single file) and organizes the data into comma-delimited fields, each related to a single species' composition, temperature, heat, or volume.

The advantage of such organization is that the data can be easily imported into a spreadsheet where it can be further organized, pruned, and plotted with relative ease. Thus, the computer removes an additional time-consuming portion of data processing.

B. Starting MASSAGE

MASSAGE.EXE is a compiled data "post-processor" which is run on the IBM-compatible PC by typing MASSAGE, followed by <RTN>. The user is asked for the file name (the name of a file containing tabular output from SOLGAS). In the example discussed so far, the user responds "MY_FILE.OUT," for this is the multiple-run output file, quite likely generated by CHOICE 12.

The user is then asked for an output file, and again using these conventions, responds MY_FILE.PRN.

C. Output from MASSAGE

MASSAGE is really designed to be used with a spreadsheet, and the following comments must be viewed in that light. One can import MY_FILE.PRN into any of several spreadsheets or graphics packages, recognizing that the file is comma delimited.

MASSAGE's product, e.g., MY_FILE.PRN, will contain one header line, consisting of the following column labels:

"ID," "T(K),", "P (bar)," "V (l)," "Pre-Heat (kJ)," "net Heat (kJ," "Species ID," "N (initial)," "N (equilib)," and "P (bar)."

Under the column headers are the data with a row for each species in each run. Runs that failed to find a solution will be skipped. The "ID" label is an arbitrary number assigned to a run in the order the data are encountered in MY_FILE.OUT. The number itself has no significance, except that it can be useful in straightening out missorted output lines.

The ID, T (temperature), P (pressure), V (volume), and heat entries are common to all lines from a given run. After these data elements, the species name is followed by species-specific information: initial and final quantities (in moles); for gases, partial pressure in bar; for condensed mixtures, mole fraction.

APPENDICES

APPENDIX A - Genera' Comments On Appendices

Because the explanations of SOLGAS may not be as clear to the readers as to the authors, some examples have been included in this report. Lacking the ability to show the computer screen as we describe the interactive responses, we have chosen to list both the screen output of SOLGAS and the responses typed when asked for them. To distinguish between the output of SOLGAS and the responses typed in, we have put the responses in **BOLD CAPS.**

In addition, comments have been added on the right sides of the pages. To distinguish these comments from SOLGAS output or **USER RESPONSE,** we have put braces ({ }) around each comment.

A brief, descriptive heading is given for each example.

APPENDIX B - Elementary Examples

I. The relationships among water, ice, and water vapor.

Thermodynamic data and phase diagrams are two facets of the description of a material system. Ideally, the thermodynamic data should enable one to construct a phase diagram of a given system, and water serves as an example in this Appendix.

Water, saturated with air and at 1 atm pressure (0.101325 MPa exactly), freezes at 0° C or 273.15 K. By definition of the thermodynamic temperature scale, water, ice, and water vapor are in equilibrium at 273.16 K. Ideally, thermodynamic data associated with these three phases should predict the triple point at 273.16 K. They should also predict a boiling point at 373.15 K, as required by the definition of the Celsius scale.

This appendix has the advantage not only of showing how SOLGAS operates in a simple case, it also shows the relationship between thermodynamic data and familiar phase relations.

II. Input and output of SOLGAS using ice and water as an example.

A. Sources of data

Fig. B.1. Thermodynamic data of water species. The file name is WATER.DAT.

```
TITLE Demo File for melting ice and boiling water
T=  298.15
*G  Ar      Ar     0     154.845  20.786   0          0          0          0           ; JANAF 86
G   H2O(g)  H2O  -241826  188.834  26.315  1.717E-02  -2.330E-06  2.615E+05  -2.080E+07  ; JANAF 86
*P  H2O(l)  H2O  -285830   69.950 116.062 -2.100E-01  3.030E-04  -3.400E+05              ; JANAF 86
P   H2O(s)  H2O  -292800   44.586  37.050 ; Extrapolated from H(fusion) & Cp(ice) in CRC
```

Fig. B.1 gives the thermodynamic data file from which the calculations in this section and in Section B.III were calculated (See Section II.A.2 for a description of the meanings of the values and symbols). One sees that the gaseous data and the data on liquid water were taken from the JANAF tables, while the data on solid ice were taken from the Handbook of Chemistry and Physics[10] published by the CRC Press.

B. This section contains only output from use of the file WATER.DAT.

Fig. B.2. Output from the input file WATER.DAT

```
  [1] Computer selects a starting estimate
    :
    :                                          (menu items deleted for brevity)
    :
  [5] Recalc. with same data, pts. & quantities
    :
 Select an Option: 5                               (The problem is re-     )
                                                   (calculated with the new)
 T -    273.200 K                                  (value for temperature  )
 P -      1.000 bar
 V -  0.229D-04 liters

              Initial Qty    Final Qty      Pressure
               (moles)        (moles)         (bar)
 Ar           0.10000D-05    0.10000D-05    0.99387D+00
 H2O(g)       0.00000D+00    0.61707D-08    0.61329D-02

 H2O(l)       0.10000D+01    0.10000D+01
 H2O(s)       0.00000D+00    0.00000D+00

   Species number   4 (H2O(s)   ) would be considered
   in the equilibrium set of phases if its free energy
   of formation were          1.9 J/Mole more negative

 Pre-heat            -     -1.912 kJ
 Heat of reaction -         0.000 kJ
 Total heat          -     -1.912 kJ

Paused -- Push RETURN to continue          (user entered a carriage return)

  [1] Computer selects a starting estimate
    :
    :                                          (menu items deleted for brevity)
    :
 [10] Save the last calculated result
    :
 Select an Option: 10                              (save these results)

  [1] Computer selects a starting estimate
    :
    :                                          (menu items deleted for brevity)
    :
  [9] Exit program
    :

 Select an Option: 9                               (Exit the program...)

C:\>                                               (... and back in DOS)
```

Fig. B.2. (continued)

```
C:\> SOLGAS

        **** SOLGAS Version 2.44 ****

 Input File Name : WATER.DAT
 Output File Name: WATER.OUT

Demo File for melting ice and boiling water          (Title of data file)

 2 - No. of Condensed Phases of Fixed Composition     (information about the)
 1 - No. of Mixtures                                  (data file WATER.OUT   )
 2 - No. of Species in Mixture # 1
 3 - No. of Elements
 2 - No. of Linearly Independent Components

 Converting to Linearly Independent Components per    (SOLGAS has determined )
                                                      (that some species are )
Comp.\Elem.   Ar      H       O                       (linearly dependent; it)
Comp.# 1    1.000   0.000   0.000                     (alters formulae per   )
Comp.# 2    0.000   2.000   1.000                     (this table            )

 Initial Temperature of Starting Materials -  298.15 K

 Input the Reaction Temperature (K): 273.1

  [1] Computer selects a starting estimate
  [2] Manually select a starting estimate
  [3] Change No. of pts. to calculate & quantities of starting materials
  [4] Change quantities of starting material only
  [5] Recalc. with same data, pts. & quantities
  [6] Change the reaction temperature
  [7] Change value of fixed pressure or volume
  [8] Change to a new system
  [9] Exit program
 [10] Save the last calculated result
 [11] Save the last series of results
 [12] Read run data from a parameter file

 Select an Option: 1

 Input No. of points to be calculated (<100),
  allowing for variation in quantities of starting materials: 1

Raw material species in sequence are :
 Ar        H2O(l)
```

(continued on next page)

Fig. B.2. (continued)

```
 [1]  N varies irregularly                     | N is the quantity
 [2]  N is constant                            |   in moles
 [3]  N varies by a constant increment         |

Input option for Ar       : 1
Input option for H2O(l)   : 1
Option for Ar        is  1; Input  1 value(s) for  1 point(s): 1E-6
Option for H2O(l)    is  1; Input  1 value(s) for  1 point(s): 1

 T -    273.100 K                          (result is displayed on)
 P -      1.000 bar                        (the screen.)
 V -  0.228D-04 liters

             Initial Qty    Final Qty     Pressure
               (moles)       (moles)        (bar)
 Ar          0.10000D-05    0.10000D-05   0.99391D+00
 H2O(g)      0.00000D+00    0.61251D-08   0.60878D-02

 H2O(l)      0.10000D+01    0.00000D+00
 H2O(s)      0.00000D+00    0.10000D+01

    Species number   3 (H2O(l)  ) would be considered
    in the equilibrium set of phases if its free energy
    of formation were          0.2 J/Mole more negative

 Pre-heat          -     -1.920 kJ
 Heat of reaction  -     -5.978 kJ
 Total heat        -     -7.898 kJ

Paused -- Push RETURN to continue         (user entered carriage return)

  [1] Computer selects a starting estimate
   :
   :                                       (menu items deleted for brevity)
   :
 [10] Save the last calculated result
   :
 Select an Option: 10                            (the results are saved)
                                                 (to the file WATER.OUT)
  [1] Computer selects a starting estimate
   :
   :                                       (menu items deleted for brevity)
   :
  [6] Change the reaction temperature
   :
 Select an Option: 6

 Input the Reaction Temperature (K): 273.2

          (continued on next page)
```

C. General comments about calculated results from file

Output from this calculation, as given in Section B, is worth examining. After responding to questions from SOLGAS, the user gets some summary information about the input file on the screen. Perhaps the most interesting is the determination by SOLGAS that, despite the fact that there are three elements in the system being examined, there are only two components. Component #1 is made of pure Ar, and Component #2 is made up of two gr.-atoms of H and one gr.-atoms of O. This illustrates that SOLGAS is basing its calculation of thermodynamic equilibrium on components in the system, and not on particular chemical substances. Frequently, the two are identical.

Following the screen output in Section B farther, one notes that at 273.10 K, SOLGAS calculates that condensed H_2O is solid, while at 273.20 K, condensed water is liquid. This calculation obviously agrees with our definition of the ice point lying at 273.15 K. One should note that SOLGAS would be unable to calculate the exact freezing point, but can only approach it almost as closely as desired. Note that SOLGAS in its calculation should approach the triple point in this case: 273.16 K. It would not "know" that the phases were saturated with air -- to get the ice point -- unless additional information were provided.

The gaseous pressure of water is given by SOLGAS as 0.006136 bar at 273.2, which is 0.05° C. One interpolates the pressure of water over liquid water at 273.2 from the CRC handbook as 4.596 mm Hg, or .0061275 bar, less than 0.1% difference. One sees that the heat of fusion is within 7 cal per mole (30 joules per mole) of the published value in the CRC handbook.

The agreement is exceedingly good, and it was accomplished without having to adjust any of the basic data. While the heat of formation of solid water at 298.15 is probably not possible to measure, one can calculate it readily. The heat of formation and entropy of liquid water was taken from the JANAF tables, as noted in the library section of these Appendixes. The heat capacity of liquid water in the vicinity of room temperature is tabulated in these tables and was fitted to an equation. The heat capacity data of both Dickinson and Osborne[11] and Giauque and Stout[12] for ice were fitted earlier to a linear equation. The heat of fusion of water is a well known value, and was taken from the CRC handbook, 6008 J/mol.

A useful calculation made by SOLGAS is sandwiched between the report of the amounts of the several species (initially and at equilibrium) and the report of the various calculated heats. We refer to the section beginning "Species number ...."; SOLGAS here informs the user that an unstable species was only unstable by so many joules. The user could easily use this information to adjust heats or entropies of a species to force stability, if desired.

III. Prediction of boiling point of water

A. Use of "parameter file" to calculate the boiling point of water

As mentioned previously, the thermodynamic data used for the calculation were those shown in Fig. B.1. In order to make the calculation, a parameter file, WATER.CTL, was generated; the contents of this file were shown in Fig. 2 of the main section (III.G) of this report. CHOICE 12 was utilized, and the parameter control file, WATER.CTL, varied the temperature from 373.000 to 374.000.

B. Data obtained by use of the parameter file WATER.CTL, shown in Fig. B.3, in CHOICE 12 of SOLGAS.

C. Comments on agreement of calculated boiling point with definition of water's boiling point.

The data of Section B. show that a gaseous pressure of 1.013 bar is not reached until the temperature is 373.8 K, more than 0.6 degrees above the defined boiling point. This fact may well be surprising to the reader, for it is inconsistent with known phase relations. The Celsius temperature scale is defined based on the 100 degree difference between the freezing point of water and its boiling point, equilibrium with vapor at a vapor pressure of 1 atm. or 1.013 bar. Keep in mind that the data for water in the gaseous state refer to the ideal gas, whereas the boiling point of water is defined based on real gas.

One could check to see that the heat of vaporization calculated via SOLGAS, and that interpolated from the JANAF tables, are in close agreement. This should be no surprise since the data came from the JANAF tables.

Fig. B3. Data from SOLGAS as obtained by using the file WATER.CTL after choosing CHOICE 12.

```
Demo File for melting ice and boiling water
   :                                        (initial info deleted for brevity )
T -    373.000 K                            ( table deleted for brevity)
   :
T -    373.200 K                            ( table deleted for brevity)
   :
T -    373.400 K                            ( table deleted for brevity)
   :
T -    373.600 K
P -      1.013 bar
V -  0.962D-02 liters

             Initial Qty     Final Qty      Pressure
               (moles)        (moles)         (bar)
Ar           0.10000D-05    0.10000D-05    0.32301D-02
H2O(g)       0.00000D+00    0.31261D-03    0.10098D+01

H2O(l)       0.10000D+01    0.99969D+00
H2O(s)       0.00000D+00    0.00000D+00

   Species number   4 (H2O(s)   ) would be considered
   in the equilibrium set of phases if its free energy
   of formation were       2855.6 J/Mole more negative

Pre-heat          -       5.795 kJ
Heat of reaction -       0.013 kJ
Total heat        -       5.807 kJ

T -    373.800 K
P -      1.013 bar
V -     30.681 liters

             Initial Qty     Final Qty      Pressure
               (moles)        (moles)         (bar)
Ar           0.10000D-05    0.10000D-05    0.10130D-05
H2O(g)       0.00000D+00    0.10000D+01    0.10130D+01

H2O(l)       0.10000D+01    0.00000D+00
H2O(s)       0.00000D+00    0.00000D+00

   Species number   3 (H2O(l)   ) would be considered
   in the equilibrium set of phases if its free energy
   of formation were         11.9 J/Mole more negative

Pre-heat          -       5.810 kJ
Heat of reaction -      40.736 kJ
Total heat        -      46.546 kJ

T -    374.000 K                            ( table deleted for brevity)
```

APPENDIX C - Calculation Of a Simple Phase Diagram: The Uranium-Fluorine System Between UF_4 And UF_6.

There are seven known solid fluorides of uranium: UF_3, UF_4, U_4F_{17}, U_2F_9, UF_5 (which exists in two known forms), and UF_6. The hexafluoride is the best known; it is used as the gaseous medium for enriching uranium in the fissionable isotope U-235. The composition region between UF_4 and UF_6 is fairly well known, and some of the thermodynamic data are available to describe the system. Perhaps the easiest way to describe what is known is to refer the reader to Fig. C.1, where most of the available thermodynamic data are tabulated. These are tabulated in the form useable by SOLGAS, and in this case, they come directly from the library to be described in Appendix F. Incidentally, the data of Fig. C.1 will also be used in Appendix D, where we show how to calculate temperatures reached when UF_4 is reacted in F_2.

In the present example, a control file (for menu CHOICE 12) was constructed in which many runs varying only in U to F ratio (ranging from UF_4 to UF_6) were specified. All runs were done at a temperature of 350° K and a constant pressure of 10 bar. The results provide the data to construct a phase diagram.

Fig. C.2 shows a plot of the partial pressures of the gaseous species (only). One can see that there are four regions, distinguished by the four different equilibrium partial pressures of $UF_6(g)$. Fig. C.3 shows the equilibrium quantities (in moles per mole of U in the initial mixture). In the regions of the diagram depicted by the horizontal lines, two condensed phase uranium fluorides are present which determine the equilibrium vapor pressure of UF_6. Note that much the same information as is obtained in Fig. C.2 could be obtained in four runs of SOLGAS, one run in each of the condensed-phase regions.

Fig. C.1. Thermodynamic data for uranium fluorides

```
TITLE UFx Equilibria
T=        298.15
*G  Ar        Ar           0 154.845  20.786          0         0         0         0 ; JANAF 86
 G  F         F        79390 158.750  23.238 -2.793E-03 8.417E-07 2.055E+04 -2.42E+06 ; JANAF 86
*G  F2        F2           0 202.789  31.902  7.320E-03 -1.96E-06 -2.91E+05 2.563E+07 ; JANAF 86
 G  U2F10(g)  U2F10 -3998845 577.496 257.236  3.376E-02 -1.11E-05 -3.44E+06 2.845E+08 ; JML 83
 G  UF4(g)    UF4   -1598700 368.000  91.200                                          ; NBS 82
 G  UF5(g)    UF5   -1917118 375.891 120.944  1.602E-02 -5.27E-06 -1.60E+06 1.327E+08 ; JML 83
 G  UF6(g)    UF6   -2148236 377.446 142.621  2.027E-02 -6.65E-06 -2.01E+06 1.661E+08 ; JML 83
 P  U2F9(s)   UF4.5 -1996929 170.678 137.385  1.755E-03 4.037E-06 -1.75E+06 1.492E+08 ; JML 83
 P  U4F17(s)  UF4.25 -1953698 161.201 132.847  1.738E-03 3.881E-06 -1.69E+06 1.442E+08 ; JML 83
 P  UF3(s)    UF3   -1502100 123.430  95.100                                          ; NBS 82
*P  UF4(s)    UF4   -1910218 151.640 127.024  4.599E-03 2.019E-06 -1.55E+06 1.307E+08 ; JML 83
 P  UF5(a)    UF5   -2074245 189.717 147.536 -9.358E-04 6.178E-06 -1.92E+06 1.644E+08 ; JML 83
 P  UF6(l)    UF6   -2179456 283.020 147.048  0.1296978           ; derived from data in JML 83
*P  UF6(s)    UF6   -2198118 227.793  50.631  3.893E-01                                   ; JML 83
```

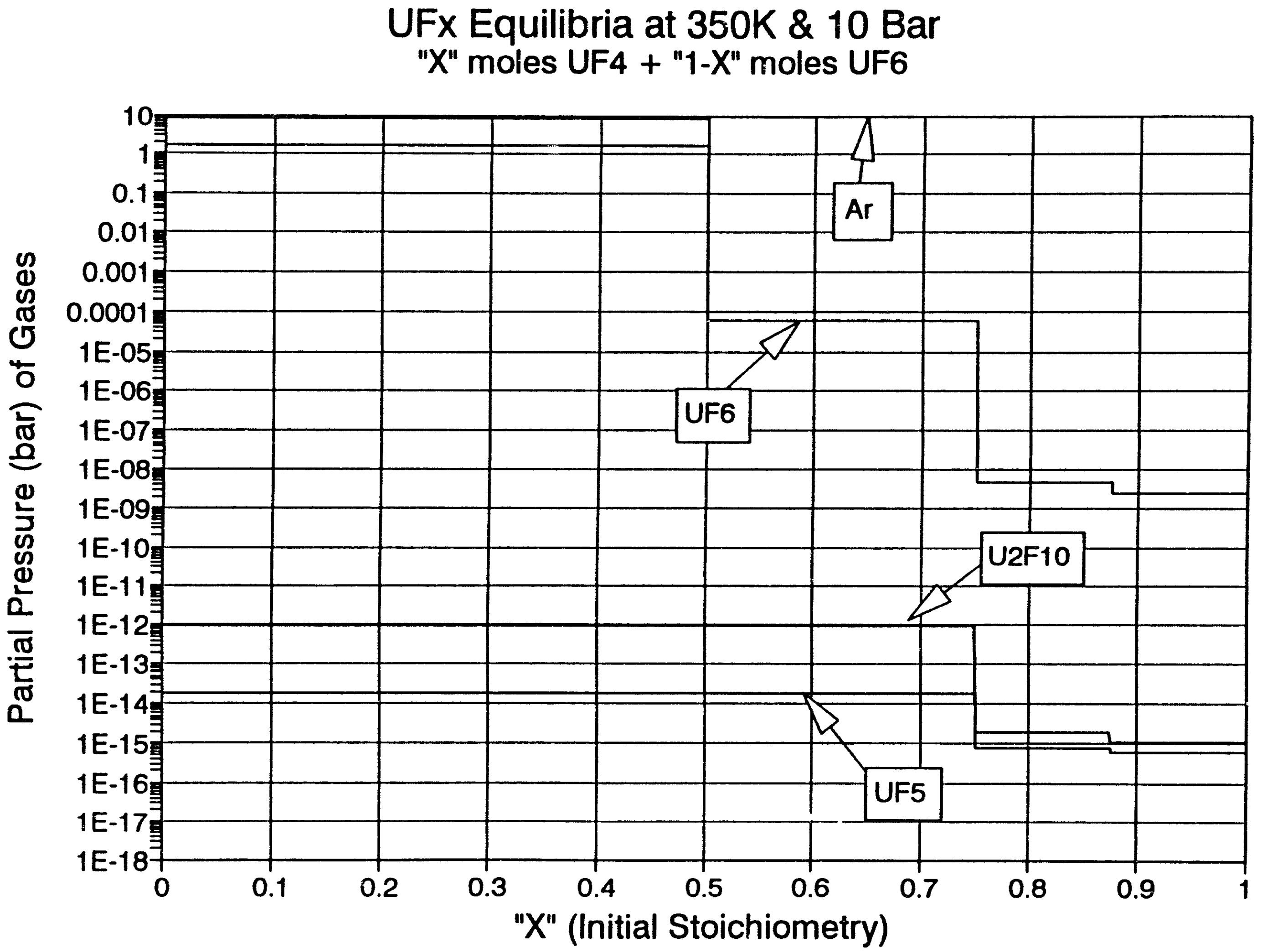

Fig. C.2. A partial phase diagram of the U-F system, calculated from the thermodynamic data of Fig. C.1.

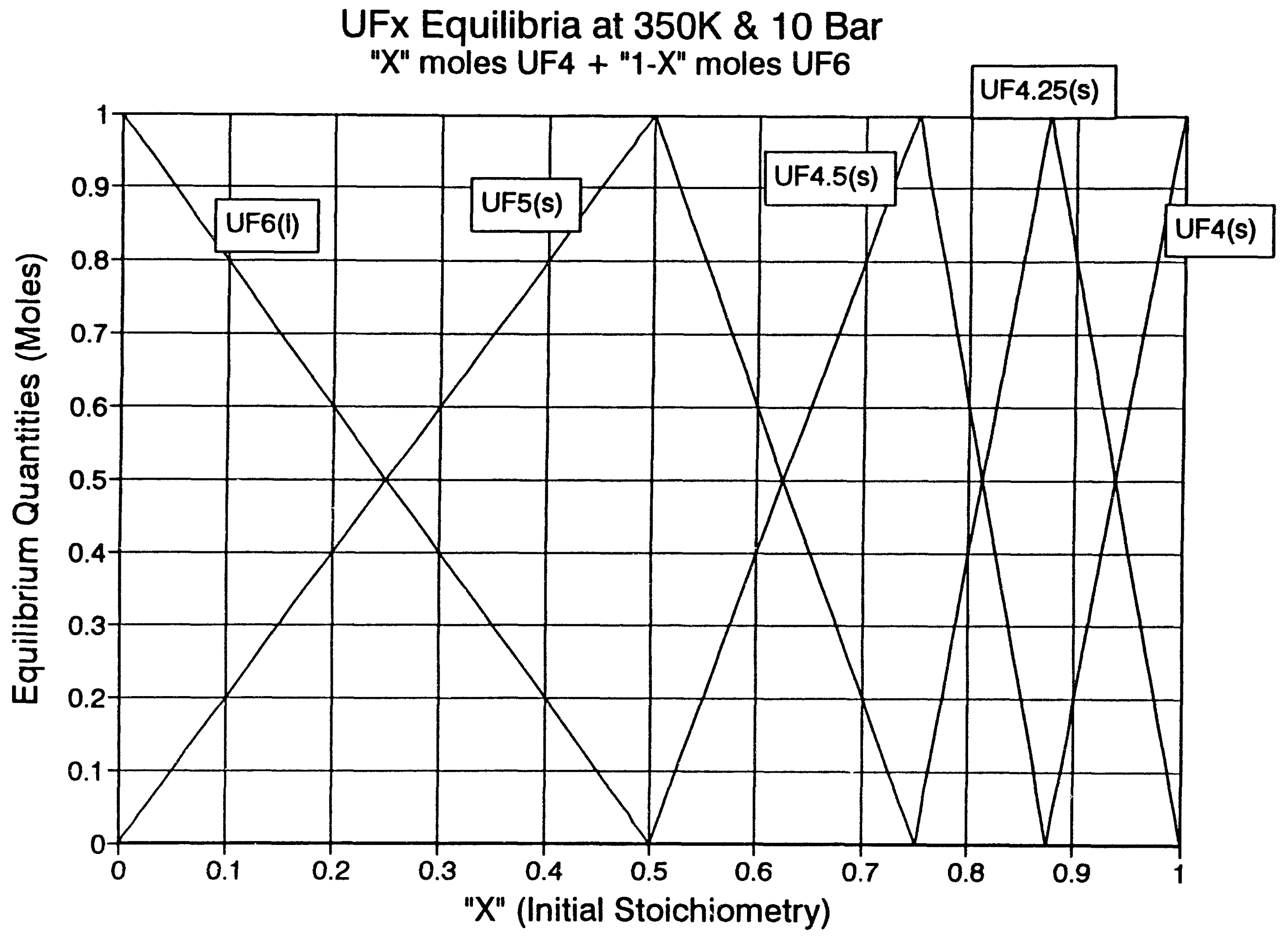

Fig C.3. Equilibrium quantities of uranium fluorides between UF_4 and UF_6.

APPENDIX D - Burning UF_4 In Fluorine. Use Of ADIABAT.

SOLGAS customarily calculates equilibrium under either constant temperature and pressure conditions, or under constant temperature and volume conditions. However, from the standpoint of safety or perhaps fuel economy, the maximum temperature rise a reaction can achieve can also be of interest. ADIABAT is a program designed to use SOLGAS iteratively to generate this information.

Fig. C.1, UF4_BURN.DAT, is the thermodynamic data file created to investigate the temperature rise expected when UF_4 reacts with F_2 (data for Ni(s) were added to the file shown in that figure in order to make the calculations shown here). The "problem" is as follows:

A one-liter container, made of nickel and weighing 500 g, holds 15 g of UF_4. Fluorine gas, at a pressure of 0.5 bar, is introduced into the system. The system begins at 25° C (298.15 K). At a constant pressure and neglecting the effect of the container, what is the adiabatic change in system temperature and gas volume? If one included the container's ability to absorb heat, what is the adiabatic change in system temperature and gas volume? Further, under either of the above conditions, what is the effect on pressure of gases if the volume of the system is held constant? Thus, there are four conditions one might reasonably examine in connection with this problem.

The first step is to create a control file through which, to operate ADIABAT, one might give each of the four starting conditions. Such a file, UF4_BURN.DRV, is shown in Fig. D.1. The first column of Fig. D.1 (all entries of which are "zeroes") indicates that SOLGAS is to investigate the temperature variation in each of the four cases. Column two presents, for each case, a "guess" temperature as a starting point for ADIABAT. Column three indicates either a constant pressure (0.5 bar, in this case) or a constant volume (1.0 liters -- the negative sign telling SOLGAS the value is a volume, not a pressure). Column four gives the moles of argon gas input to ensure an equilibrium pressure could be achieved. Column 5 contains the moles of F_2 which is calculated from the ideal gas law at the starting temperature, pressure, and volume. Column 6 specifies the moles of UF_4, calculated from the 15 g known to be input. Column 7 includes (or omits) the mass of the nickel, again in moles rather than grams.

Fig. D.1. Image of the control file UF4_BURN.DRV

Ctrl	T	P or V	Ar	F_2	UF_4	Ni
0	800	0.500	1E-6	0.0202	0.0478	0
0	300	0.500	1E-6	0.0202	0.0478	8.52
0	300	-1.00	1E-6	0.0202	0.0478	0
0	300	-1.00	1E-6	0.0202	0.0478	8.52

In order to perform the desired calculation, one must assure that ADIABAT.EXE, SOLGAS.EXE, and COMMAND.COM (from the user's DOS) are all present in the default directory. To initiate the calculation, one types ADIABAT <RTN>. The user is asked, in order, for a name of a SOLGAS thermodynamics data file (UF4_BURN.DAT), a control file name (UF4_BURN.DRV), and an output file name (UF4_BURN.PRN). The names in parentheses follow, through their extensions, the convention described here. There is no requirement to use this. A record of the screen display during the run is shown in Fig. D.2.

Fig D.2. Output of the screen during SOLGAS under control file UF4_BURN.DRV.

```
D:\> ADIABAT

Adiabatic Driver for SOLGAS v. 2.44 -- written in Turbo Pascal v 6.0
    version 7.0                        by L.D.Trowbridge --  5/18/93

SOLGAS thermodynamics data file name? UF4_BURN.DAT
       Control file name?             UF4_BURN.DRV
       Results file name?             UF4_BURN.PRN
Run #    1 T =  942.30 H = -0.003, X = ( 0.000, 0.020, 0.048, 0.000)
Run #    2 T =  327.56 H =  0.000, X = ( 0.000, 0.020, 0.048, 8.520)
Run #    3 T =  964.22 H = -0.002, X = ( 0.000, 0.020, 0.048, 0.000)
Run #    4 T =  327.56 H =  0.000, X = ( 0.000, 0.020, 0.048, 8.520)
Finished.

D:\>
```

In Fig. D.2, the run number is given first, followed by the equilibrium temperature. ADIABAT operates, as described previously, by searching for the temperature at which the algebraic sum of the "Pre-heat" and the "Heat of reaction" is zero, within 0.02 J. The deviation, to three significant figures, is given in the next column. The last columns give the amounts of the input materials to three significant figures.

The results are sent to an output file, under the name UF4_BURN.PRN, as called for by the user initially. An image of this file is shown in Fig. D.3. One can conveniently import the file into a spreadsheet, and an image of such a file after importation is shown in Fig. D.4.

Both output schemes yield the same kinds of information, of course. At a constant pressure of 0.5 bar, the temperature could rise as high as 942 K, if no account is taken of heat loss to the container. However, if the container mass is included in the calculation, the rise in temperature is only to 328 K. (Incidentally, one must look to Fig. C.2 to learn that the starting temperature of the reaction was 298.15 K.)

If one maintains a constant volume, the temperature rise can be even higher when the mass of the container is ignored: 964 K. Including the container, however, yields essentially the same temperature rise as the constant pressure process. Of course, in a real situation, the instantaneous temperature rise will vary from point to point in the system, depending on the instantaneous availability of reactants, heat transfer, and interaction with container walls. However, the calculations establishes limits, either to approach or avoid, as the demands of the situation dictate.

ADIABAT output does not tell the user what species are present at equilibrium. To find that, one would have to use the above data to specify starting conditions and rerun SOLGAS. If one were to do this in the above case, one would see that quite different products were produced at low temperatures relative to those produced at high temperatures. At low temperatures, the reaction of UF_4 with the amount of F_2 specified produces a mixture of solid U_2F_9 and solid UF_5. At high temperatures, the solid intermediate fluorides disproportionate to solid UF_4 and gaseous uranium fluorides, most of which is UF_6, although some gaseous UF_5 and its dimer, U_2F_{10}, are also present.

Fig. D.3. Output from ADIABAT, UF4_BURN.PRN, obtained via SOLGAS.

```
"ID","T(K)","P (bar)","V (l)","Net Heat (kJ)", "Name","Qi(1)","Name", "Qi(2)",
    "Name","Qi(3)","Name","Qi(4)"
1, 942.30, 5.000E-0001, 3.590E+0000,-3.0000E-0003,"Ar      ", 1.0000E-0006,
    "F2      ", 2.0200E-0002,"UF4(s)  ", 4.7800E-0002,"Ni(s)   ", 1.0000E-0006
2, 327.56, 5.000E-0001, 5.450E-0005, 0.0000E+0000,"Ar      ", 1.0000E-0006,
    "F2      ", 2.0200E-0002,"UF4(s)  ", 4.7800E-0002,"Ni(s)   ", 8.5200E+0000
3, 964.22, 1.776E+0000, 1.000E+0000,-2.0000E-0003,"Ar      ", 1.0000E-0006,
    "F2      ", 2.0200E-0002,"UF4(s)  ", 4.7800E-0002,"Ni(s)   ", 1.0000E-0006
4, 327.56, 3.620E-0005, 1.000E+0000, 0.0000E+0000,"Ar      ", 1.0000E-0006,
    "F2      ", 2.0200E-0002,"UF4(s)  ", 4.7800E-0002,"Ni(s)   ", 8.5200E+0000
```

[Note: Results for each run actually occupy a single line.]

Fig. D.4. Image of UF4_BURN.PRN after importing into Quattro Pro.

ID	T(K)	P(bar)	V (l)	Net Heat	Name	Qi(1)	Name	Qi(2)	Name	Qi(3)	Name	Qi(4)
1	942.30	0.5	3.59	-0.003	Ar	0.0000	F2	0.0202	UF4(s)	0.0478	Ni(s)	0
2	327.56	0.5	0.00005	0	Ar	0.0000	F2	0.0202	UF4(s)	0.0478	Ni(s)	8.52
3	964.22	1.776	1	-0.002	Ar	0.0000	F2	0.0202	UF4(s)	0.0478	Ni(s)	0
4	327.56	0.00003	1	0	Ar	0.0000	F2	0.0202	UF4(s)	0.0478	Ni(s)	8.52

APPENDIX E - Calculations In The Cl-F-O-H System

Experimentally, when ClF_3 is reacted with moisture, a mixture of chlorine oxides or chlorine oxyfluorides is observed. The observation is at variance with thermodynamic calculations, even though the thermodynamic data are believed to be reliable within the accuracy needed by the calculation. To illustrate this behavior, two series of runs were done.

The first run (the straightforward approach) explored the equilibrium of the reaction between water and ClF_3 in a mixture with 50% of an inert gas, Argon. The constant pressure was 0.5 bar, the temperature 350 K. Species permitted in the example were:

The gases Ar, H_2O, HCl, HF, Cl_2, Cl_2O, ClF, ClF_3, ClO, ClO_2, $ClOF_3$, FClO, ClO_2F, ClO_3F, F_2, O_2; the liquid phases H_2O and HF; and the solid phase H_2O.

Thermodynamic data were taken from the LIBRARY.WK1 data base. Starting material consisted of 50 mol % Ar, 50 mol % $\times$ X_i ClF_3 and 50 mol% $\times$ $(1-X_i)$ gaseous H_2O (where X_i varies from zero to one). A control parameter file was created which defined approximately 100 different compositions, and SOLGAS was used under control of CHOICE 12, which was described in Section III.G.

Results of the extensive calculations were processed by means of the program MASSAGE, and these further results were imported into the spreadsheet program Quattro Pro. After the data were suitably sorted, they were plotted as shown in Fig. E.1.

Fig. E.1 shows nice linear relationships between the reactants and the products. Clearly, at equilibrium, gaseous water will convert ClF_3 into a mixture of HF, O_2, and Cl_2 if only enough water is present. However, in practice, this is not what is observed.

Fig. E.2 shows the calculated equilibrium if O_2 is not present in the list of species to be examined by SOLGAS. Qualitatively, the calculational results are in agreement with experiment. One sees that the oxides and oxyfluorides of chlorine are part of the equilibrium mix as long as elemental oxygen is not a possibility. One concludes, of course, that there is a kinetic barrier to formation of elemental oxygen in the reaction studied.

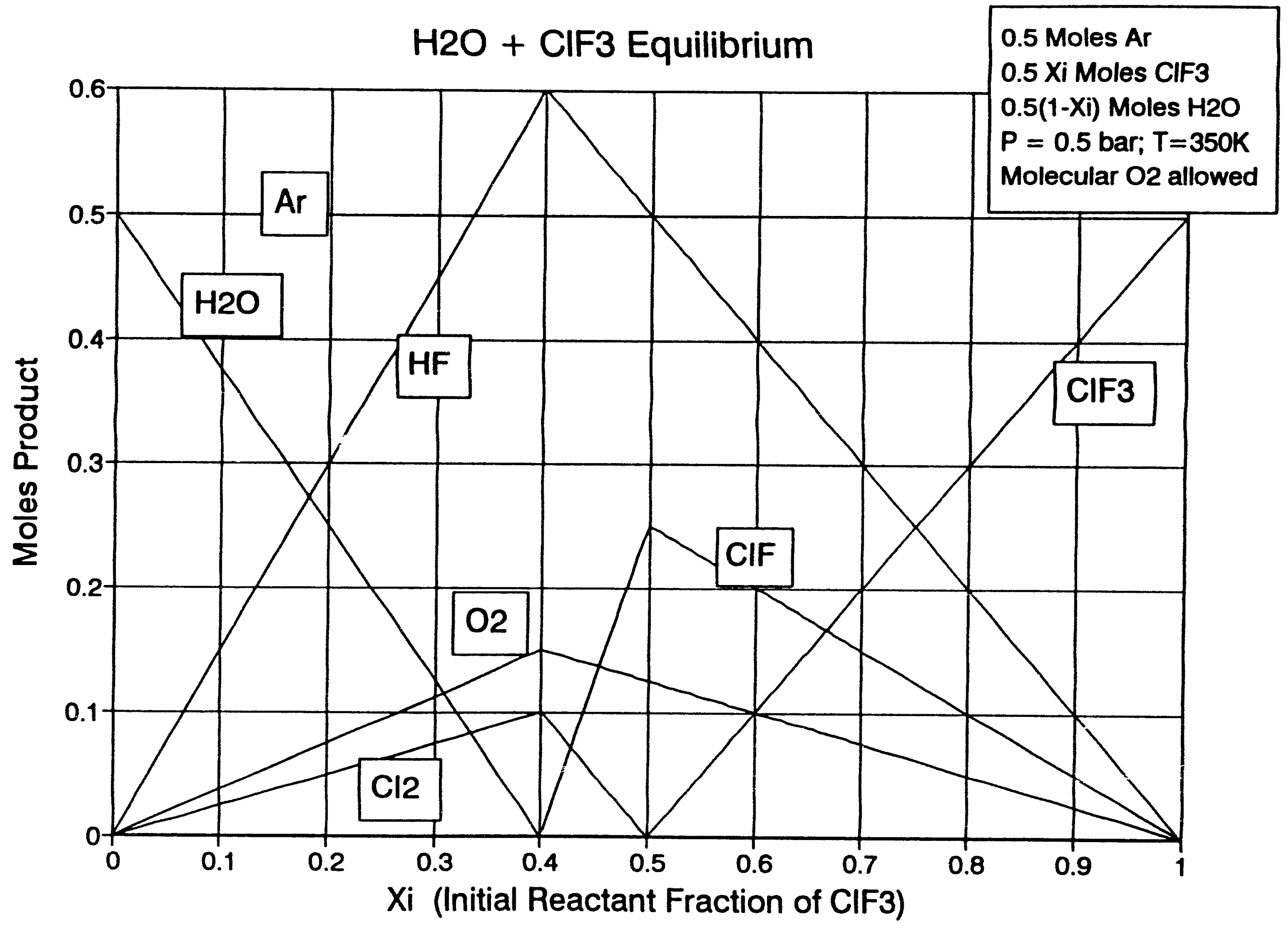

Fig. E.1. Plot of output from SOLGAS via Choice 12 for the *F-Cl-O-H* system when O_2 is allowed.

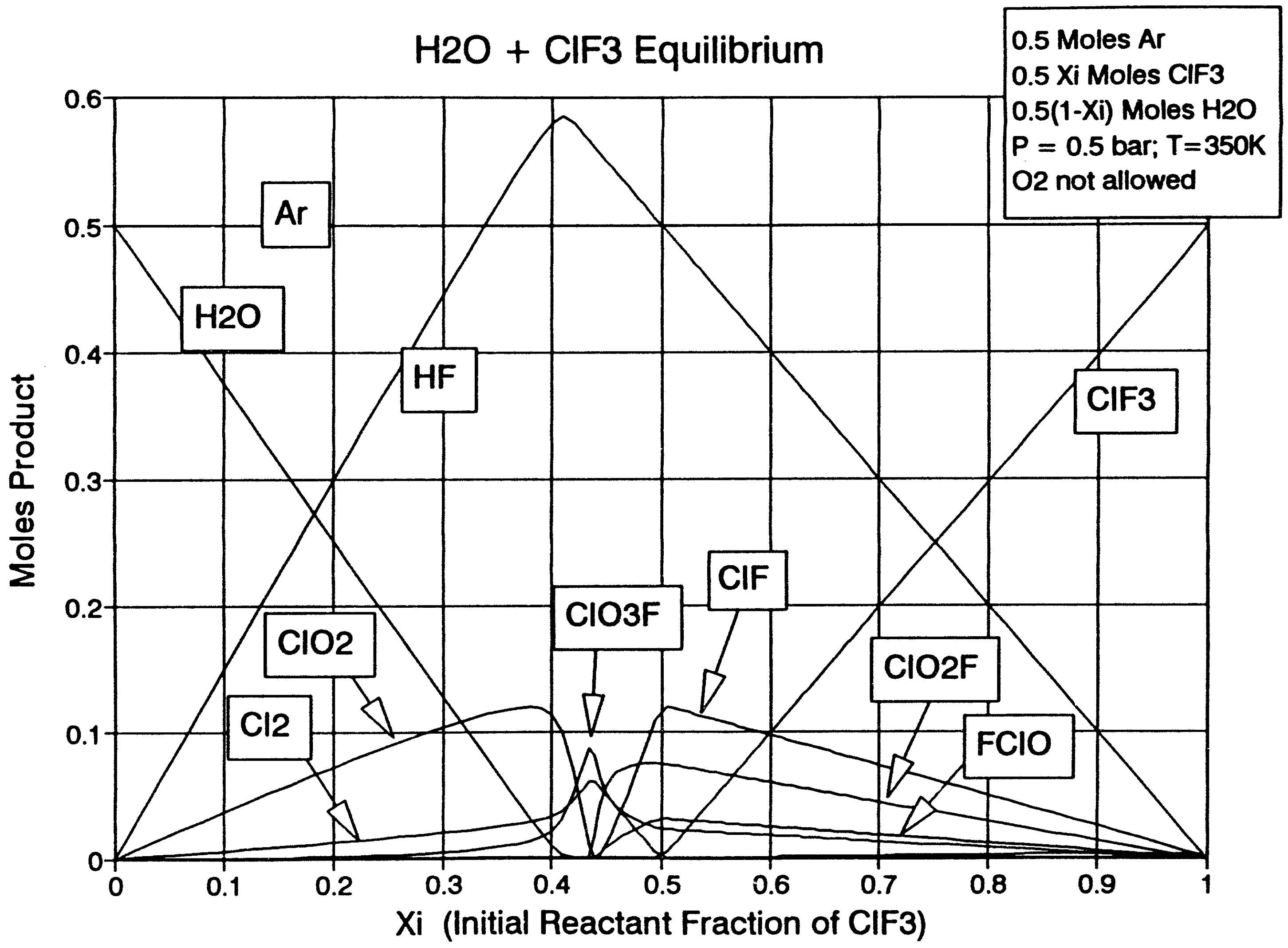

Fig. E.2. Plot of output from SOLGAS via Choice 12 for the F-Cl-O-H system when O_2 is not allowed.

APPENDIX F - Use Of a Spreadsheet As a Database For SOLGAS Input

I. Advantage of a spreadsheet database

Many, perhaps most, users of thermodynamic data work within a rather circumscribed region of data. Data on a few hundred compounds can be readily handled in a spreadsheet, a type of computer application that a great many people are becoming familiar with.

One rather obvious advantage is that once the data are firmly placed in the file (checked and rechecked), the possibility of mechanical (human) errors in working with the data is greatly reduced.

A second advantage of a database, such as is to be described, is the ease of use. Most users want to make a few calculations, aimed toward a specific answer within a project. They do not want to "make a career" of calculating thermodynamic quantities. A simple database that can be used by the occasional user can be highly desirable.

SOLGAS will satisfactorily operate on input files created in Lotus 123, Quattro Pro, or EXCEL[13]. We assume other spreadsheet packages could be used as well. One must be a little careful in preparing a file. The next section describes a file that can be imported directly into any of the three software products mentioned here.

II. Cautions In The Use Of Other Spreadsheets.

The library described in the next section can be imported into other spreadsheets. But the user may have to make some minor adjustments.

This version of SOLGAS will accept data files (e.g., MY_FILE.DAT) with variables separated either by spaces or by commas. Sometimes one may think the delimiter is a space (ASCII number 32 in decimal or 20 in hexadecimal notation) when it really is a tab (ASCII and hexadecimal number 9). We mention these things to emphasize that the data files must be space be or comma delimited.

Each user who uses the Lotus 1-2-3 library for importation into another spreadsheet may have to make minor adjustments in order to produce a file useable by SOLGAS.

III. A Lotus 1-2-3 version of LIBRARY

Two somewhat different spreadsheet versions of a library have been constructed by the authors, one for Lotus 123 and one for Quattro Pro. The Lotus version will be used as an example. Either library is available from the authors. The Lotus 123 library has been satisfactorily imported into EXCEL, although the translation routine has not been used to change the Lotus macros into EXCEL macros.

Fig. F.1 shows a part of the Lotus 123 library. The first row in Fig. F.1 actually starts at row 9 in the library, although the first column is column A in the library.

Fig. F.2, taken from an "opening screen" in the spreadsheet, describes the function of macros used to assist in creation of SOLGAS data sets from the library. To start a new data file, the user types <ALT>+N, which eliminates any "x's" or "*'s" in either Column A or B which might have been left over from a previous session. Next, the user goes through the library (which is much more extensive than is shown in Fig. F.2) and selects both the materials to be examined in the equilibrium mix (by entering an "x" in Column A of the desired species) and the materials to be used as input species (by means of an "*" in Column B.) This operation is not easily automated, and usually requires judgement on the part of the user.

The user then uses the second macro: <ALT>+C. This macro creates the data set from the selections. The things the macro does are: (1) asks the user to enter a starting temperature; (2) asks the user to enter a data file name (e.g., MY_FILE.DAT); (3) asks the user to enter a file title or description (this information can be up to 500 characters long, but SOLGAS is going to ignore all but the first 80 characters after "TITLE="). Following these interactive steps, the macro automatically, (4) extracts a file of compounds, based on the "x's", and finally (5) creates and saves a file which SOLGAS can use, with the file name the user told it in step (2).

Fig. F.1. A portion of the library in Lotus 1-2-3.

```
Sample Thermodynamics Data Library for SOLGAS v 2.44 - Spreadsheet-

Designate mat'ls to be considered with "x"in coln A, starting
mat'ls with "*" in column B, and mixture (solution) numbers in
column D.  When finished, push {alt}C to create SOLGAS data file.

Incl Start             Data at    298.150 K
      Matl                                        const          x T
 ?   SPMixCompound     Formula        Hf        S  Cp - A      Cp - B
                                   J/mol J/mol K J/mol K

      G    He          He              0 126.152  20.786           0
x    *G    Ar          Ar              0 154.845  20.786           0
      G    N2          N2              0 191.609  24.524   1.020E-02

x    *G    H2          H2              0 130.680  29.688  -8.773E-04
x    *G    O2          O2              0 205.147  26.112   1.144E-02
      G    O           O          249173 161.058  21.911
x     G    H2O(g)      H2O       -241826 188.834  26.315   1.717E-02
x     P    H2O(l)      H2O       -285830  69.950 116.062  -2.100E-01
x     P    H2O(s)      H2O       -292800  44.586  37.050  ; Calc. fr
```

Fig. F.2. Opening screen of LIBRARY.WK1.

```
Sample Spreadsheet Data Library for SOLGAS v 2.44

{alt}N: use to create a NEW data file for SOLGAS.
        In column "A", designate which species are to be  included
             in the data file with "x".
        In column "B", designate starting materials with "*".
        In column "D", enter the number of the mixture for each species
             designated as a type "M". Enter as left justified text.

{alt}C:  use to CREATE the data set once species are selected.

{alt}S:  macro to simply print out FILE IMAGE (perhaps after user
             makes cosmetic changes)

{alt}X:  clean up working areas for smaller storage.

A limitation of this system: SOLGAS allows up to 6 Cp equations per
species.  This library format allows for only one (the most common
case in our usage). Additional equations can be incorporated using
one of several methods (see LIBRARY.DOC).

To update species list, insert a line in the appropriate place in
the library above.  Add Species info.  Make sure data ends in a
semicolon (following which you may enter remarks).  To delete a
species simply delete the line from the main library list.
```

Two more macros can be useful to the user. One, using <ALT> + S, prints out the file image, perhaps after the user makes some cosmetic changes. A fourth macro, using <ALT> + X, acts to clean up working areas for smaller storage space.

Note that the work of the macros could be done manually by the user. For example, we imported the library into EXCEL, and were able to use virtually all the features of the Lotus 1-2-3 library unchanged except the macros. (See Section B, above, for a caveat on using an imported file.)

A limitation of this system is that SOLGAS allows up to 6 Cp equations per species. This library format allows for only one, the most common case in our usage. This limitation could be circumvented by including extra lines (rows) in the library which described the heat of transition, temperature of transition, and heat capacity constants, and then manually concatenating these in the test version of the data file. Alternatively, one could use multiple phases of the same species, as described previously in Section II.A.2.

Adding species to the library consists of (1) inserting a row by use of the appropriate spreadsheet commands into the main "library" area; (2) entering data in the appropriate cells; and (3) perhaps reformatting numeric formats in certain cells as desired or needed. Deleting a species from the library consists of deleting that species' row.

While the authors have not done so, transfer from data base packages are described by several spreadsheet manuals; e.g., QuattroPro, Excel, Lotus 123, or Enable. Those who have access to large data bases will most likely already have access to other, perhaps more powerful, calculation schemes.

REFERENCES

1. G. Eriksson, Acta Chem. Scand., 25(7), 2651 (1971).

2. G. Eriksson and E. Rosen, Chemica Scripta., 4, 193 (1973).

3. G. Eriksson, Chemica Scripta., 8, 100 (1975).

4. T. M. Besmann, "SOLGASMIX-PV, A Computer program to calculate equilibrium relationships in complex chemical systems," ORNL/TM-5775, Oak Ridge National Laboratory, Oak Ridge, TN, April, 1977.

5. K. E. Spear, private communication, 1986.

6. E. R. Cohen and B. N. Taylor, Physics Today, BG9-13, August 1988.

7. D. D. Wagman, et al, "The NBS tables of chemical thermodynamic properties," J. Phys. Chem. Ref. Data, 11, (2) 1982.

8. M. W. Chase, et al, "JANAF Thermochemical Tables, Third Edition," J. Phys. Chem. Ref. Data, 14, (1) 1985.

9. H. L. Schick, Thermodynamics of Certain Refractory Compounds," Vol. II, Academic Press, New York, 1966.

10. CRC Handbook of Chemistry and Physics, CRC Press, Inc., West Palm Beach, Fla., 1979.

11. H. C. Dickinson and N. S. Osborne, "The Specific Heat and Heat of Fusion of Ice," Bull. U. S. Bur. Standards 12, 49, 1915.

12. W. F. Giaque and J. W. Stout, "The Entropy of Water and the Third Law of Thermodynamics. The Heat Capacity of Ice from 15 to 273 K," J. Am. Chem. Soc. 58, 1144-50, 1936.

13. Lotus 1-2-3 is a registered trademark of Lotus Development Corporation; Quattro Pro is a registered trademark of Borland International; Excel is a registered trademark of Microsoft, Inc.

ATTACHMENT I

PROGRAM LISTING OF SOLGAS.FOR

```
      PROGRAM                   SOLGAS
C                        *** version 2.44 ***
C             Calculation of Thermodynamic Phase Equilibria
C                  at Const Pressure or Volume.
C
C "Most recent fingerprints"   Dr. J.M.Leitnaker & Dr. L.D.Trowbridge
C     are those of             Martin Marietta Energy Systems
C                              P.O. Box 2003, M.S. 7266
C                              Oak Ridge, TN 37831
C                              ph (615) 576-2496
C                              Internet: <LDT@STC10.CTD.ORNL.GOV>
C  whose efforts were funded by
C  the U.S. Department of Energy under Contract No. DE-AC05-76OR00001 and
C  the U.S. Enrichment Corp.     under Contract No. USECHQ-93-C-0001.
C
C This version adapted from SOLGASMIX-PV by P.K. Liao and K.E.Spear
C
C Modifications (1991-1993) by LDT & JML include:
C   Two phase transition bugs fixed; added externally controlled automated
C   operation; eliminated need for elemental species; detects number
C   of independent components; altered data file format with protection
C   against bad input; warns of possibly spurious solutions.  Changed to
C   use currently fashionable SI units.
C
C   Compiles with Ryan/McFarland Fortran (IBM PC); with minor changes, to
C   SOL244.FOR and DAT244.FOR, compiles with VAX (VMS?) Fortran (search
C   for the word "VAX", deactivate "R/M Fortran" lines, activate "VAX"
C   lines, and recompile).
C
C Limits:  Max Species  = 99          each of these limits is
C          Max Phases   = 20          protected from violation in the
C          Max Elements = 10          source file DAT244.FOR...
C          Max Start Mat= 20
C          Input Record Length < 501 characters    ... except this one.
C
      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL OPT12,FILEFLAG,UNDERFLG
      CHARACTER*8 EL
      CHARACTER*10 TEXT
      CHARACTER*80 TITLE
      CHARACTER*64 INFNAME,OUTFNAME,PRMFNAME
C
      COMMON DAKTF(99), AKTF(99), YTOT(20), PTOT, AKT(99), PI(20),
     $ YF(99), B(20,99), G(99), T, V, Y(99), KVAL1, KH(20),
     $ MA, MB, MO, NP, NO(99), NCODE, KQ, KS, PLUE, SLUE,
     $ APT(99), YM(99), OPT12, Rl, Rj
C
      COMMON /INPUT/ L,MP,MR,ML(20),MF(20),M1,MS,A(99,10),A0(99,10),
     $ XI(99),IOCRT,IOUT,IN,MIN,IIN(20),FILEFLAG
C
      COMMON /SPECIES/ TEXT(99),TITLE,EL(10),INFNAME
C
      COMMON /THERMO/ SO(99), HCT(99), HF(99), HFT(99), HOM(99,6),
     $ NOM(99), S(99), TOM(99,6), TSTART, OT0, OT1, C(99,6,5)
C
      COMMON /UNDERFLO/ UNDERFLG,DEXP_MIN
C
      DIMENSION BIN(20,99)
C
C The following are recommended by CODATA (1986), replacing earlier 8.31433
      Rj  =  8.3145107
C                       J/mole/K
      Rl  =  0.083145107
C                      l-bar/mole-K
C Maximum coefficient allowed by DEXP function
      DEXP_MIN = -709.7
C I/O Device numbers
      IPRM = 3
      IN   = 4
      IOCRT= 6
      IOUT = 7
      H =DLOG(1.D1)
      KVAL1 = 8
      MF(1) =1
      WRITE(IOCRT,*) '        **** SOLGAS Version 2.44 ****'
      WRITE(IOCRT,*)
C
C  *** Entry point for new data sets (option [8]) ***
C
   26 PTOT =1.
      ICHD=0
      WRITE(IOCRT,*)' Input File Name : '
      READ(*,910) INFNAME
      INQUIRE(FILE=INFNAME,EXIST=FILEFLAG)
      IF (FILEFLAG) GO TO 9901
      WRITE(*,9903) INFNAME
 9903 FORMAT(' Could not find the file ',A40)
      GO TO 26
 9901 WRITE(IOCRT,*)' Output File Name: '
      READ(*,910) OUTFNAME
      INQUIRE(FILE=OUTFNAME,EXIST=FILEFLAG)
      IF (.NOT.FILEFLAG) GO TO 9902
      WRITE(*,9904) OUTFNAME
 9904 FORMAT (' A file already exists by the name of ',A30,/,
     $' Use a different filename.')
      GO TO 9901
  910 FORMAT(A)
```

```
9902 V = 0.
     DO 48 K=1, 20
  48 KH(K) = 0
     OPEN(IOUT,FILE=OUTFNAME,STATUS='NEW')
     FILEFLAG=.FALSE.
     CALL READFILE
     IF (FILEFLAG) GOTO 26
     CALL DOF
     DO 1131 IOY=IOCRT,IOUT
1131 WRITE (IOY,113) TSTART
 113 FORMAT ('  Initial Temperature of Starting Materials =', F8.2,
    1' K',/)
  42 WRITE(IOCRT,*)' Input the Reaction Temperature (K): '
C
C ***** Entry point for temperature input (Menu Sel. [6])*******
C
     READ(*,*) T
 193 CALL HETTA
     GO TO 8
C
C  *** Menu selection [7] (const P or V specification) ****
C
  37 WRITE(IOCRT,*)' Input Pressure (as a positive number)',
    1' or Volume (as a negative number)'
     WRITE(IOCRT,*)' P(bar) or -V(liters) : '
     READ(*,*) PTOT
     V = DMAX1(-PTOT,0.D0)
C
C  **** Entry point for main menu ***
C
   8 CONTINUE
     WRITE(IOCRT,*)'  [1] Computer selects a starting estimate'
     WRITE(IOCRT,*)'  [2] Manually select a starting estimate'
     WRITE(IOCRT,*)'  [3] Change No. of pts. to calculate & quantities
    1of starting materials'
     WRITE(IOCRT,*)'  [4] Change quantities of starting material only'
     WRITE(IOCRT,*)'  [5] Recalc. with same data, pts. & quantities'
     WRITE(IOCRT,*)'  [6] Change the reaction temperature '
     WRITE(IOCRT,*)'  [7] Change value of fixed pressure or volume'
     WRITE(IOCRT,*)'  [8] Change to a new system'
     WRITE(IOCRT,*)'  [9] Exit program'
     WRITE(IOCRT,*)' [10] Save the last calculated result'
     WRITE(IOCRT,*)' [11] Save the last series of results'
     WRITE(IOCRT,*)' [12] Read run data from a parameter file'
     WRITE(IOCRT,*)' [13] Change thermodynamic data of a species'
     WRITE(IOCRT,*)
     WRITE(IOCRT,*)' Select an Option: '
     NCODE = 0
     READ(*,*) KVAL1
     IF (KVAL1 .GT. 9) NCODE = 1
     IOY=NCODE+IOCRT
     GO TO (40,40,40,7,34,42,37,26,1,64,34,1300,41), KVAL1
     WRITE(*,*) KVAL1,' is not a valid selection.'
     GOTO 8
  41 CALL CHANGEDATA
     ICHD=1
     GOTO 193
  40 WRITE(IOCRT,*)' Input No. of points to be calculated (<100),'
     WRITE(IOCRT,*)'  allowing for variation in quantities of starting
    1materials: '
     READ(*,*) NPKT
 828 WRITE(IOCRT,829) (TEXT(IIN(I)),I=1,MIN)
 829 FORMAT (1X,'Raw material species in sequence are :',/,10(2X,A10))
     WRITE(IOCRT,*)
 827 WRITE(IOCRT,*)'  [1]  N varies irregularly                  ',
    1'|  N is the quantity'
     WRITE(IOCRT,*)'  [2]  N is constant                         ',
    1'|     in moles'
     WRITE(IOCRT,*)'  [3]  N varies by a constant increment       |'
     WRITE(IOCRT,*)
     DO 902 I= 1, MIN
9020 WRITE(IOCRT,903) TEXT(IIN(I))
     READ(*,*) KH(I)
     IF ((KH(I).LE.3).AND.(KH(I).GE.1)) GOTO 902
     WRITE(*,*) ' Valid Options are 1, 2, and 3.'
     GOTO 9020
 902 CONTINUE
 903 FORMAT(1X,'Input option for ',A10,' :')
C
C *** Entry point for changing quantities (option [4]) ****
C
   7 DO 164 J=1, MIN
     N = KH(J)
     GO TO (161,162,163), N
 161 WRITE(IOCRT,904) TEXT(IIN(J)),KH(J),NPKT,NPKT
 825 CONTINUE
 904 FORMAT(1X,'Option for ',A10,' is ',I2,'; Input ',I2,
    $' value(s) for ',I2,' point(s): ')
C     WRITE(IOCRT,*)'  (Enter input values on one line separated by',
C    1' a comma or space)'
     READ(*,*) (B(J,N),N= 1,NPKT)
     GO TO 164
 162 WRITE(IOCRT,905) TEXT(IIN(J)),KH(J)
 823 CONTINUE
 905 FORMAT(1X,'Option for ',A10,' is ',I2,'; Input a '
    $,'constant N : ')
     READ(*,*) B(J,1)
     IF (NPKT .EQ. 1) GO TO 164
```

```
      DO 160 N= 2, NPKT
  160 B(J,N) = B(J,1)
      GO TO 164
  163 CONTINUE
      WRITE(IOCRT,906) TEXT(IIN(J)),KH(J)
  821 CONTINUE
  906 FORMAT(1X,'Option for ',A10,' is ',I2,','/1x,'Input the'
     $,' initial value & constant mole increment.')
      WRITE(IOCRT,*)' (Enter on one line separated by comma or space)'
      READ(*,*) B(J,1), STEP
      DO 165 N= 2, NPKT
  165 B(J,N) = B(J,N-1) + STEP
  164 CONTINUE
C
 1350 MREP = 0
      DO 23 J=1, MIN
      DO 23 N= 1, NPKT
   23 BIN(J,N) = B(J,N)
    4 CONTINUE
      IF (OPT12) GO TO 39
      IF (KVAL1 -2) 39,35,34
   35 CONTINUE
      WRITE(IOCRT,*)' Input initial estimate of the equilibrium'
     1,' composition for each species : '
      DO 3501 I=1,MS
 3502 FORMAT (1X,A10,': ')
      WRITE(*,3502) TEXT(I)
 3501 READ (*,*) Y(I)
      GO TO 166
C
C *** Entry point for re-tries on convergence failure  ***
C
   39 IF (MREP .EQ. 0) GO TO 761
      IF (MREP .GT. 1) GO TO 763
      BEST = 0.
      DO 762 I = 1, MIN
      BEST = BEST + DABS(B(I,1))
  762 CONTINUE
      GO TO 763
  761 BEST = 0.
      DO 169 J= 1, MIN
      IF (DABS(B(J,1)) .GT. BEST) BEST = DABS(B(J,1))
  169 CONTINUE
  763 BEST = BEST/FLOAT(MS)
      DO 167 I= 1, MS
  167 Y(I) = BEST
  166 DO 135 M= 1, MP
      YTOT(M) =0.
      MA=MF(M)
      MB = ML(M)
      DO 135 I= MA, MB
  135 YTOT(M) = YTOT(M) + Y(I)
C
C ***** Entry point for Re-calculation (Menu Sel [5]) ******
C
      IF (MREP .GT. 0) GO TO 192
C***** Equilibrium was not found on previous calc. ***
   34 NP=0
C
C ***** Entry for multiple point calculation ****
C
   27 NP= NP+ 1
  192 DO 190 J= 1, L
  190 B(J,NP) =0.
      DO 191 N=1, MIN
      I = IIN(N)
      XI(I) = BIN(N,NP)
      DO 191 J= 1, L
  191 B(J,NP) = B(J,NP) + A(I,J)*XI(I)
  189 DO 75 I=1, MS
   75 NO(I) = 1
      UNDERFLG = .FALSE.
      CALL GASOL
      IF (MO .LE. MP) GO TO 64
      WRITE (IOY,118)
      MREP = MREP +1
      IF (MREP .GE. 5) GO TO 234
      WRITE(IOY,233)
  233 FORMAT (' Trying different estimated equilibrium values ...'/)
      GO TO 39
C --- i.e. Exit to "re-try" routine ----
  234 CONTINUE
  118 FORMAT (/50H The equilibrium composition has not been obtained)
      IF (.NOT.OPT12) CALL PAUSE
      IF (OPT12) GO TO 1306
      IF (NP .LT. NPKT) MREP = 0
      IF (NP - NPKT) 27,8,8
C --- i.e. back to beginning, or to menu ---
   64 IF (ICHD.EQ.1) GOTO 11
      GOTO 12
   11 WRITE (IOY,13)
   13 FORMAT (/,' Thermodynamic values were changed while running.')
      WRITE (IOY,16)
   16 FORMAT (2X,'Name',10X,'DH298',7X,'S298',2X,'"a" of Cp')
      DO 14, I=1,MS
   14 WRITE (IOY,15) TEXT(I), HF(I), S(I), C(I,1,1)
   15 FORMAT (1X,A10,F12.2,F10.3,F10.3)
   12 WRITE (IOY, 1110) T
```

```
      IF (PTOT.GE.0.01.AND.PTOT.LT.1.0D5) WRITE (IOY, 1111) PTOT
      IF (PTOT.LT.0.01.OR.PTOT.GE.1.0D5) WRITE (IOY, 1112) PTOT
 1110 FORMAT (/,5H T = , F10.3, 2H K)
 1111 FORMAT (5H P = ,F10.3, 4H bar)
 1112 FORMAT (5H P = ,D10.3, 4H bar)
      VGAS=0.
      DO 6401 I=MF(1),ML(1)
 6401 VGAS=VGAS+Y(I)
      VGAS=VGAS*T*Rl/PTOT
      IF (VGAS.GE.0.01.AND.VGAS.LT.1.0D5) WRITE (IOY, 6402) VGAS
      IF (VGAS.LT.0.01.OR.VGAS.GE.1.0D5) WRITE (IOY, 6403) VGAS
 6402 FORMAT (' V = ',F10.3,' liters')
 6403 FORMAT (' V = ',D10.3,' liters')
      MREP = 0
      IF (MO .EQ. 0.) WRITE (IOY,121)
  121 FORMAT (/33H The small Y-values are not exact)
      IF (UNDERFLG) WRITE (IOY,1121)
 1121 FORMAT (/,' Warning: Underflow encountered during this run. Check
     1results carefully!')
      CALL CONSERVATION_OF_MASS(IOY)
  185 WRITE (IOY,105)
  105 FORMAT (/,15X,'Initial Qty',4X,'Final Qty ',5X,'Pressure'/,
     115X,9H  (moles),6X,8H (moles),9X,5H(bar)
C    2,9X,8HActivity
     3)
 1001 FORMAT (/,4X,'Mixture number ',I3, ' would be considered')
 1002 FORMAT (/,4X,'Species no. ',I3,' (',A10,') would be considered')
 1003 FORMAT (4X,'in the equilibrium set of phases if its free energy')
 1004 FORMAT (4X,'of formation were ',F12.1,' J/Mole more negative')
 1005 FORMAT(' ',12X,'Mixture ',I3)
 1006 FORMAT (' ',4X,'Species'
C    1,11X,'Activity'
     2)
 1007 FORMAT (' ',4X,A10,7X,E13.5)
      MIXN = 0
      DO 140 M = 1, MP
      BUPR = 0
      IF (M .GT. 1.) WRITE (IOY,125)
  125 FORMAT (41X, 13HMole fraction)
      MA = MF(M)
      MB = ML(M)
      DO 1040 I = MA,MB
      BUPR = BUPR + Y(I)
 1040 WRITE(IOY,124) TEXT(I),XI(I),Y(I),YF(I)
C    1,AKT(I)
      IF(BUPR .GT. 0) MIXN = MIXN + 1
  140 CONTINUE
  124 FORMAT (1X,A10, 4(2X, D13.5))
      IF(MR.LE.0) GOTO 802
```

```
      WRITE(IOY,120)(TEXT(I),XI(I),Y(I),I=M1,MS)
  120 FORMAT (/(1X,A10, 2(2X, D13.5)))
  802 CONTINUE
      IF(MIXN .EQ. MP) GO TO 1041
      WRITE(IOY,1001) KQ
      WRITE(IOY,1003)
      WRITE(IOY,1004) PLUE
      WRITE(IOY,1005) KQ
      WRITE(IOY,1006)
      MLBA = MF(KQ)
      MLBZ = ML(KQ)
      DO 1010 MLB = MLBA, MLBZ
 1010 WRITE(IOY,1007)TEXT(MLB), APT(MLB)
 1041 NIXT = 0
      DO 1015 NLB = M1, MS
 1015 IF(Y(NLB) .GT. 0) NIXT = NIXT + 1
      IF(NIXT .EQ. MR) GO TO 1042
      WRITE(IOY,1002)KS, TEXT(KS)
      WRITE(IOY,1003)
      WRITE(IOY,1004) SLUE
 1042 CALL HETTA
      IF(NCODE.NE.1) CALL PAUSE
      IF (OPT12) GO TO 1306
      IF (NP - NPKT) 27,8,8
C --- i.e., Do next point in series, or return to menu if series is done
    1 STOP
C
C Processing controlled by external "parameter file" - Menu Sel [12]
C
 1300 CONTINUE
 1302 WRITE(IOCRT,*) ' Parameter File Name:'
      READ(*,910) PRMFNAME
      INQUIRE(FILE=PRMFNAME,EXIST=FILEFLAG)
      IF (FILEFLAG) GO TO 1303
      WRITE(*,9903) PRMFNAME
      GO TO 1399
 1303 NMATLS = MIN
 1304 OPEN (IPRM,FILE=PRMFNAME,STATUS='OLD',ACTION='READ')              R/M Fortran
C1304 OPEN (IPRM,FILE=PRMFNAME,STATUS='OLD',READONLY)                   VAX
C1304 OPEN (IPRM,FILE=PRMFNAME,STATUS='OLD')                            1040ST
      DO 1305 I=1,MIN
      B(I,1) = 0.0
 1305 KH(I) = 2
      WRITE(IOCRT,*) ' Running SOLGAS under control of ',PRMFNAME
      WRITE(IOCRT,1391) (TEXT(IIN(I)),I=1,MIN)
 1306 CONTINUE
      READ (IPRM,*,END=1399) T, PTOT, (B(I,1), I=1,NMATLS)
      V = DMAX1(-PTOT,0.D0)
      WRITE(IOCRT,1392) T, PTOT, (B(I,1), I=1,NMATLS)
 1391 FORMAT ('           T(K)      P(bar)  ',20A10)
 1392 FORMAT (' Doing:',20(F8.3,' '))
      OPT12 = .FALSE.
      KVAL1 = 6
      CALL HETTA
      KVAL1 = 5
      NPKT = 1
      OPT12 = .TRUE.
      GOTO 1350
 1399 OPT12 = .FALSE.
      CLOSE (IPRM)
      WRITE(*,*)
      GO TO 8
      END
C
C
      SUBROUTINE ABER
      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL OPT12,FILEFLAG
C
      COMMON DAKTF(99), AKTF(99), YTOT(20), PTOT, AKT(99), PI(20),
     $ YF(99), B(20,99), G(99), T, V, Y(99), KVAL1, KH(20),
     $ MA, MB, MO, NP, NO(99), NCODE, KQ, KS, PLUE, SLUE,
     $ APT(99), YM(99), OPT12, Rl, Rj
C
      COMMON /INPUT/ L,MP,MR,ML(20),MF(20),M1,MS,A(99,10),AO(99,10),
     $ XI(99),IOCRT,IOUT,IN,MIN,IIN(20),FILEFLAG
C
      M = MO
      YTOT(M) = 0.
      DO 2 I = MA, MB
    2 YTOT(M) = YTOT(M) + Y(I)
      IF (YTOT(M) .GE. 1.D-8) GO TO 151
      YTOT(M) = 0.
      RETURN
C
  151 IF (M .GT. 1) GO TO 82
      IF (V .GT. 0.) PTOT = Rl*T*YTOT(1)/V
      YTOT(1) = YTOT(1)/PTOT
   82 DO 127 I = MA, MB
  127 YF(I) = Y(I)/YTOT(M)
C     CALL FACTOR
      DO 141 I = MA, MB
  141 AKT(I) = AKTF(I)*YF(I)
      RETURN
      END
C
C
      SUBROUTINE XBER
      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL OPT12,UNDERFLG,FILEFLAG
C
      COMMON DAKTF(99), AKTF(99), YTOT(20), PTOT, AKT(99), PI(20),
     $ YF(99), B(20,99), G(99), T, V, Y(99), KVAL1, KH(20),
     $ MA, MB, MO, NP, NO(99), NCODE, KQ, KS, PLUE, SLUE,
     $ APT(99), YM(99), OPT12, Rl, Rj
C
      COMMON /INPUT/ L,MP,MR,ML(20),MF(20),M1,MS,A(99,10),AO(99,10),
     $ XI(99),IOCRT,IOUT,IN,MIN,IIN(20),FILEFLAG
C
      COMMON /UNDERFLO/ UNDERFLG,DEXP_MIN
C
      DIMENSION OYF(99)
C
      M = MO
      PLOG = DMAX1(DLOG(PTOT),1.D1)
      MA = MF(M)
      MB = ML(M)
      DO 38 I = MA, MB
      IF (NO(I) .EQ. 0 .OR. Y(I) .GT. 0.) GO TO 38
      PIA = -G(I)
      DO 87 J = 1, L
   87 PIA = PIA + A(I,J)*PI(J)
      IF (PIA .GT. PLOG) PIA = PLOG
      IF (PIA .GT. DEXP_MIN) THEN
         AKT(I) = DEXP(PIA)
      ELSE
         AKT(I) = 0.0
         UNDERFLG = .TRUE.
      END IF
      YF(I) = AKT(I)
   38 CONTINUE
      IVAR = 0
  147 IVAR = IVAR + 1
      IF (IVAR .EQ. 75) RETURN
C
C     CALL FACTOR
      DO 139 I = MA, MB
      IF (AKTF(I) .EQ. 1. .OR. YF(I) .EQ. 0.) GO TO 139
C
C  Following code, through 139, is probably never accessed in this version
C
      OYF(I) = YF(I)
      ACT = AKTF(I)*YF(I)
C
      WRITE(*,*) 'Warning: Undefined variable DAKTF was accessed.'      trace
C
      YF(I) = YF(I) - ACT*DLOG(ACT/AKT(I))/(DAKTF(I)*YF(I) + AKTF(I))
      IF (YF(I) .LT. 0.) YF(I) = 0.1*OYF(I)
  139 CONTINUE
      DO 137 I = MA, MB
      IF (AKTF(I) .EQ. 1. .OR. YF(I) .EQ. 0.) GO TO 137
      IF (DABS(OYF(I)/YF(I) - 1.) .GT. 1.D-4) GO TO 147
  137 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE HETTA
      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL OPT12,FILEFLAG
      COMMON DAKTF(99), AKTF(99), YTOT(20), PTOT, AKT(99), PI(20),
     $ YF(99), B(20,99), G(99), T, V, Y(99), KVAL1, KH(20),
     $ MA, MB, MO, NP, NO(99), NCODE, KQ, KS, PLUE, SLUE,
     $ APT(99), YM(99), OPT12, Rl, Rj
C
      COMMON /INPUT/ L,MP,MR,ML(20),MF(20),M1,MS,A(99,10),AO(99,10),
     $ XI(99),IOCRT,IOUT,IN,MIN,IIN(20),FILEFLAG
C
      COMMON /THERMO/ SO(99), HCT(99), HF(99), HFT(99), HOM(99,6),
     $ NOM(99), S(99), TOM(99,6), TSTART, OTO, OT1, C(99,6,5)
C
      TREF = 298.15
      IOY=NCODE+6
      OTO = 0.
      OT1 = 0.
      GO TO (78,78,78,78,78,41,41,41,999,78,78,41,41), KVAL1
   41 CONTINUE
      DO 95 I = 1, MS
   95 HFT(I) = HF(I) + HEAT(NOM(I),TREF,T,I,SO(I))
C
C HEAT calculates entropy change as well as heat content. S is discarded
C when calculating starting material heat, a minor one-time inefficiency.
C
      DO 28 N = 1, MIN
      I = IIN(N)
   28 HCT(I) = HEAT(NOM(I),TSTART,T,I,S_DUMMY)
      RTI = 1./(Rj*T)
      DO 174 I=1,MS
  174 G(I) = RTI*(HFT(I) - T*(S(I) + SO(I)))
      IF (OPT12) GO TO 78
      RETURN
C
C  Calculate heat change from starting mix to equilibrium mix
C
   78 HP = 0.
      HR = 0.
      DO 79 N = 1, MIN
      I = IIN(N)
      HP = HP + HCT(I)*XI(I)
   79 HR = HR - HFT(I)*XI(I)
      DO 32 I = 1, MS
   32 HR = HR + HFT(I)*Y(I)
      HT = HP + HR
      WRITE (IOY,109) HP*.001, HR*.001, HT*.001
  109 FORMAT (/,
     $20H Pre-heat         = ,F10.3, 3H kJ/,
     $20H Heat of reaction = ,F10.3, 3H kJ/,
     $20H Total heat       = ,F10.3, 3H kJ/)
  999 RETURN
      END
C
C
      FUNCTION HEAT (NM,T_INIT,T_FINAL,M,ENTROPY)
C
C     Calculates molar enthalpy change from T(init) to T(final) for a
C     single species. Heat difference is returned with the function call;
C     Entropy difference is also found and is returned via parameter list.
C
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER EQN
C
      COMMON /THERMO/ SO(99), HCT(99), HF(99), HFT(99), HOM(99,6),
     $ NOM(99), S(99), TOM(99,6), TSTART, OTO, OT1, C(99,6,5)
C
      SIGN=1.0
      TI=T_INIT
      TF=T_FINAL
      IF (TI.LT.TF) GOTO 1100
      SIGN=TF
      TF=TI
      TI=SIGN
      SIGN=-1.
1100  EQN=0
      HEAT=0.
      S_CUM=0.
1101  EQN=EQN+1
      IF (EQN.EQ.NM) GOTO 1102
      IF (TI.LT.TOM(M,EQN)) GOTO 1102
      GOTO 1101
1102  TTI=TI
      IF (EQN.EQ.NM) GOTO 1105
      IF (TF.LE.TOM(M,EQN)) GOTO 1105
      TTF=TOM(M,EQN)
      HEAT=HEAT+HOM(M,EQN)
      S_CUM=S_CUM+HOM(M,EQN)/TOM(M,EQN)
      GOTO 1110
 1105 TTF=TF
 1110 HEAT = HEAT + CPINT(1,M,EQN,TTI,TTF)
      S_CUM = S_CUM + CPINT(2,M,EQN,TTI,TTF)
      TI=TTF
      IF (TI.LT.TF) GOTO 1101
      HEAT=SIGN*HEAT
      ENTROPY=SIGN*S_CUM
      RETURN
      END
C
C
      FUNCTION CPINT(N,I,J,T0,T1)
C
C  N : Select dH or dS integration of Cp equation
C  I : species #
C  J : Cp Eqn ID#
C  T0 and T1 : temperature integration limits
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON /THERMO/ S0(99), HCT(99), HF(99), HFT(99), HOM(99,6),
     $ NOM(99), S(99), TOM(99,6), TSTART, OT0, OT1, C(99,6,5)
C
      COMMON/CPHOLD/T2,T3,T4,TA,TB,TC,TD,TE,TF,TG
C
C   Re-calculate Temperature polynomial terms only when they change...
C
      IF (T0 .NE. OT0 .OR. T1 .NE. OT1) GO TO 3
      GO TO (1,2) N
    3 OT0 = T0
      OT1 = T1
      T2 = T0+T1
      T3 = T0*T1
      T4 = T3*T3
      TA = T1-T0
      TB = TA*T2/2.
      TC = TA*(T2*T2-T3)/3.
      TD = DLOG(T1/T0)
      TE = TA/T3
      TF = TB/T4
      TG = TC/(T3*T4)
      GO TO (1,2) N
C
C Enthalpy calculation:
C
    1 CPINT = C(I,J,1)*TA+C(I,J,2)*TB+C(I,J,3)*TC+C(I,J,4)*TE+
     $C(I,J,5)*TF
      RETURN
```

```
C
C Entropy calculation:
C
    2 CPINT = C(I,J,2)*TA+C(I,J,3)*TB+C(I,J,1)*TD+C(I,J,4)*TF+
     $C(I,J,5)*TG
      RETURN
      END
C
C
C      SUBROUTINE FACTOR
C  Inactive routine (atrophied, or a place holder for future developments)
C      RETURN
C      END
C
C      SUBROUTINE SPEQUA
C     ...a subroutine for calculation of quantities which are
C        derivable from the equilibrium composition. Inactive...
C      RETURN
C      END
C
C
      SUBROUTINE PAUSE
C
C *** Bandaid for flaky response to "PAUSE" in one obscure FORTRAN dialect ***
C
      CHARACTER*1 Q
      WRITE(*,1)
    1 FORMAT (' Paused -- Push RETURN to continue')
      READ (*,2) Q
    2 FORMAT (A)
      RETURN
      END
C
C
C
      SUBROUTINE CONSERVATION_OF_MASS(IODEVICE)
      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL OPT12,FILEFLAG,UNDERFLG
      CHARACTER*8 EL
      CHARACTER*10 TEXT
      CHARACTER*80 TITLE
      CHARACTER*64 INFNAME
C
      COMMON DAKTF(99), AKTF(99), YTOT(20), PTOT, AKT(99), PI(20),
     $ YF(99), B(20,99), G(99), T, V, Y(99), KVAL1, KH(20),
     $ MA, MB, MO, NP, NO(99), NCODE, KQ, KS, PLUE, SLUE,
     $ APT(99), YM(99), OPT12, Rl, Rj
C
      COMMON /INPUT/ L,MP,MR,ML(20),MF(20),M1,MS,A(99,10),A0(99,10),
     $ XI(99),IOCRT,IOUT,IN,MIN,IIN(20),FILEFLAG
C
      COMMON /SPECIES/ TEXT(99),TITLE,EL(10),INFNAME
C
      DIMENSION COM(10,2)
      LOGICAL TITLEPRT
C
      TITLEPRT=.FALSE.
      DO 1 I_ELEM=1,10
      COM(I_ELEM,1)=0.0
    1 COM(I_ELEM,2)=0.0

      DO 10 I_ELEM=1,L
      DO 10 I_MATL=1,MS
      COM(I_ELEM,1)=COM(I_ELEM,1)+A(I_MATL,I_ELEM)*XI(I_MATL)
      COM(I_ELEM,2)=COM(I_ELEM,2)+A(I_MATL,I_ELEM)*Y(I_MATL)
   10 CONTINUE
C
      DO 25 I_ELEM=1,L
      DEFECT=COM(I_ELEM,2)/COM(I_ELEM,1)-1.0
      IF (ABS(DEFECT).LT.1.0D-09) GOTO 25
      IF (TITLEPRT) GOTO 24
      WRITE(IODEVICE,*) 'Warning: Mass balance not preserved in this run
     1. Examine results carefully!'
      WRITE(IODEVICE,*) 'Element  |  Mass Balance Fractional Error'
      TITLEPRT=.TRUE.
   24 WRITE(IODEVICE,20) EL(I_ELEM),DEFECT
   25 CONTINUE
C
   20 FORMAT ( ' ',A8,' |      ',D10.3)
      RETURN
      END
C
C
C
      SUBROUTINE GASOL
C
C     A source file for SOLGAS, version 2.44
C
      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL OPT12,FILEFLAG
C
      COMMON DAKTF(99), AKTF(99), YTOT(20), PTOT, AKT(99), PI(20),
     $ YF(99), B(20,99), G(99), T, V, Y(99), KVAL1, KH(20),
     $ MA, MB, MO, NP, NO(99), NCODE, KQ, KS, PLUE, SLUE,
     $ APT(99), YM(99), OPT12, Rl, Rj
C
      COMMON /INPUT/ L,MP,MR,ML(20),MF(20),M1,MS,A(99,10),A0(99,10),
     $ XI(99),IOCRT,IOUT,IN,MIN,IIN(20),FILEFLAG
```

```
C
      DIMENSION BO(20), F(99), IFAS(20), ISOL(20), LX(99), NSUM(99),
     $OPI(10), R(20,21), YFTOT(20), YX(99)
C
      MX = 0
      BMAX = 0.
      DO 183 J = 1, L
      IF(DABS(B(J,NP)) .GT. BMAX) BMAX=DABS(B(J,NP))
  183 BO(J) = B(J,NP)
      BMAX = BMAX/FLOAT(MS)
  171 IS = -1
      NG = 0
   55 ISUM = 0
      MSA = 0
      IF (MS .LT. M1) GO TO 47
      DO 52 I = M1, MS
      IF (Y(I) .EQ. 0.) GO TO 52
      MSA = MSA + 1
      ISOL(MSA) = I
      ISUM = ISUM + 2**(I + MP - M1)
   52 CONTINUE
   47 MPA = 0
      NG = NG + 1
      IF (NG .EQ. 100) NG = 1
      NSUM(NG) = ISUM
      YSUM = 0.
      DO 152 M = 1, MP
      IF (YTOT(M) .EQ. 0.) GO TO 152
      MPA = MPA + 1
      IFAS(MPA) = M
      NSUM(NG) = NSUM(NG) + 2**(M - 1)
      YSUM = YSUM + YTOT(M)
  152 CONTINUE
      IF (NSUM(NG) .LT. IS .OR. MPA + MSA .GT. L) GO TO 59
      GO TO 69
   86 NG = NG - 1
   59 IS = IS + 2
      DO 154 N = 1, MPA
      M = IFAS(N)
  154 YTOT(M) = 0.
      ISUM = 0
      MPA = 1
      YTOT(1) = 1.
      IF (MSA .EQ. 0) GO TO 73
      DO 68 N = 1, MSA
      I = ISOL(N)
   68 Y(I) = 0.
      MSA = 0
   73 IF (IS .EQ. 1) GO TO 74
```

```
      IT = IS
   61 M = 0
   71 M = M + 1
      IF (2**M .LE. IT) GO TO 71
      IF (M .GT. MP) GO TO 57
      MPA = MPA + 1
      IFAS(MPA) = M
      YTOT(M) = 1.
      MA = MF(M)
      MB = ML(M)
      DO 150 I = MA, MB
      IF (NO(I) .EQ. 1) Y(I) = YSUM
  150 CONTINUE
      GO TO 136
   57 I = M + ML(MP) - MP
      MO = M
      IF (I .GT. MS) RETURN
      IF (NO(I) .EQ. 0) GO TO 59
      MSA = MSA + 1
      ISOL(MSA) = I
      ISUM = ISUM + 2**(M - 1)
      Y(I) = YSUM
  136 IT = IT - 2**(M - 1)
      IF (IT .GT. 1) GO TO 61
      IF (MPA + MSA .GT. L) GO TO 59
   74 IFAS(1) = 1
      IF (IS .LE. IABS(NSUM(NG))) NG = NG + 1
      IF (NG .EQ. 100) NG = 1
      NSUM(NG) = IS
      IF (NG .GT. 1) NSUM(NG-1) = IABS(NSUM(NG-1))
   69 IF (NG .EQ. 1) GO TO 129
      DO 148 K = 2, NG
      IF (NSUM(NG) .EQ. NSUM(K-1)) GO TO 86
  148 CONTINUE
      IF (NG .EQ. 2) GO TO 129
C      MSUM = -NSUM(NG-2)
C *** For unknown reasons, the above line chokes R/M Fortran ***
C *** when replaced with the two below, everything is OK     ***
      IIII = NSUM(NG-2)
      MSUM = -IIII
      IF (MSUM.LT.0.OR.NSUM(NG).EQ.MSUM.AND.NSUM(NG-1).LT.0) GO TO 129
      NSUM(NG-2) = MSUM
      IF (NSUM(NG) .EQ. MSUM) GO TO 86
  129 DO 142 M = 1, MP
      MA = MF(M)
      MB = ML(M)
      IF (YTOT(M) .GT. 0.) GO TO 130
      DO 126 I = MA, MB
      AKT(I) = 0.
      AKTF(I) = 1.
      Y(I) = 0.
  126 YF(I) = 0.
      GO TO 142
  130 DO 45 I = MA, MB
      IF (Y(I) .GT. BMAX) Y(I) = BMAX
      IF (NO(I) .EQ. 1 .AND. Y(I) .LT. 1.D-8) Y(I) = 1.D-8
      AKTF(I) = 1.
   45 LX(I) = 0
      MO = M
      CALL ABER
      IF (YTOT(M) .EQ. 0.) GO TO 47
  142 CONTINUE
      LS1 = L + MPA + MSA
      LS = LS1 - 1
      LS2 = LS + 2
  134 DMIN = 1.D-6
      IVAR = 0
      IVARJ = ML(1) - MS
   16 DO 6 J = 1, LS1
      DO 6 K = J, LS2
    6 R(J,K) = 0.
      DO 9 N = 1, MPA
      L1 = L + N
      M = IFAS(N)
      MA = MF(M)
      MB = ML(M)
      DO 5 I = MA, MB
      IF (Y(I) .EQ. 0.) GO TO 5
      F(I) = G(I) + DLOG(AKT(I))
      R(L1,LS2) = R(L1,LS2) + F(I)*Y(I)
      DO 77 J = 1, L
      IF (A(I,J) .EQ. 0.) GO TO 77
      AY = A(I,J)*Y(I)
      R(J,L1) = R(J,L1) + AY
      R(J,LS2) = R(J,LS2) + AY*F(I)
      DO 80 K = J, L
   80 R(J,K) = R(J,K) + AY*A(I,K)
   77 CONTINUE
    5 CONTINUE
      DO 9 J = 1, L
    9 R(J,LS2) = R(J,LS2) - R(J,L1)
      IF (MSA .EQ. 0) GO TO 63
      DO 67 N = 1, MSA
      I = ISOL(N)
      K = L + MPA + N
      R(K,LS2) = G(I)
      DO 67 J = 1, L
   67 R(J,K) = A(I,J)
```

```
 63 DO 31 J = 2, LS1
    N = J - 1
    DO 31 K = 1, N
 31 R(J,K) = R(K,J)
    IF (MX .EQ. 0) GO TO 156
    DO 131 J = 1, L
131 B0(J) = B(J,NP)
    DO 172 N = 1, MPA
    M = IFAS(N)
    IF (MF(M) .NE. ML(M)) GO TO 172
    MA = MF(M)
    DO 89 J = 1, L
    IF (A(MA,J) .EQ. A0(MA,J)) GO TO 89
    AKVOT = (1. - A(MA,J)/A0(MA,J))*Y(MA)
    DO 91 K = 1, L
    IF (A(MA,K) .EQ. A0(MA,K)) B0(K) = B0(K) + AKVOT*A(MA,K)
 91 CONTINUE
    GO TO 172
 89 CONTINUE
172 CONTINUE
156 DO 44 K = 1, L
 44 R(K,LS2) = R(K,LS2) + B0(K)
    DO 10 K = 1, LS
    ELMAX = 0.
    DO 11 J = K, LS1
    IF (DABS(R(J,K)) .LE. ELMAX) GO TO 11
    MROW = J
    ELMAX = DABS(R(J,K))
 11 CONTINUE
    IF (ELMAX .GT. 0.) GO TO 36
    IF (K .GT. L) GO TO 10
    IF (B0(K)) 59,10,59
 36 IF (MROW .EQ. K) GO TO 13
    DO 15 N = K, LS2
    RADBYT = R(MROW,N)
    R(MROW,N) = R(K,N)
 15 R(K,N) = RADBYT
 13 KA = K + 1
    DO 46 J = KA, LS1
    RKVOT = R(J,K)/R(K,K)
    DO 46 N = KA, LS2
 46 R(J,N) = R(J,N) - RKVOT*R(K,N)
 10 CONTINUE
    DO 20 N = 1, LS1
    K = LS2 - N
    IF (R(K,K) .NE. 0. .AND. R(K,LS2) .NE. 0.) GO TO 62
    PI(K) = 0.
    K = K - L - MPA
    IF (K .LE. 0) GO TO 20
```

```
    I = ISOL(K)
    Y(I) = 0.
    GO TO 55
 62 PI(K) = R(K,LS2)/R(K,K)
    KA = K - 1
    IF (KA .EQ. 0) GO TO 20
    DO 58 J = 1, KA
 58 R(J,LS2) = R(J,LS2) - PI(K)*R(J,K)
 20 CONTINUE
    IF (IVAR .EQ. 0 .OR. IVARJ .GE. 0 .OR. SLAM .LT. 0.1) GO TO 66
    DO 70 J = 1, L
    IF (DABS(PI(J)) .LT. 1.D-8) GO TO 70
    IF (DABS(OPI(J)/PI(J) - 1.) .GT. DMIN) GO TO 65
 70 CONTINUE
    NR = 0
    IF (NG .EQ. 1) GO TO 155
    DO 157 K = 2, NG
    IF (NSUM(NG) .EQ. -NSUM(K-1)) NR = NR + 1
157 CONTINUE
155 PIXMAX = 0
    DO 143 M = 1,MP
    IF (YTOT(M) .GT. 0.) GO TO 143
    MO = M
    CALL XBER
    YFTOT(M) = 0.
    DO 144 I = MA, MB
    YFTOT(M) = YFTOT(M) + YF(I)
    YX(I) = YF(I)
    AKT(I) = 0.
144 YF(I) = 0.
    IF (M .EQ. 1) YFTOT(M) = YFTOT(M)/PTOT
    IF(YFTOT(M) .GT. PIXMAX) PIXMAX = YFTOT(M)
    IF(YFTOT(M) .EQ. PIXMAX) KQ = M
    PFTOT = YFTOT(KQ)
    IF (M .NE. KQ .OR. PFTOT .EQ. 0) GO TO 143
    MA = MF(KQ)
    MB = ML(KQ)
    DO 792 I = MA,MB
    YF(I) = YX(I) / PFTOT
792 CONTINUE
    MA = MF(KQ)
    MB = ML(KQ)
    DO 787 I = MA,MB
    YM(I) = YF(I)
    YF(I) = 0
    AKT(I) = 0
787 CONTINUE
143 CONTINUE
    PLUM = 0
```

```
      PLUG = 0
C
C Next Line is a Bandaid fix to an array bounds error in MF and ML.
C This luckily did not usually affect the result, as R/M cheerfully fetches
C out-of-bounds array elements, and these happened to be seldom used.
C
      IF((KQ.LE.0).OR.(KQ.GT.20)) KQ=1
      MA = MF(KQ)
      MB = ML(KQ)
      DO 783 I = MA,MB
      APT(I) = AKTF(I)*YM(I)
      IF(APT(I) .LE. 0) GO TO 783
      PLUG = PLUG + DLOG(APT(I)) + G(I)
      DO 784 J = 1,L
      IF(A(I,J) .EQ. 0) GO TO 784
      PLUM = PLUM + A(I,J) * PI(J)
  784 CONTINUE
  783 CONTINUE
      PLUS = PLUG - PLUM
      PLUE = Rj*T*PLUS
      INEW = 0
  145 DIFM = 1.
      DO 158 M = 1, MP
      IF (YTOT(M) .GT. 0. .OR. YFTOT(M) .LE. DIFM) GO TO 158
      KA = M
      DIFM = YFTOT(M)
  158 CONTINUE
      IF (DIFM .EQ. 1. .AND. INEW .EQ. 0) GO TO 138
      IF (DIFM .EQ. 1. .AND. INEW .EQ. 1) GO TO 59
      INEW = 1
      IF (NR .EQ. 0) GO TO 159
      NR = NR - 1
      YFTOT(KA) = 1.
      GO TO 145
  159 NSUM(NG) = -NSUM(NG)
      YTOT(KA) = 1.
      MA = MF(KA)
      MB = ML(KA)
      DO 153 I = MA, MB
  153 Y(I) = YSUM*YX(I)/YFTOT(KA)
      GO TO 47
  138 IF (MS .LT. M1) GO TO 51
      DIFM = 0.
      COMAX = -1.0D60
      DO 54 I = M1, MS
      IF (NO(I) .EQ. 0 .OR. Y(I) .GT. 0.) GO TO 54
      PIA = -G(I)
      DO 56 J = 1, L
   56 PIA = PIA + A(I,J)*PI(J)
      IF(PIA .GT. COMAX) COMAX = PIA
      IF(COMAX .EQ. PIA) KS = I
      IF (PIA .LT. DIFM) GO TO 54
      KA = I
      DIFM = PIA
   54 CONTINUE
      SLUM = 0
      SLUG = G(KS)
      DO 687 J = 1,L
      IF(A(KS,J) .EQ. 0) GO TO 687
      SLUM = SLUM + A(KS,J) * PI(J)
  687 CONTINUE
      SLUS = SLUG - SLUM
      SLUE = Rj*T*SLUS
      IF (DIFM .EQ. 0.) GO TO 51
      Y(KA) = YSUM
      GO TO 55
   51 IF (MX .EQ. 1) GO TO 93
      DO 168 N = 1, MPA
      M = IFAS(N)
      IF (MF(M) .NE. ML(M)) GO TO 168
      MA = MF(M)
      DO 173 J = 1, L
      IF (A(MA,J) .EQ. AO(MA,J)) GO TO 173
      MX = 1
      GO TO 171
  173 CONTINUE
  168 CONTINUE
   93 IVARJ = 0
      GO TO 66
   65 DO 60 J = 1, L
   60 OPI(J) = PI(J)
   66 SLAM = 1.
      DO 12 N = 1, MPA
      L1 = L + N
      M = IFAS(N)
      MA = MF(M)
      MB = ML(M)
      DO 12 I = MA, MB
      IF (Y(I) .EQ. 0.) GO TO 12
      PIA = F(I) - PI(L1)
      DO 19 J = 1, L
   19 PIA = PIA - A(I,J)*PI(J)
      YX(I) = PIA*Y(I)
      IF (PIA .GT. SLAM) SLAM = PIA
   12 CONTINUE
      IF (SLAM .GT. 1.) SLAM = 0.999*(SLAM - 0.5)/(SLAM*SLAM)
      IF (MSA .EQ. 0) GO TO 72
      DO 53 N = 1, MSA
      I = ISOL(N)
      K = L + MPA + N
      IF (PI(K) .LT. 0.) L1 = -I
      IF (IVAR.GT.0.AND.PI(K).LT.-Y(I).AND.-PI(K)/YSUM.GT.1.D8) GO TO 99
   53 Y(I) = DABS(PI(K))
   72 YSUM = 0.
      DO 128 N = 1, MPA
      M = IFAS(N)
      MA = MF(M)
      MB = ML(M)
      DO 29 I = MA, MB
      IF (Y(I) .EQ. 0.) GO TO 29
      Y(I) = Y(I) - SLAM*YX(I)
      IF (Y(I) .LT. 1.E-10) Y(I) = 0.
   29 CONTINUE
      MO = M
      CALL ABER
      IF (YTOT(M) .GT. 0.) GO TO 128
      NSUM(NG) = -NSUM(NG)
      GO TO 47
  128 YSUM = YSUM + YTOT(M)
      IVAR = IVAR + 1
      IF (IVAR .EQ. 25 .OR. IVAR .EQ. 50) DMIN = 100.*DMIN
      IF (IVAR .EQ. 75) GO TO 59
      IF (IVARJ .LT. 0 .OR. SLAM .LT. 1.) GO TO 16
      IF (L1 .GT. 0) GO TO 25
   99 Y(-L1) = 0.
      GO TO 55
   25 IVARJ = IVARJ + 1
      IF (IVARJ .EQ. 10) GO TO 88
      DO 3 N = 1, MPA
      M = IFAS(N)
      MA = MF(M)
      MB = ML(M)
      DO 3 I = MA, MB
      IF (Y(I) .EQ. 0.) GO TO 3
      IF (DABS(YX(I))/Y(I) .GT. 1.D-6) GO TO 16
    3 CONTINUE
   88 DO 149 N = 1, MPA
      M = IFAS(N)
      MO = M
      CALL XBER
      IF (MA .NE. MB) GO TO 133
      DO 92 J = 1, L
      IF (A(MA,J) .NE. AO(MA,J)) Y(MA) = A(MA,J)/AO(MA,J)*Y(MA)
   92 CONTINUE
  133 DO 149 I = MA, MB
      IF (NO(I) .EQ. 0 .OR. Y(I) .GT. 0.) GO TO 149
      Y(I) = YTOT(M)*YF(I)
      IF (Y(I) .LT. 1.D-10 .OR. LX(I) .EQ. 1) GO TO 149
      IF (M .EQ. 1 .AND. YF(I) .GT. PTOT) Y(I) = YTOT(1)*PTOT
      IF (M .GT. 1 .AND. YF(I) .GT. 1.) Y(I) = YTOT(M)
      LX(I) = 1
      GO TO 134
  149 CONTINUE
      IF (IVARJ .EQ. 10) MO = 0
      RETURN
      END
C
C
      SUBROUTINE READFILE
      IMPLICIT REAL*8 (A-H,O-Z)
C
C Array    Max Species  = 99
C Limits:  Max Elements = 10
C          Max Start Mat= 20
C          Max mixtures = 20
C          Max size of data line = 500 characters
C          Min # of gases = 1
C
C   v 2.44, using "NEW" (> Mar 93) SOLGAS input format processing
C
      CHARACTER*8 EL
      CHARACTER*10 TEXT
      CHARACTER TITLE*80
      CHARACTER*64 INFNAME,OUTFNAME,PRMFNAME
      LOGICAL FILEFLAG
C
      COMMON /INPUT/ L,MP,MR,ML(20),MF(20),M1,MS,A(99,10),AO(99,10),
     $ XI(99),IOCRT,IOUT,IN,MIN,IIN(20),FILEFLAG
C
      COMMON /SPECIES/ TEXT(99),TITLE,EL(10),INFNAME
C
      COMMON /THERMO/ S0(99), HCT(99), HF(99), HFT(99), HOM(99,6),
     $ NOM(99), S(99), TOM(99,6), TSTART, OT0, OT1, C(99,6,5)
C
C  *** END solgas Common; Local data definitions begin:
C
      CHARACTER CHR500(500)*1,STR5*5,CHR10(10)*1,STR10*10,TITSTR*80
     1 ,STR500*500,STR2*2,STR80*80,TEMP_STR*78,CHR30(30)*1,STR30*30
     2 ,old_type*1,cur_type*1,E8*8,C8(8)*1,S30*30,C30(30)*1,STR3*3
      EQUIVALENCE (CHR10,STR10),(STR500,CHR500(1),STR5,STR2,STR3),
     1 (CHR500(6),TITSTR),(CHR500(3),TEMP_STR),(CHR30,STR30),
     2 (S30,C30),(C8,E8)
      INTEGER tmp,IP,IK,OLDMIXNO
      DOUBLE PRECISION X(7,6),SUBVAL
C
C  Initialize and zero variables:
```

```
C
      maxele=10
      maxstmat=20
      maxmix=20
      maxsp=99
      tmp=3
      MIN=0
      OLDMIXNO=1
      MP=1
      MR=0
      L=0
      old_type='G'
      DO 111 I=1,maxele
  111 EL(I)='         '
      DO 112 I=1,maxsp
      DO 112 J=1,maxele
  112 A(I,J)=0
      DO 113 I=1,maxmix
  113 ML(I)=0
      open(unit=IN,file=INFNAME,status='old',action='read')
      OPEN(unit=tmp,STATUS='SCRATCH',ACTION='READ/WRITE')
C     open(unit=IN,file=INFNAME,status='old',READONLY)
C     OPEN(unit=tmp,STATUS='SCRATCH',RECL=500)
C     OPEN(unit=IN,file=INFNAME,STATUS='OLD')
C     OPEN(unit=tmp,STATUS='SCRATCH')
C
C  Process TITLE line
C
      READ(IN,2) STR500
C     READ(IN,1) (CHR500(I),I=1,500)
    1 FORMAT (500A1)
    2 FORMAT (A500)
    3 FORMAT (1X,500A1)
      DO 74 K=1,500
   74 IF (CHR500(K).EQ.','.OR.CHR500(K).EQ.'-') CHR500(K)=' '
      IF (STR5.EQ.'TITLE') GOTO 709
      WRITE(*,*) 'First line of data file must begin with "TITLE":'
      GOTO 7776
  709 TITLE=TITSTR
C
C  Process Comments and Temperature line:
C
  710 CONTINUE
      read(IN,2,END=732) STR500
C     READ(IN,1,END=732) (CHR500(I),I=1,500)
      DO 73 K=1,500
   73 IF (CHR500(K).EQ.',') CHR500(K)=' '
      IF (STR2.NE.'T='.AND.STR3.NE.'T =') GOTO 710
      GOTO 75
```

```
          732 WRITE(*,*) 'No properly formatted starting material temperature li
             1ne found in data file.'
              GOTO 7777
           75 IF (CHR500(3).EQ.'=') CHR500(3)=' '
              REWIND (tmp)
              WRITE(tmp,*) TEMP_STR
              REWIND (tmp)
              READ(tmp,*) TSTART
        C
        C  Species line loop
        C
              DO 999 MS=1,maxsp+1
              read(IN,2,ERR=15,END=16) STR500                           VAX&RMF
        C     READ(IN,1,ERR=15,END=16) (CHR500(I),I=1,500)              1040ST
              GOTO 20
           15 WRITE(*,*) 'Error encountered reading the input file'
              GOTO 7776
           20 CONTINUE
              DO 60 k=1,500
R/M        60 IF (CHR500(K).eq.',') CHR500(k)=' '
R/M           K=0
VAX        61 K=K+1
VAX           IF (CHR500(K).EQ.' ') GOTO 61
1040STC
1040STC    Process "species type"
        C
              IF (CHR500(K).NE.'*') GOTO 613
              K=K+1
VAX&RMF       MIN=MIN+1
1040ST        if (MIN.LE.maxstmat) GOTO 6121
              WRITE(*,*) 'Too many starting materials; only 20 are allowed.'
              GOTO 7776
         6121 IIN(MIN)=MS
              IF (CHR500(K).EQ.' ') K=K+1
          613 CONTINUE
              cur_type=CHR500(K)
              if (cur_type.EQ.old_type) GOTO 615
              if ((cur_type.EQ.'M').AND.(old_type.EQ.'G')) goto 615
              if ((cur_type.EQ.'P').AND.(old_type.EQ.'G')) goto 615
              if ((cur_type.EQ.'P').AND.(old_type.EQ.'M')) goto 615
              WRITE(*,*) 'Invalid Species type or sequence (G...M...P)'
              GOTO 7776
        C
VAX&RMF615    old_type=cur_type
1040ST        if (cur_type.EQ.'G') ML(1)=ML(1)+1
              if (cur_type.EQ.'P') MR=MR+1
              IF (cur_type.NE.'M') GOTO 62
              J=0
              STR10='          '
```

```
      IF(CHR500(K+1).EQ.' ') K=K+1
  616 K=K+1
      IF(CHR500(K).EQ.' ') GOTO 617
      J=J+1
      CHR10(J)=CHR500(K)
      GOTO 616
  617 CONTINUE
      IF (J.NE.0) GOTO 618
      WRITE(*,*) 'Too many blanks between M and mixture number.'
      GOTO 7776
  618 REWIND(tmp)
      WRITE(tmp,*) STR10
      REWIND(tmp)
      READ(tmp,*) Mixno
      GOTO 619
  619 Mixno=Mixno+1
      K=K-1
      IF (Mixno.EQ.OLDMIXNO) GOTO 621
      IF (Mixno-OLDMIXNO.EQ.1) goto 620
      WRITE(*,*) 'Mixture mumbers are out of sequence'
      GOTO 7776
  620 OLDMIXNO=Mixno
      MP=mixno
      IF (MP.LE.maxmix) GOTO 621
      WRITE(*,622) maxmix
  622 FORMAT (' Too many mixtures; up to ',I3,' are allowed.')
      GOTO 7776
  621 ML(MP)=ML(MP)+1
C
   62 CONTINUE
      K=K+1
      IF (CHR500(K).NE.' ') GOTO 62
   63 K=K+1
      IF (CHR500(K).EQ.' ') GOTO 63
C
C   Process "name"
C
      STR30='                              '
      J=1
   64 CONTINUE
      IF (J.LE.30) CHR30(J)=CHR500(K)
      K=K+1
      J=J+1
      IF (CHR500(K).NE.' ') GOTO 64
      TEXT(MS)=STR30
C
   65 K=K+1
      IF (CHR500(K).EQ.' ') GOTO 65
      STR30='                              '
      J=1
   66 CONTINUE
      IF (J.LE.30) CHR30(J)=CHR500(K)
      K=K+1
      J=J+1
      IF (CHR500(K).NE.' ') GOTO 66
C
C Formula Reading section
C
      IP=1
  101 CONTINUE
      IF (CHR30(IP).EQ.' ') GOTO 900
      IF (CHR30(IP).GT.'Z') GOTO 666
      IF (CHR30(IP).LT.'A') GOTO 666
      E8='        '
      C8(1)=CHR30(IP)
      IP=IP+1
      IF (CHR30(IP).GT.'z') GOTO 110
      IF (CHR30(IP).LT.'a') GOTO 110
      C8(2)=CHR30(IP)
      IP=IP+1
C
C process subscript
C
  110 CONTINUE
      S30='
      IK=0
  120 CONTINUE
      IF (CHR30(IP).GT.'9') GOTO 130
      IF (CHR30(IP).LT.'.') GOTO 130
      IF (CHR30(IP).EQ.CHAR(47)) GOTO 130
      IK=IK+1
      C30(IK)=CHR30(IP)
      IP=IP+1
      GOTO 120
C
C evaluate subscript
C
  130 CONTINUE
      IF(IK.GT.0) GOTO 140
      SUBVAL=1.0
      GOTO 150
  140 REWIND(tmp)
      WRITE(tmp,*) S30
      REWIND(tmp)
      READ(tmp,*) SUBVAL
  150 CONTINUE
C
C Load into table
```

```
C
      IF(L.eq.0) GOTO 211
  200 DO 210 I=1,L
      IF (E8.EQ.EL(I)) GOTO 220
  210 CONTINUE
  211 CONTINUE
      L=L+1
      IF (L.LE.maxele) GOTO 219
      WRITE(*,218) maxele
  218 FORMAT (' Too many elements; up to ',I3,' are allowed.')
      GOTO 7776
  219 EL(L)=E8
      I=L
  220 A(MS,I)=subval
      IF(IP.LT.30) GOTO 101
  900 CONTINUE
      GOTO 67
C
C  Error...
C
  666 CONTINUE
      WRITE(*,*) STR30
      WRITE(*,*) (' ',I=2,IP),'^'
      WRITE(*,*) 'Illegal formula format'
      GOTO 7777
C
   67 K=K+1
      IF (CHR500(K).EQ.' ') GOTO 67
C
C Now should be positioned on numeric fields; 1st blank out comments
C
      J=501
      DO 68 I=K,500
      IF (CHR500(I).NE.';') GOTO 68
      J=I
      GOTO 69
   68 CONTINUE
   69 CONTINUE
      IF (J.NE.501) GOTO 695
      WRITE(*,*) 'Data in each species'' line must end in a semicolon'
      GOTO 7776
C
  695 DO 70 I=J,500
   70 CHR500(I)=' '
   71 CONTINUE
C
      DO 72 I=1,7
      DO 72 J=1,6
   72 X(I,J)=0
      REWIND(tmp)
      WRITE(tmp,1) (CHR500(I),I=k,500)
      REWIND (unit=tmp)
      READ(tmp,*,END=80,ERR=78) ((X(I,J),I=1,7),J=1,6)
   80 CONTINUE
      K=0
      DO 95 J=1,6
      DO 95 I=1,3
   95 IF (X(I,J).NE.0) K=J
      IF (K.NE.0) GOTO 96
   78 WRITE(*,*) 'Thermo data not properly read for species "',
     1 TEXT(MS),'"'
      GOTO 7776
   96 NOM(MS)=K
      HF(MS)=X(1,1)
      S(MS)=X(2,1)
      DO 97,j=1,K
      IF (J.EQ.1) GOTO 971
      HOM(MS,J-1)=X(1,J)
      TOM(MS,J-1)=X(2,J)
  971 DO 97,i=1,5
   97 C(MS,J,I)=X(i+2,J)
  300 FORMAT (7(1X,F10.2))
  999 CONTINUE
      WRITE (*,992) MaxSp
  992 FORMAT (' Too many species; Up to ',I3,' are allowed.')
      GOTO 7776
C
C  File has been read; do summary report and bookkeeping
C
   16 MS=MS-1
C           ^ this should always be reached on an "EOF" signal
C
      IF (MP .EQ. 1) GO TO 146
      DO 132 M= 2, MP
      MF(M) = ML(M-1) + 1
      ML(M) = ML(M-1) + ML(M)
  132 CONTINUE
  146 M1 = ML(MP) + 1
      MF(1)=1
      IF(MS.EQ.(ML(MP)+MR)) GOTO 147
      WRITE(*,*) 'Accounting Error! ',MS,'><',ML(MP)+MR
      GOTO 7776
  147 CONTINUE
      IF (ML(1).GT.0) GOTO 148
      WRITE (*,*) 'The data set contains no gaseous species. At least on
     1e gas must be'
      WRITE (*,*) 'included to support the specified pressure. Revise da
     1ta set and rerun.'
```

```
      GOTO 7777
  148 CONTINUE
      DO 33 I = 1, MS
      XI(I)=0.
      DO 33 J=1, L
   33 AO(I,J) = A(I,J)
C
C  "Header" informatio..
C
      DO 9991 IOX=IOCRT,IOUT
      WRITE(IOX,*) TITLE
      WRITE(IOX,*)
      WRITE (IOX,250) MR,MP
      DO 9991 M=1,MP
      WRITE (IOX,251) (ML(M)-MF(M)+1),M
 9991 CONTINUE
  250 FORMAT (1X,I2,' = No. of Condensed Phases of Fixed Composition'
     1,/,1X,I2,' = No. of Mixtures')
  251 FORMAT (1X,I2,' = No. of Species in Mixture #',I2)
  998 CLOSE(IN)
      CLOSE (tmp)
      RETURN
C
C  Error encountered... abort this file...
C
 7776 WRITE(*,*) '  Offending data line is:'
      write(*,100) (CHR500(I),I=1,500)
  100 FORMAT (10X,'|',50A1,'|')
 7777 CONTINUE
      FILEFLAG=.TRUE.
      WRITE(*,*)
      WRITE(*,*)'Until corrected, SOLGAS cannot use the file: ',INFNAME
      CLOSE(IOUT)
      GOTO 998
      END
C
C
      SUBROUTINE DOF
C
C     A source file for SOLGAS, version 2.44
C
C     Checks for linear independence of chemical formulae in the SOLGAS
C     input data set; revises "elements" and formulae if needed.
C
      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL OPT12,ZERO,FILEFLAG
      CHARACTER*10 MATLNM
      CHARACTER*80 TITLE
      CHARACTER*64 INFNAME
      CHARACTER*8 EL
C
      COMMON DAKTF(99), AKTF(99), YTOT(20), PTOT, AKT(99), PI(20),
     $ YF(99), B(20,99), G(99), T, V, Y(99), KVAL1, KH(20),
     $ MA, MB, MO, NP, NO(99), NCODE, KQ, KS, PLUE, SLUE,
     $ APT(99), YM(99), OPT12, Rl, Rj
C
      COMMON /INPUT/ L,MP,MR,ML(20),MF(20),M1,MS,A(99,10),AO(99,10),
     $ XI(99),IOCRT,IOUT,IN,MIN,IIN(20),FILEFLAG
C
      COMMON /SPECIES/ MATLNM(99),TITLE,EL(10),INFNAME
C
      DIMENSION A1(99,10),CONV(10,10)
C
      maxele=10
      IOCRT=6
      IOUT =7
      DO 410 I=1,maxele
      DO 410 J=1,maxele
  410 CONV(I,J)=0.0
      DO 412 I=1,99
      DO 412 J=1,maxele
  412 A1(I,J)=0.0
      N_INDEP = 0
C **                                                     CHECK FOR ELEMENTS
      DO 420 NCMP = 1, MS
         NELE_CNT = 0
         DO 424 NELE = 1, L
            IF (AO(NCMP,NELE).EQ.0) GOTO 424
               NELE_CNT =NELE_CNT+1
               LAST_ELE =NELE
  424    CONTINUE
         IF ((NELE_CNT.EQ.1).AND.(A(NCMP,LAST_ELE).NE.0)) GOTO 426
            GOTO 428
  426    CONTINUE
         NCC =NCMP
         N_INDEP =N_INDEP+1
         DO 430 NELE2 = 1,L
  430       CONV(N_INDEP,NELE2) =A(NCC,NELE2)
         DO 432 IC =1 , MS
            A1(IC,N_INDEP) = A(IC,LAST_ELE)
            A(IC,LAST_ELE) = 0.0
  432    CONTINUE
  428 CONTINUE
  420 CONTINUE
C                                                 END of CHECK FOR ELEMENTS
      NCC = 0
  422 IF (NCC.GE.MS) GOTO 440
C                                                 BEGIN  NEXT COMPONENT
```

```
      IF (NCC.GE.MS) GOTO 450
      NCC = NCC+1
C                                               BEGIN Find non-zero species
      ZERO = .TRUE.
  476 CONTINUE
        DO 474 I =1 , L
           IF (ABS(A(NCC,I)).LE.1.E-6) GOTO 474
           ZERO = .FALSE.
           NZE =I
  474   CONTINUE
        IF (ZERO) NCC = NCC+1
        IF ((NCC.GT.MS).OR.(.NOT.ZERO)) GOTO 477
        GOTO 476
C                                               END of NON ZERO SPECIES
  477   IF (ZERO) GOTO 486
C                                               BEGIN ADD TO COMPONENT LIST
        N_INDEP =N_INDEP+1
        DO 484 NELE = 1,L
  484      CONV(N_INDEP,NELE) =A(NCC,NELE)
        A1(NCC,N_INDEP) =1.0
  486   CONTINUE
  450 CONTINUE
      NCS = NCC+1
C                                               END of NEXT COMPONENT
  488 IF (NCS.GT.MS) GOTO 490
        RATIO = A(NCS,NZE)/A(NCC,NZE)
        DO 492 J = 1,L
  492      A(NCS,J) = A(NCS,J)-RATIO*A(NCC,J)
        A1(NCS,N_INDEP) = RATIO
        NCS = NCS+1
        GOTO 488
  490 CONTINUE
      GOTO 422
  440 CONTINUE
C                         ****** END of Main process, BEGIN Results Output
C
      WRITE (IOCRT,550) L,N_INDEP
      WRITE (IOUT,550) L,N_INDEP
  550 FORMAT(1X,I2,' = No. of Elements',/
     x , 1X,I2,' = No. of Linearly Independent Components',/)
      IF (L.NE.N_INDEP) GOTO 560
C     WRITE(IOCRT,551)
C 551 FORMAT ('- No component change needed.',/,/)
      DO 557 I=1,99
         DO 557 J=1,maxele
  557       A(I,J)=A0(I,J)
      GOTO 510
  560 CONTINUE
      WRITE(IOCRT,*)' Converting to Linearly Independent Components per'
      WRITE(IOCRT,*)
      WRITE (IOCRT,561) (EL(I),I=1,L)
  561 FORMAT (' Comp.\Elem.    ',10A8)
      DO 520 I = 1 , N_INDEP
  520    WRITE (IOCRT,522)   I, (CONV(I,J),J=1,L)
  522 FORMAT (' Comp.#',I2,10F8.3)
      DO 559 I=1,99
         DO 559 J=1,maxele
  559       A(I,J)=A1(I,J)
      L = N_INDEP
      WRITE(IOCRT,*)
      WRITE(IOUT,*)
  510 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE CHANGEDATA
C
C     Change thermodynamic data routine
C
      IMPLICIT REAL*8 (A-H,O-Z)
      CHARACTER*8 EL
      CHARACTER*10 TEXT
      CHARACTER TITLE*80
      CHARACTER*64 INFNAME,OUTFNAME,PRMFNAME
      LOGICAL FILEFLAG
C
      COMMON /INPUT/ L,MP,MR,ML(20),MF(20),M1,MS,A(99,10),A0(99,10),
     $ XI(99),IOCRT,IOUT,IN,MIN,IIN(20),FILEFLAG
C
      COMMON /SPECIES/ TEXT(99),TITLE,EL(10),INFNAME
C
      COMMON /THERMO/ S0(99), HCT(99), HF(99), HFT(99), HOM(99,6),
     $ NOM(99), S(99), TOM(99,6), TSTART, OT0, OT1, C(99,6,5)
C
    5 WRITE (*,*)
      WRITE (*,*) ' Species present in the current data set:'
      WRITE (*,10) (I,TEXT(I),I=1,MS)
      WRITE(*,*)
      WRITE(*,*) ' If no (further) data changes are desired enter "0",'
      WRITE (*,*) ' otherwise enter the number of the species whose data
     1 are to be changed:'
      READ (*,*) ISPECIES
      IF (ISPECIES.EQ.0) RETURN
   10 FORMAT (5(1X,I2,1X,A10,2X))
      WRITE (*,*) '  1.  Heat of Formation at 298 K'
      WRITE (*,*) '  2.  Entropy at 298 K'
      WRITE (*,*) '  3.  Heat capacity constant'
   WRITE (*,*)
   WRITE (*,*) ' Select property you wish to change: '
   READ (*,*) KHOZE
   GO TO (30,40,50), KHOZE
30 WRITE (*,31) TEXT(ISPECIES), HF(ISPECIES)
31 FORMAT (/,' The heat of formation of ',A10,' is ',F12.0,' J/mol.'
  1/' Input a new value: ')
   READ (*,*) DIF_VAL
   HF(ISPECIES)=DIF_VAL
   GOTO 5
40 WRITE (*,41) TEXT(ISPECIES),S(ISPECIES)
41 FORMAT (/,' The entropy of ',A10,' is ',F10.3,' J/mol K.',/,
  1' Input a new value: ')
   READ (*,*) DIF_VAL
   S(ISPECIES)=DIF_VAL
   GOTO 5
50 WRITE(*,*) ' Only the constant parameter in the Cp equation may b
  1e modified.'
   WRITE (*,51) TEXT(ISPECIES),C(ISPECIES,1,1)
51 FORMAT(/,' Cp "A" value of ',A10,' is ',F10.3,' J/mol K.',/,
  1' Input a new value: ')
   READ (*,*) DIF_VAL
   C(ISPECIES,1,1)=DIF_VAL
   GOTO 5
   END
```

ATTACHMENT II

PROGRAM LISTING OF MASSAGE.PAS

```
program massage5a;

{$N+,E+}

{  Data post-processor for SOLGAS 2.44 output files.  Intended to convert
   SOLGAS output tables into an easily spreadsheet-importable file.

   Version  1 - 10/28/91 - by L.D.Trowbridge
   revision 2 - 11/12/91 - changed to detect "false start" results
   revision 3 - 12/27/91 - added ability to detect "exponent overflow"
                           (e.g. "-.10347-101", meaning "-.10347D-101"
   revision 4 -  4/16/92 - also will read Pascal-style output files
                           ( e.g. 1.234E+0001 instead of .1234D+01)
   revision 5 -  5/17/93 - handles (only) SOLGAS 2.43 & 2.44 output, which
                           includes gas volume in its output.
   revision 5a-  8/25/93 - handles (only) SOLGAS 2.44 output, which now
                           increases species name length to 10 (formerly 8).

   MASSAGE was written by L.D.Trowbridge, 5/17/93
   Martin Marietta Energy Systems, Inc.
   PO Box 2003, M.S. 7266
   Oak Ridge, TN 37831-7266
   ph (615) 576-2496
   Internet: <LDT@STC10.CTD.ORNL.GOV>

   this work was performed under contract to the U.S. Department of Energy
}

uses dos,crt;

type linevar =string[80];
     linetypevar = (temperature,pressure,volume,heat,for_data,
                    pas_data,unknown);

var  fin,fout                  : text;
     act,pp,ni,nf              : array [1..99] of double;
     name                      : array[1..99] of string[10];
     s1,s2,T,P,V,Preheat,
        Hrxn,Net_heat          : double;
     id,matl,sign,c1,c2,oops   : integer;
     found                     : boolean;
     l2                        : string[2];
     l4                        : string[4];
     l10                       : string[10];
     l15                       : string[15];
     nmin,nmout                : string[60];
     line                      : linevar;
     linetype                  : linetypevar;
     ln10                      : double;
```

```
procedure Terminate;
begin
    close(fin);
    close(fout);
    writeln('Finished.');
    Halt;
end;

procedure crash;
begin
  writeln;
  Writeln('translation error: code = ',oops,' in following line:');
  writeln(line);
  terminate;
end;

Procedure Output_data;
var m:integer;
begin
  FOR m:=1 to MATL DO
     WRITELN(fout,id:4,',',T:7:2,',',P:13,',',V:13,',',
             preheat:13,',',hrxn:13,',',net_heat:13
             ,',"',name[m]:10,'",',
             ni[m]:13,',',nf[m]:13,',',pp[m]:13{,',',act[m]:13});
end;

function translate(asc:string):double;
var mant,ex:double;
    cd:integer;
begin
{trim spaces off the ends}
   while asc[length(asc)]=' ' DO asc:=copy(asc,1,length(asc)-1);
   while pos(' ',asc)>0 DO
      begin
         asc:=copy(asc,pos(' ',asc)+1,99);
      end;
   s2:=1;
   if asc[1]='-' then s1:=-1 else s1:=+1;
   if s1=-1 then asc:=copy(asc,2,99);
   c1:=pos('+',asc);
   if c1=1 then asc:=copy(asc,2,99);             {scrap leading "+"}
   c1:=pos('+',asc);                              {find the sign in the exponent}
   if c1=0 THEN c1:=pos('-',asc);

   cd:=pos('D',asc);                              {there may or may not be a}
   if cd=0 then cd:=pos('E',asc);                 {D or E in the exponent   }
   if c1=0
      then ex:=0.0
      else
      begin
        if (asc[c1]='-') THEN s2:=-1;           {sign of the exponent}
        l4 := copy(asc,c1+1,4);
        val(l4,ex,oops); if oops>0 then crash;  {value of the exponent}
        if cd>0 then c1:=cd;
        asc:=copy(asc,1,c1-1);                  {trim the exponent}
      end;
   val(asc,mant,oops); if oops>0 then crash;    {calculate mantissa}
   translate:= s1*mant*exp(s2*ex*ln10);            {done...}
end;

procedure process_T;
begin
   c1:=pos('=',line)+1;
   c2:=pos('K',line)-1;
   c2:=c2-c1+1;
   L10:=copy (line,c1,c2);
   T:=translate(L10);
   if oops>0 then crash;
   found := true;
{} write('Working on Run #',ID:4,' ,T = ',T:8:2);
end;

procedure process_P;
begin
   c1:=pos('=',line)+1;
   c2:=pos('ar',line)-2;
   c2:=c2-c1+1;
   l15:=copy (line,c1,c2);
   P := Translate(l15);
   found := true;
{} write(', P = ',P:10:5);
end;

procedure process_V;
begin
   c1:=pos('=',line)+1;
   c2:=pos('liter',line)-1;
   c2:=c2-c1+1;
   l15:=copy (line,c1,c2);
   V := Translate(l15);
   found := true;
{} writeln(', V = ',V:6:1);
end;

Function Identification(A:linevar):linetypevar;
var id:linetypevar;
begin
    id := unknown;
    if copy(A,2,3)  = 'T =' THEN id:= temperature;
    if copy(A,2,3)  = 'P =' THEN id:= pressure;
    if copy(A,2,3)  = 'V =' THEN id:= volume;
    if ((copy(A,23,1) = 'D') and (copy(A,17,1) = '.')) THEN id:= for_data;
    if ((copy(A,21,1) = 'E') and (copy(A,16,1) = '.')) THEN id:= pas_data;
    if ((copy(A,32,2) = 'kJ') OR (copy(A,32,2) = 'KJ')) THEN id:= heat;
    identification:= ID;
end;

procedure Newline;
begin
     if EOF(fin) THEN Terminate;
     readln(fin,line);
end;

Procedure Process_FOR_Data;
begin
   matl:=matl+1;
   name[matl]:= copy(line,2,10);
   l15:=copy(line,15,12);ni[matl]:=translate(l15);
   l15:=copy(line,30,12);nf[matl]:=translate(l15);
   act[matl]:=-1.0e-99;
   pp[matl]:=-1.0e-99;
   if pos('D',copy(line,40,99)) >0 then
   begin
      l15:=copy(line,45,12);pp[matl]:=translate(l15);
      if pos('D',copy(line,56,99)) >0 then
      begin
         l15:=copy(line,60,12);
         act[matl]:=translate(l15);
      end;
   end;
end; {process_for_data}


Procedure Process_PAS_Data;
begin
   matl:=matl+1;
   name[matl]:= copy(line,2,10);                      {Name is 10-long}
   l15:=copy(line,14,13);ni[matl]:=translate(l15);
   l15:=copy(line,29,13);nf[matl]:=translate(l15);
   act[matl]:=-1.0e-99;
   pp[matl]:=-1.0e-99;
   if pos('E',copy(line,41,99)) >0 then
   begin
      l15:=copy(line,44,13);pp[matl]:=translate(l15);
      if pos('E',copy(line,55,99)) >0 then
      begin
         l15:=copy(line,59,13);
```

```
          act[matl]:=translate(l15);
        end;
     end;
end; {process_pas_data}

label  re_try;

begin  {MAIN}
     ln10:=ln(10);
     for c1:=1 to 4 DO writeln;
     writeln('       Data File reorganizer for SOLGAS ver. 2.44');
     writeln;
     writeln('  {Converts SOLGAS output to spreadsheet ".PRN" format)');
     writeln;
     write('SOLGAS output data file name?     ');
     readln(nmin);

{   add filter for bad names here}
     assign(fin,nmin);
     reset (fin);

     write('Name for spreadsheet-import file? ');
     readln(nmout);

{   add filter for bad names here}
     assign(fout,nmout);
     rewrite (fout);
     writeln(fout,'"ID","T(K)","P (bar)","V (l)","Pre-Heat (kJ)"',
                 ',"Rxn Heat (kJ)","Net Heat (kJ)","Species ID","',
                 'N(initial)","N(equilib)","P (bar)"');  {,"Act (bar)"');}

     ID:=0;
REPEAT
BEGIN
{search for "T-line"}
     ID:=succ(id);
     found:= false;
     repeat
        BEGIN
           Newline;
           if Identification(line)=temperature THEN Process_T;
        end;
        until found;

{search for "P-line"}
re_try: found:= false;      {re-entry point if data doesn't follow P}
     repeat
        BEGIN
           Newline;
           if Identification(line)=pressure THEN Process_P;
        end;
        until found;

{search for "V-line"}
     found:= false;
     repeat
        BEGIN
           Newline;
           if Identification(line)=volume THEN Process_V;
        end;
        until found;
        matl:=0;

{search for "Material-lines"}
     found:= false;
     repeat
        BEGIN
           Newline;
           linetype:= identification(line);

           if linetype=temperature then begin
              process_T;
              goto Re_try;
           end;

           if (linetype=pas_data) THEN process_pas_data;
           if (linetype=for_data) THEN process_for_data;

           if linetype=heat THEN found:= true;  {actually, end of list }
        end;
        until found;

{preheat}
    l15:=copy(line,20,11); Preheat:= translate(l15);
    readln(fin,line); l15:=copy(line,20,11); Hrxn:= translate(l15);
    readln(fin,line); l15:=copy(line,20,11); Net_Heat:= translate(l15);

    Output_data;

END {repeat of main loop}
UNTIL EOF(fin);
Terminate;
end.
```

ATTACHMENT III

PROGRAM LISTING OF ADIABAT.PAS

```
program ADIABAT;      {version 7, for SOLGAS v 2.44}       {LDT     5/19/93}
{$N+,E+}              {uses or emulates 80387}            {Turbo Pascal 6.0}
{$M $4000,0,0}        {...needed to reduce heap size, allowing use of "EXEC"}

{  "ADIABAT" for SOLGAS.  This program is intended to use standard SOLGAS
   input files, drive SOLGAS via DOS redirection and use of an internally
   generated macro file, read SOLGAS's output, alter the control macro,
   and re-run SOLGAS until some input value (usually the overall heat of
   reaction) converges to zero.

   This version is set up presently to...

      ...Run with SOLGAS.EXE (version 2.44)
      ...vary T or a single starting material's initial quantity.
      ...find an adiabatic solution, i.e. the "net heat = 0" point
      ...run at constant pressure or volume as specified by control file
            (Const P ==> enter +P (Bar); Const V ==> -V (liters))
      ...write brief output containing one line per final run, listing
            run #, T, P, V, heat, names and quantities of materials).

   See ADIABAT.DOC for more information.

   ADIABAT.PAS written by L.D.Trowbridge, 5-19-93
   Martin Marietta Energy Systems, Inc.
   PO Box 2003, M.S. 7266
   Oak Ridge, TN 37831-7266
   ph (615) 576-2496
   Internet: <LDT@STC10.CTD.ORNL.GOV>

   this work performed under contract to the U.S. Department of Energy }

uses dos,crt;

type linevar =string[80];
     linetypevar = (temperature,pressure,volume,heat,for_data,
                             pas_data,unknown);

const  SOLGAS_EXE='SOLGAS.EXE';
       cr = ^M^J;
       max_tries = 20;   {This algorithm IS a form of "twenty questions"...}
       criterion = 0.02; {can't be much smaller because of the number of   }
                         {significant digits retained by the SOLGAS output }

var  fin,mac,data_file,ctrl_file,results_file          : text;
     name_root,data_file_name, ctrl_file_name,results_file_name:string[80];
     temp,no_starting_matls     : integer;
     act,pp,ni,nf               : array [1..99] of double;
     name                       : array[1..99] of string[10];
     P_or_V,s1,s2,T,P,V,Preheat,
       Hrxn,Net_heat            : double;
     i,id,matl,sign,c1,c2,oops  : integer;
     found                      : boolean;
     l2                         : string[2];
     l4                         : string[4];
     l10                        : string[10];
     l15                        : string[15];
     nmin,nmout                 : string[60];
     line,line2                 : linevar;
     linetype                   : linetypevar;
     ln10                       : double;

{Interpolation and control variables; some aren't used as yet: }
     divergence                 : boolean;
     T_test,P_test,V_test,H_test: array[1..2] of real;
     Q_test                     : array[1..2,1..7] of real;
     PP_test                    : array[1..2,1..99] of real;
     id_of_matl                 : array[1..7] of integer;
     active,inactive,it_cnt,ctrl_var     : integer;

procedure RUN_DOS(cmd:string);
begin
    swapvectors;
    exec('command.com','/c '+cmd);
    swapvectors;
    if DosError<>0 then
       writeln('Dos Error #:',Doserror,' running "',cmd,'"');
end;

procedure Terminate;
begin
    writeln('Finished.');
    RUN_DOS('del $$temp.*');
    close(results_file);
    close(ctrl_file);
    Halt;
end;

procedure crash;
begin
  writeln;
  Writeln('translation error: code = ',oops,' in following line:');
  writeln(line);
  terminate;
end;

procedure Output_macro_file(b:integer);
var m:integer;
```

```
      no_starting_matls:=0;
      repeat readln(data_file,line) UNTIL
         ((copy(line,1,2)='T=')
                OR
         (copy(line,1,3)='T,=')
                OR
             EOF(data_file));
      IF EOF(data_file) THEN
      begin
         writeln;
         writeln('No starting material temperature line found in "',
                  data_file_name,'".');
         close(data_file);
         halt;
      end;
      WHILE NOT EOF(data_file) DO                         {Read SOLGAS data  }
      begin                                               {file to determine }
         readln(data_file,line);                          {the IDs of the    }
         inc(i);                                          {starting materials}
         if copy(line,1,1)='*' THEN
         begin
            inc(no_starting_matls);
            id_of_matl[no_starting_matls]:=i;
         end;
      end;
      close(data_file);
   { end of input file-dependent section}

      IF PARAMCOUNT=0 THEN
      begin
         write('       Control file name?             ');
         readln(ctrl_file_name);
      end ELSE ctrl_file_name:=name_root+'.DRV';
      assign(ctrl_file,ctrl_file_name);
      reset (ctrl_file);

      IF PARAMCOUNT=0 THEN
      begin
         write('       Results file name?             ');
         readln(results_file_name);
      end ELSE results_file_name:=name_root+'.PRN';

      assign(mac,'$$temp.out'); rewrite(mac);
         writeln(mac,'dummy line'); close(mac);

      assign(results_file,results_file_name);
      rewrite (results_file);
      brief_header;   { or Header; }
      ID:=0;
```

```
   REPEAT
   BEGIN
      divergence:= false;  it_cnt:=0;  id:=succ(id);
      read(ctrl_file,ctrl_var,T_test[1],P_or_V);
      H_Test[1]:=-9.999; H_Test[2]:=-9.999;
      for i:= 1 to no_starting_matls do read(ctrl_file,Q_test[1,i]);
      readln(ctrl_file);

      t_test[2]:=t_test[1];
      for i:=1 to no_starting_matls DO q_test[2,i]:=q_test[1,i];
      if ctrl_var=0
         THEN t_test[2]:=t_test[2]*1.10
         else q_test[2,ctrl_var]:=q_test[2,ctrl_var]*1.10;
      RUN_SOLGAS(1);
      active:=2;
      repeat
      begin
         RUN_SOLGAS(active);
         inactive:=2; if active=2 then inactive:=1;
         if (abs(h_test[active])>Criterion) then
            if ctrl_var=0
               then Interpolate(H_test[1],H_test[2],T_test[1],T_test[2])
               else Interpolate(H_test[1],H_test[2],
                  q_test[1,ctrl_var],q_test[2,ctrl_var]);
      end;
      until ((it_cnt>max_tries) OR divergence OR (ABS(H_test[active])<=Criterion));
      write('Run #',ID:5);
      if not divergence then
        begin
           write(' T = ',T_Test[active]:7:2,' H = ',H_Test[active]:6:3,', X = (');
           for i:=1 to no_starting_matls do
              write(Q_Test[active,i]:6:3,',');
           writeln()');   {backspace...}
           if (it_cnt>max_tries) then
           begin           {non-converged data, but print out both anyway}
              active:=1; brief_output_data;
              active:=2; brief_output_data;
              writeln(' - too many iterations; results saved, but are suspect');
           end
           else  {AOK}
              brief_Output_data;
            { OR use:  "Output_data;"  for full spreadsheet-importable output}
        end
        else {divergence}
           writeln(' diverged - results discarded');
   END {repeat of main loop}
   UNTIL EOF(ctrl_file);
   Terminate;
END.
```

```
begin
    assign(mac,'$$temp.mac');
    rewrite(mac);
    writeln(mac,data_file_name,cr,'$$temp.out',cr,T_Test[b]:8:3);
    writeln(mac,'7',cr,P_or_V:15:5);
    writeln(mac,'1',cr,'1');
    for m:=1 to No_starting_matls do writeln(mac,'1');
    for m:=1 to No_starting_matls do
        if Q_Test[b,m]<=1e-6 then writeln(mac,'1E-6')
                     else writeln(mac,Q_test[b,m]:10:6);
    writeln(mac,cr,'10',cr,'9');
    close(mac);
end;

procedure interpolate(VAR H1,H2,C1,C2:real);    {Newton's method, I think.  }
var c3:double;                                  {finds new Cn that makes Hn }
begin                                           {zero, storing Cn in poorer }
    if abs(h2-h1)<1e-6 then                     {of C1 or C2.               }
      begin
        writeln('Convergence failure...');
        divergence:=true;
        exit;
      end;
    c3:=-h1*(c2-c1)/(h2-h1)+C1;
    if abs(c3-c1) < abs(c3-c2)
      then begin  active:=2;  c2:=c3;  end
      else begin  active:=1;  c1:=c3;  end;
end;

procedure brief_header;
var m:integer;
begin
   write(results_file,'"ID","T(K)","P (bar)","V (l)","Net Heat (kJ)"');
   for m:= 1 to no_starting_matls DO
      write(results_file,',"Name","Qi(',m:1,')"');
   writeln(results_file);
end;

Procedure Brief_Output_data;

var m:integer;
    warning: string[15];
begin
   if it_cnt>max_tries then warning:=',"UNCONVERGED"' else warning:=' ';
   WRITE(results_file,id:4,',',T_test[active]:7:2,',',P:12,',',V:12,
   ',',H_test[active]:13);
   FOR m:=1 to no_starting_matls DO
      WRITE(results_file,',"',name[id_of_matl[m]]:10,'",',Q_test[active,m]:13);
   writeln(results_file,warning);
end;

(*
procedure header;
begin
   writeln(results_file,'"ID","T(K)","P (Bar)","V (l)","Pre-Heat (kJ)"',
                 ',"Rxn Heat (kJ)","Net Heat (kJ)","Species ID",""',
                 'N(initial)","N(equilib)","P (Bar)","Act (Bar)"');
end;

Procedure Output_data;
var m:integer;
beg
           to MATL DO
           N(results_file,id:4,',',T:7:2,',',P:13,',',V:12,',',
             preheat:13,',',hrxn:13,',',net_heat:13
             ,',"',name[m]:10,'",',
             ni[m]:13,',',nf[m]:13,',',pp[m]:13,',',act[m]:13);
end;
*)

function translate(asc:string):double;      {Translates Pascal or Fortran  }
var mant,ex:double;                         {numeric string to a number    }
    cd:integer;
begin

{trim spaces off the ends}
   while asc[length(asc)]=' ' DO asc:=copy(asc,1,length(asc)-1);
   while pos(' ',asc)>0 DO
      begin
         asc:=copy(asc,pos(' ',asc)+1,99);
      end;
   if pos('***',asc)>0 THEN
   BEGIN
       Writeln('translation error in "',asc,'"');
       divergence:=true;
       exit;
   END;

   s2:=1;
   if asc[1]='-' then s1:=-1 else s1:=+1;
   if s1=-1 then asc:=copy(asc,2,99);
   c1:=pos('+',asc);
   if c1=1 then asc:=copy(asc,2,99);           {scrap leading "+"}
   c1:=pos('+',asc);                           {find the sign in the exponent}
   if c1=0 THEN c1:=pos('-',asc);

   cd:=pos('D',asc);                           {there may or may not be a}
   if cd=0 then cd:=pos('E',asc);              {D or E in the exponent   }
```

```
    if c1=0
       then ex:=0.0
       else
       begin
         if (asc[c1]='-') THEN s2:=-1;          {sign of the exponent}
         l4 := copy(asc,c1+1,4);
         val(l4,ex,oops); if oops>0 then crash;  {value of the exponent}
         if cd>0 then c1:=cd;
         asc:=copy(asc,1,c1-1);                  {trim the exponent}
       end;
    val(asc,mant,oops); if oops>0 then crash;    {calculate mantissa}
    translate:= s1*mant*exp(s2*ex*ln10);            {done...}
end;

procedure process_T;                   {Temperature Line in SOLGAS Output}
begin
   c1:=pos('=',line)+1;
   c2:=pos('K',line)-1;
   c2:=c2-c1+1;
   L10:=copy (line,c1,c2);
   T:=translate(L10);
   if oops>0 then crash;
   found := true;
end;

procedure process_V;                   {Volume Line in SOLGAS Output}
begin
   c1:=pos('=',line)+1;
   c2:=pos('liter',line)-1;
   c2:=c2-c1+1;
   l15:=copy (line,c1,c2);
   V:=translate(L15);
   if oops>0 then crash;
   found := true;
end;

procedure process_P;                   {Pressure Line in SOLGAS Output}
begin
   c1:=pos('=',line)+1;
   c2:=pos('bar',line)-1;
   c2:=c2-c1+1;
   l15:=copy (line,c1,c2);
   P := Translate(l15);
   found := true;
end;

Function Identification(A:linevar):linetypevar;
var tp:linetypevar;
begin
```

```
    tp := unknown;
    if copy(A,2,3)  = 'T =' THEN tp:= temperature;
    if copy(A,2,3)  = 'P =' THEN tp:= pressure;
    if copy(A,2,3)  = 'V =' THEN tp:= volume;
    if ((copy(A,23,1) = 'D') and (copy(A,17,1) = '.')) THEN tp:= for_data;
    if ((copy(A,21,1) = 'E') and (copy(A,16,1) = '.')) THEN tp:= pas_date;
    if copy(A,32,2) = 'kJ' THEN tp:= heat;
    identification:= tp;
end;

procedure Newline;
begin if EOF(fin) THEN Terminate; readln(fin,line); end;

Procedure Process_FOR_Data;
begin
   matl:=matl+1;
   name[matl]:= copy(line,2,10);
   l15:=copy(line,15,12);ni[matl]:=translate(l15);
   l15:=copy(line,30,12);nf[matl]:=translate(l15);
   act[matl]:=-1.0e-99;
   pp[matl]:=-1.0e-99;
   if pos('D',copy(line,40,99)) >0 then
   begin
      l15:=copy(line,45,12);pp[matl]:=translate(l15);
      if pos('D',copy(line,56,99)) >0 then
      begin
         l15:=copy(line,60,12);
         act[matl]:=translate(l15);
      end;
   end;
end; {process_for_data}

Procedure Process_PAS_Data;
begin
   matl:=matl+1;
   name[matl]:= copy(line,2,10);                  {now uses 10-long name!}
   l15:=copy(line,14,13);ni[matl]:=translate(l15);
   l15:=copy(line,29,13);nf[matl]:=translate(l15);
   act[matl]:=-1.0e-99;
   pp[matl]:=-1.0e-99;
   if pos('E',copy(line,41,99)) >0 then
   begin
      l15:=copy(line,44,13);pp[matl]:=translate(l15);
      if pos('E',copy(line,55,99)) >0 then
      begin
         l15:=copy(line,59,13);
         act[matl]:=translate(l15);
      end;
   end;
```

```
end; {process_pas_data}

procedure Interpret_solgas_output_table(n:integer);
label  re_try;                               {Identify and process a single }
var i:integer;                               {SOLGAS output table}
begin

{search for "T-line"}
     found:= false;
     repeat
        BEGIN
           Newline;
           if Identification(line)=temperature THEN Process_T;
        end;
        until found;

{search for "P-line"}
re_try: found:= false;      {re-entry point if data doesn't follow P}
     repeat
        BEGIN
           Newline;
           if Identification(line)=pressure THEN Process_P;
        end;
        until found;

{search for "V-line"}
     found:= false;
     repeat
        BEGIN
           Newline;
           if Identification(line)=volume THEN Process_V;
        end;
        until found;
     matl:=0;
{search for "Material-lines"}
     found:= false;
     repeat
        BEGIN
           Newline;
           linetype:= identification(line);
           if linetype=temperature then begin
              process_T;
              goto Re_try;
           end;
           if (linetype=pas_data) THEN process_pas_data;
           if (linetype=for_data) THEN process_for_data;
           if linetype=heat THEN found:= true;  {actually, end of list, in thi
        end;
        until found;
```

```
{preheat}
   l15:=copy(line,20,11); Preheat:= translate(l15);
   readln(fin,line); l15:=copy(line,20,11); Hrxn:= translate(l15);
   readln(fin,line); l15:=copy(line,20,11); Net_Heat:= translate(l15);

{save run results in internal feedback arrays}
   T_test[n]:=T; H_test[n]:=net_heat;
   for i:=1 to No_starting_matls do Q_test[n,i]:= ni[id_of_matl[i]];
end;

procedure RUN_SOLGAS(k:integer);
begin
   output_macro_file(k);
   it_cnt:=succ(it_cnt);
   RUN_DOS('del $$temp.out');
   RUN_DOS(SOLGAS_EXE+' < $$temp.mac > nul');      {runs from macro; sends }
                                                   { screen output to limbo}
   assign(fin,'$$temp.out');
   reset(fin);
   interpret_SOLGAS_output_table(K);
   close(fin);
{ writeln('Run #',ID:5,' T1 = ',T_Test[1]:7:2,' T2 = ',T_Test[2]:7:2,
                          ' H1 = ',H_Test[1]:8:3,' H2 = ',H_Test[2]:8:3); }
end;

begin  {MAIN}
     assign(input,'');  reset (input);        {These two lines allow DOS}
     assign(output,''); rewrite (output);     {redirection on this file }

     ln10:=ln(10);
     for c1:=1 to 4 DO writeln;
     writeln(
'Adiabatic Driver for SOLGAS v. 2.44 -- written in Turbo Pascal v 6.0');
     writeln(
'    version 7.0                       by L.D.Trowbridge --  5/18/93');
     writeln;
     name_root:=ParamStr(1);
     IF PARAMCOUNT=0 THEN
     begin
        write('SOLGAS thermodynamics data file name? ');
        readln(data_file_name);
     end ELSE data_file_name:=name_root+'.DAT';
     assign(data_file,data_file_name); reset (data_file);

{following section is the only one depending on the format of the }
{SOLGAS input data file }

     i:=0;
```

DISTRIBUTION

Los Alamos National Laboratory

D. E. Peterson

Lawrence Berkeley Laboratory

L. Brewer

Oak Ridge K-25 Site

D. P. Armstrong
E. J. Barber
G. B. Boroughs
W. D. Bostick
D. H. Bunch
D. A. Harkins
R. L. Higgins
R. J. Jarabek
J. M. Leitnaker (10)
D. L. Mason
J. H. Pashley
J. F. Preston
A. S. Quist (3)*
R. L. Ritter
J. A. Stockdale
L. D. Trowbridge
App. Tech. Library
K-25 Plant Records

Oak Ridge National Laboratory

J. Bentley
D. N. Braski
J. Brynestad
T. B. Lindemer

Paducah Gaseous Diffusion Plant

P. G. Brown
P. B. Kreitz
M. G. Otey
S. F. Seltzer
P. D. Wooldridge
Technical Library

Penn. State University

K. E. Spear

Portsmouth Gaseous Diffusion Plant

D. M. Manuta
W. D. Netzer
A. J. Saraceno
Technical Library

University of Kansas

P. W. Gilles

*2 copies intended for OSTI