LBL-22663 c.2

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

APPLIED SCIENCE DIVISION

RECEIVED
LAWRENCE
BERKELEY LABORATORY
MAR 2 1987
LIBRARY AND
DOCUMENTS SECTION

PROCESS STUDIES FOR A NEW METHOD OF REMOVING H_2S FROM INDUSTRIAL GAS STREAMS

D.W. Neumann* and S. Lynn
(*Ph.D. Thesis)

July 1986

TWO-WEEK LOAN COPY
This is a Library Circulating Copy
which may be borrowed for two weeks.

APPLIED SCIENCE
DIVISION

LBL-22663 c.2

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

DISCLAIMER

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

LBL-22663

PROCESS STUDIES FOR A NEW METHOD OF REMOVING H_2S FROM INDUSTRIAL GAS STREAMS

by

Dan W. Neumann and Scott Lynn

July 1986

Applied Science Division
Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

This work was supported by the Morgantown Energy Technology Center, Assistant Secretary for Fossil Energy, Office of Coal Utilization, Advanced Research and Technology Development, Division of Surface Coal Gasification through the U.S. Department of Energy under contract number DE-AC03-76SF00098.

Process Studies for a New Method of Removing H_2S from Industrial Gas Streams

Dan Neumann

Abstract

A process for the removal of hydrogen sulfide from coal-derived gas streams has been developed. The basis for the process is the absorption of H_2S into a polar organic solvent where it is reacted with dissolved sulfur dioxide to form elemental sulfur. After sulfur is crystallized from solution, the solvent is stripped to remove dissolved gases and water formed by the reaction. The SO_2 is generated by burning a portion of the sulfur in a furnace where the heat of combustion is used to generate high pressure steam. The SO_2 is absorbed into part of the lean solvent to form the solution necessary for the first step.

The kinetics of the reaction between H_2S and SO_2 dissolved in mixtures of N,N-Dimethylaniline (DMA)/ Diethylene Glycol Monomethyl Ether and DMA/Triethylene Glycol Dimethyl Ether was studied by following the temperature rise in an adiabatic calorimeter. This irreversible reaction was found to be first-order in both H_2S and SO_2, with an approximate heat of reaction of 28 kcal/mole of SO_2. The sole products of the reaction appear to be elemental sulfur and water. The presence of DMA increases the value of the second-order rate constant by an order of magnitude over that obtained in the glycol ethers alone. Addition of other tertiary aromatic amines enhances the observed kinetics; heterocyclic amines (e.g. pyridine derivatives) have been found to be 10 to 100 times more effective as catalysts when compared to DMA.

The reaction provides a foundation for the development of a process for removing H_2S from a fuel gas made by gasifying high-sulfur coal. A computer simulation was written to permit rapid analysis of potential process alternatives. Calculations in the simulation were performed in unit sequential fashion with

convergence accelerated using Wegstein's method. Modifications were made to standard engineering unit models to account for the presence of the reaction between H_2S and SO_2. A flowsheet and detailed heat and material balances for a process for the reduction of the H_2S content of a gasified coal stream from 60,000 to 1 ppm are presented.

Acknowledgements

Berkeley is a wonderful, often bizarre place, with many opportunities for learning, both within and outside of the University of California. This atmosphere is created by many factors, the most important of which is people. My stay here has been enhanced by a number of interesting and helpful individuals and organizations.

First and foremost, I wish to thank Professor Scott Lynn, who has guided my thesis work with intelligence, patience, and good humor.

Thanks also to my lab mates over the years, particularly Tom Colson, who performed some of the kinetics experiments, and Steve Sciamanna, a good friend who has taught me much engineering.

Bravo and a tip of the hat to Kay Ekman, Ellen Borrowman, Ruth Cooper, and Gege Muse of the Graduate Office, who facilitate dealing with the University and are friendly even in difficult times.

I am very grateful for the support, entertainment, and computer hardware (especially a fast 287 processor) provided by Len Ott and CompuPro.

A special thanks to R.S. Zarbaugh and I.M. Jenkins, who, though conspicuously non-existent, provided endless entertainment.

Thanks to the Shell Companies Foundation and to Texaco for their generous contributions to the livelihood of the author.

This work was supported in part by the Morgantown Energy Technology Center, Assistant Secretary for Fossil Energy, Office of Coal Utilization, Advanced Research and Technology Development, Division of Surface Coal Gasification, through the Department of Energy under Contract DE-AC03-76SF00098.

TABLE OF CONTENTS

	Abstract	1
	Acknowledgements	i
I.	**Introduction**	1
	A. Need for H_2S Removal Technology	1
	B. H_2S Removal Technology	1
	C. University of California Sulfur Recovery Process (UCBSRP)	3
	D. Scope of Thesis Work	4
II.	**Kinetics of the Reaction Between H_2S and SO_2 in Organic Solvents**	5
	A. Background	5
	B. Experimental Methods	5
	C. Methods of Analysis	7
	1. *Acid-Base Experiments*	7
	2. *Kinetics*	9
	3. *Heat of Reaction*	11
	D. Results and Discussion	12
	1. *Results for the Reaction of H_2S and SO_2 in Triglyme/DMA/H_2O*	12
	2. *Results for the Reaction in DMA/DGM/Water*	14
	3. *Results for the Reaction in DMA/Triglyme/Methanol*	16
	4. *Heat of Reaction*	20
	5. *Summary of DMA Results*	20
	E. Catalyst Survey	21
	1. *Catalyst Structure*	21
	2. *Parametric Studies of the Reaction using 3PC in DGM and Triglyme*	24
	3. *Water Effect*	27
	F. Selection of Catalyst	27

Table of Contents (Continued)

III. Process Development 31

A. Background 31

B. Simulation 31

1. *Main Program* 31

2. *Unit Operations* 34

C. Process Description 38

1. *Process Specification* 38

2. *Processing Steps* 38

3. *Summary* 46

References 52

List of Variables 55

Appendix 56

A-1. Simulation Programs 56

A. Main Program - BERKSULF.F77 56

B. Tray Absorber/Stripper/Distillation Routine 58

C. Heat Exchanger Subroutine 61

D. Thermodynamics Subroutine 63

A-2. Simulation Usage 64

A-3. Simulation Code 67

A-4. Simulation Data Entry Programs 228

I. Introduction

A. Need for H_2S Removal Technology

Hydrogen sulfide, found in a number of industrial process streams, poses a significant toxicity and odor problem. Recent concern for the environment has induced industry to intensify efforts to achieve high levels of removal of H_2S from gaseous waste streams. In many cases essentially complete removal of sulfurous compounds is necessary. Because the requisite processes are costly in terms of capital expenditures and energy usage, improvements in technology are the subject of ongoing research.

Investigations are currently being conducted into the removal of hydrogen sulfide from natural gas, refinery gas and gasified coal. The extent to which these streams are contaminated and the amount of H_2S that must be removed varies with the application. For instance, when coal is gasified most of the organic sulfur is converted to H_2S that ranges in concentration from several ppm to a few percent depending on the source of the coal. Before the gas can be used, the H_2S content will generally need to be reduced to the ppm level, although reduction to the ppb range may be required in some applications.

B. H_2S Removal Technology

Since its development in the late 19th century and through subsequent refinements, the principle method of H_2S removal has been the Claus process. This operation involves the following steps:

1. An absorption-stripping cycle applied to the H_2S-laden gas stream, typically using ethanolamines as absorbents.

2. Oxidation of 1/3 of the H_2S to SO_2.

3. Heterogeneous catalytic reaction between the H_2S and SO_2 at high temperature to produce sulfur and water -- i.e.

$$SO_2 + H_2S \rightarrow 3/x\ S_x + 2H_2O$$

4. Condensation of the sulfur by indirect heat exchange or countercurrent scrubbing with liquid sulfur.

The Claus process is an economic sulfur producer that is capable of handling large quantities of contaminated gas. The major drawback is that the process is equilibrium-limited to 95-97% conversion in 2 to 4 stages at the temperatures required to prevent condensation of sulfur in the reactors. This would result in a stack gas that contained, after combustion of the H_2S, more than 3000 ppm SO_2 (Ferguson, 1975, pg. 1). In general, further processing must be provided to reduce the concentration of sulfurous compounds in the effluent to environmentally acceptable levels.

A great number and variety of processes have been developed to treat the exit gas from a Claus plant. Generally, in the first step of any "tail gas" process, the sulfur in the gas is converted to a single form, either by reduction of excess SO_2 to H_2S or by oxidation of H_2S to SO_2. The sulfurous material is then concentrated and recycled to the beginning of the Claus plant or converted to a product which can be sold or disposed. A detailed discussion of these "tail gas" clean-up units is beyond the scope of the present work. The interested reader is referred to several excellent reviews (for example Ferguson (1975); Kohl and Riesenfeld (1979)) for descriptions of these processes. In order to illustrate the general approach taken and some of the problems encountered in recovering residual hydrogen sulfide, an example of a commercially successful process, the Beavon Sulfur Removal Process, is given below.

In the Beavon Sulfur Removal Process, SO_2 in the Claus plant exit gas is first reduced to H_2S over a cobalt-molybdenum catalyst. This stream is then fed to a Stretford absorber where the H_2S is absorbed into an aqueous sodium bicarbonate solution as bisulfide. The bisulfide is oxidized by vanadium (as vanadate ion) to sulfur, which is removed by filtration. Vanadium is reoxidized by anthraquinone

disulfonic acid (ADA), which is regenerated by air or oxygen.

The Beavon/Stretford process has excellent sulfur removal capability; exit concentrations of H_2S in the ppm range can be achieved. However, the consumption of valuable fuel in the reduction step and the large circulation rates needed due to the low absorption capacity of the solution (~ 1 g sulfur/lit) reduce the attractiveness of the process. Furthermore, the sulfur formed by the reaction is of low quality and often difficult to remove from solution. Because of the high pumping requirements and the net energy consumption, the Beavon/Stretford process is limited to the treatment of gases containing no more than 10 tons/day of sulfur (Kohl and Riesenfeld, 1979).

C. University of California Berkeley Sulfur Recovery Process (UCBSRP)

The process discussed in this thesis and currently under development in the Lawrence Berkeley Laboratory at the University of California is an attempt to overcome some of the limitations and problems of current technology. In the UCBSRP hydrogen sulfide is absorbed into a polar organic solvent where it is reacted with dissolved sulfur dioxide to form sulfur. After sulfur is recovered from the solvent by crystallization, the solvent is stripped to remove dissolved gases and water formed in the reaction. SO_2 is generated by burning a portion of the sulfur in a furnace, where the heat of combustion is used to generate high pressure steam. The SO_2 is absorbed into the lean solvent to form the solution necessary for the first step.

Conceptually the UCBSRP is analogous to the Claus process carried out in the liquid phase. However, unlike the gas-phase Claus reaction, the reaction between H_2S and SO_2 dissolved in organic liquids can be performed at temperatures below 150°C where the reaction is irreversible and goes essentially to completion. With judicious use of this added driving force, the hydrogen sulfide level in the outlet gas can be reduced to less than 1 ppm without the addition of a costly tail gas

unit.

Several potential advantages of this process over conventional approaches are:

1. Reduced capital and energy costs.

2. Efficient removal of H_2S -- reduce effluent to less than 1 ppm hydrogen sulfide.

3. Applicability to a wide range of gas streams, varying both in H_2S content and in conditions (e.g. composition, pressure, etc.).

D. Scope of Thesis Work

The present study is part of the ongoing development of the UCBSRP. One aim is elucidation of the kinetics of the reaction between hydrogen sulfide and sulfur dioxide in organic solvents. A variety of compounds have been investigated as potential solvents and catalysts for the reaction between hydrogen sulfide and sulfur dioxide. The reaction in these systems and those features of the kinetics important to the process are discussed.

In order to better evaluate the process and to aid in its development, a computer simulation was created. The simulation provides a means of identifying key process constraints and simplifies the task of performing calculations on a variety of flow configurations. Details of the computational algorithms found in the simulation and directions for its use are given.

Finally a flowsheet for the removal of H_2S from gasified coal is presented. This demonstrates several novel facets of the process as well as indicating the feasibility of applying it to the desulfurization of an industrially important gas stream.

II. Kinetics of the Reaction Between H_2S and SO_2 in Organic Solvents

A. Background

When the reaction between H_2S and SO_2 is carried out in organic liquids at temperatures below 150°C, it is irreversible and goes essentially to completion. The equilibrium limitation encountered in the gas-phase reaction at temperatures above the dew point of sulfur is thereby overcome. This shift in the equilibrium state is caused both by the lower temperature and by the reduced activity of the reaction products when in solution. In many organic solvents, such as triethylene glycol dimethyl ether (Triglyme) and diethylene glycol methyl ether (DGM), the reaction is impractically slow. Moreover, at room temperature the sulfur formed is too finely divided to be readily separated. Urban (1961) found that the presence of N,N-dimethyl aniline (DMA) increased the crystal size of the precipitated sulfur. Furthermore, DMA and other tertiary amines accelerate the reaction to the extent that 99+% removal of H_2S is possible with careful selection of the solvent mixture.

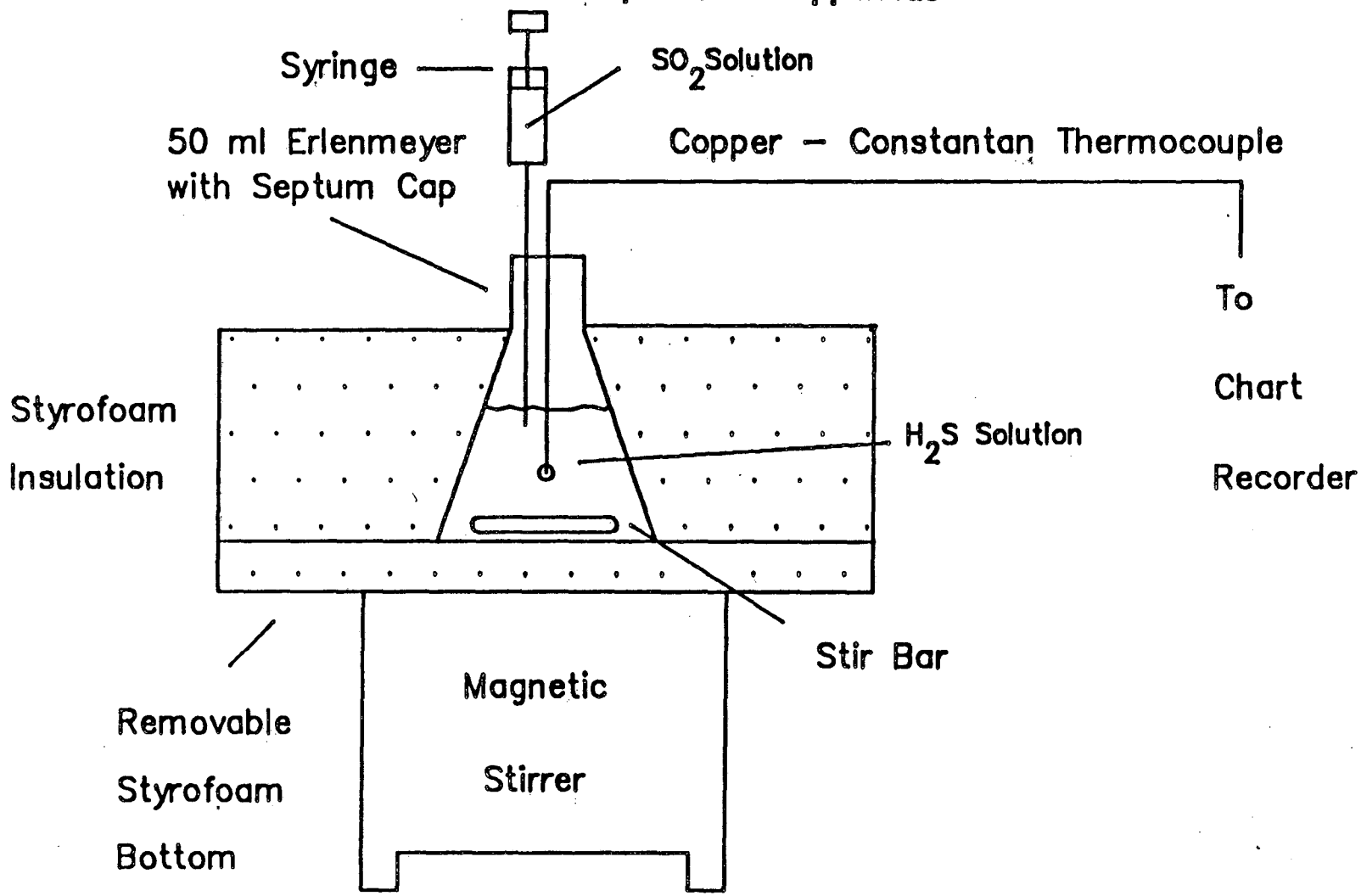
The present work was undertaken to study the reaction between H_2S and SO_2 in mixtures of tertiary amines with Triglyme and with DGM. Experiments performed by monitoring the temperature rise of this exothermic reaction in an adiabatic calorimeter show the effects of various solvent compositions on the kinetics of the reaction. A knowledge of these effects can be applied to the selection of suitable solvents for processing H_2S-laden gas streams.

B. Experimental Methods

All rate measurements were made in a 50-ml Erlenmeyer flask that contained a magnetic stir bar and was sealed by a septum cap (see Fig. II-1). Insulation was provided by a styrofoam block with a hole drilled out for the reactor. Reaction progress was monitored by recording the temperature rise as indicated by the change in the millivolt-level potential produced by a bare, type T thermocouple

Figure II-1

Experimental Apparatus



connected to a chart recorder. A measured quantity of a solution of one reactant was first placed in the reaction vessel. The reaction was initiated when a measured quantity of a solution of the second reactant at the same temperature was injected quickly by syringe into the stirred vessel.

The range of initial reactant concentrations was limited to 0.05 to 0.25 M SO_2 and 0.1 to 0.5 M H_2S. These compositions were high enough to produce sufficient reaction to permit accurate measurement of the temperature rise without exceeding an arbitrarily selected maximum temperature increase of 5 ^{o}C. Since the temperature coefficient of the reaction rate constant was found to be relatively low (see below), the rate constants obtained from these adiabatic experiments are very nearly equal to those for the reaction carried out under isothermal conditions - i.e. at the average temperature of a run.

C. Methods of Analysis

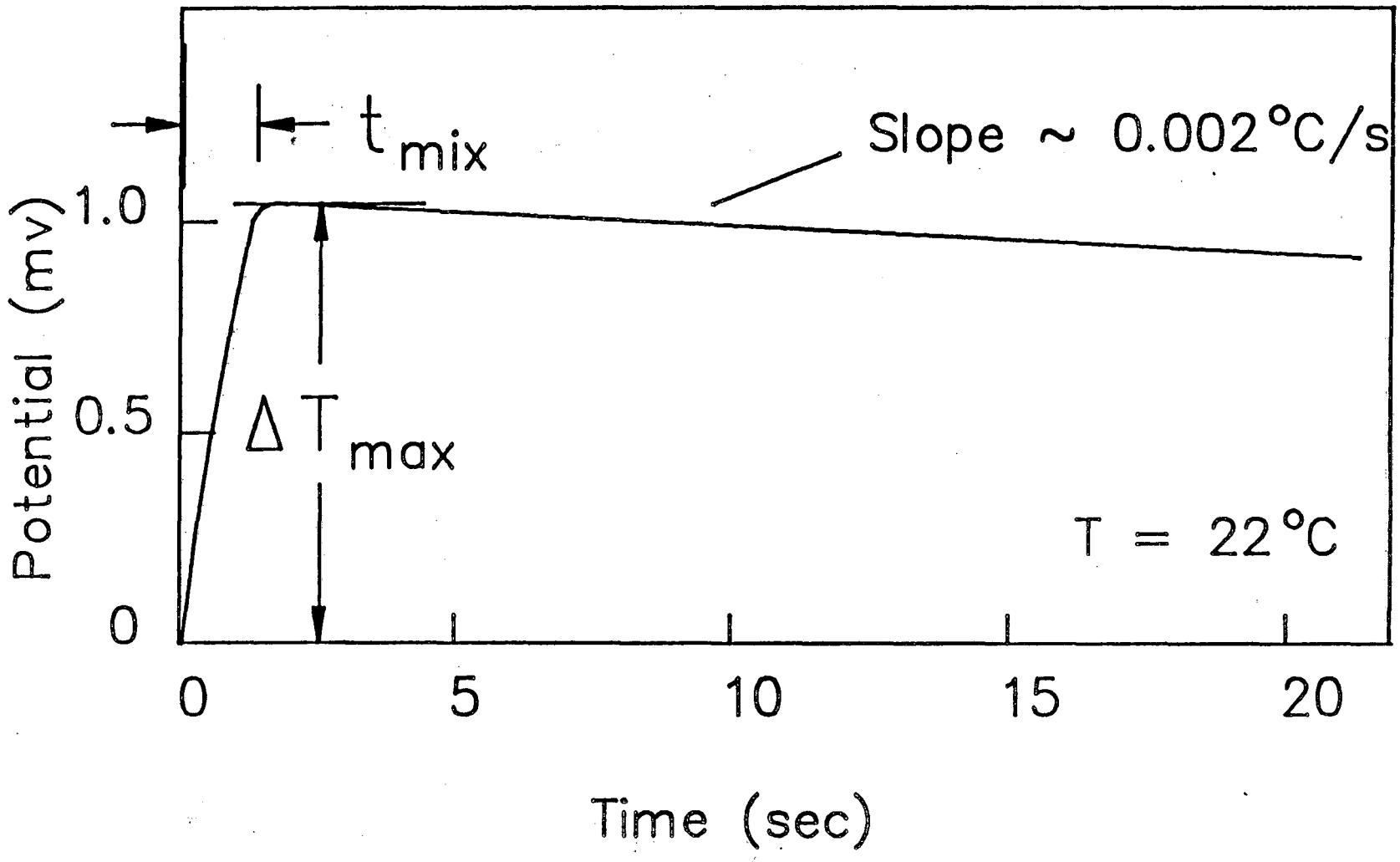
1. *Acid-Base Experiments*

Reactions between NaOH and HCl were carried out in the calorimeter to determine the rate of mixing, the thermal mass of the apparatus, and the rate of heat loss from the system. The experimental method was that used in the H_2S-SO_2 reactions and the quantities and concentrations of reactants were chosen to give similar temperature rises and thermal masses. Since the acid-base reaction is practically instantaneous, the time necessary for complete reaction is the mixing time. A heat balance applied to the apparatus and solution allows calculation of the thermal mass of the reactor. Furthermore, by measuring the decrease in temperature with time after the initial rise, the rate of heat loss from the system can be determined.

A chart recorder trace of thermocouple potential (i.e., temperature) versus time for an acid-base reaction is shown in Fig. II-2. The average value of t_{mix} was approximately 0.7 sec for the solution volumes of acid and base and stirrer speeds used in these experiments. Presumably, in cases where the reaction between H_2S

Figure II-2

Chart Recorder Trace for Acid-Base Reaction



and SO_2 lasted longer than this period, the reaction was then occurring in a homogeneous solution.

From the known heat of reaction for $H^+ + OH^-$ (= -13.4 kcal/mole of H_2O formed) and the heat capacity of the aqueous solutions (approximated as water, C_p = 1 cal/g ^oC), the thermal mass of the apparatus was determined to be 0.6 cal/ ^oC. Since in these expermiments the thermal mass of the solution for a typical run was 12-15 cal/^oC., the contribution of the apparatus to the total heat capacity of the system was small.

The heat leakage from the reaction vessel and styrofoam block corresponds to a rate of temperature decrease that is less than 0.002 ^oC/sec. Thus, there would be at most a temperature drop of about 0.02 ^oC during the ten seconds required for a typical H_2S - SO_2 run. Since in the majority of experiments the temperature rise was 5 ^oC or less, this amounts to an uncertainty of about 0.4% in the maximum temperature rise.

2. *Kinetics*

If the reaction between H_2S and SO_2 is first order in both reactants, then the differential rate equation for the disappearance of SO_2 (A) or H_2S (B) is

$$-dC_A/dt = -1/2\ dC_B/dt = k_2C_AC_B \qquad \text{(II-1)}$$

By stoichiometry

$$C_B = C_B^0 - 2(C_A^0 - C_A) \qquad \text{(II-2)}$$

in which the superscript 0 refers to time = 0. Substituting Eq. II-2 into Eq. II-1 and rearranging yields the following rate expression:

$$k_2dt = dC_A/[C_A(2C_A-2C_A^0+C_B^0)] \qquad \text{(II-3)}$$

Integration of Eq. II-3 and simplification gives

$$k_2t = [1/(2C_A^0-C_B^0)] \ln [(C_B^0/C_A^0)(C_A/C_B)] \qquad \text{(II-4)}$$

If a reaction is first order with respect to both A and B, then a plot of the integrated rate expression on the right-hand side of Eq. II-4 versus time should lie on a straight line with a slope of k_2. Note that when the initial concentrations of H_2S and SO_2 are stoichiometrically equivalent, Eq. II-4 reduces to the more familiar second-order relationship

$$2k_2t = 1/C_A - 1/C_A^0 \qquad \text{(II-5)}$$

Equation 4 is in the form of concentration and time whereas the quantities measured experimentally are temperature and time. The conversion from temperature to concentration is based on two assumptions. First, the temperature rise for the exothermic reaction between H_2S and SO_2 is assumed to be proportional to the extent of reaction -- i.e., is a measure of reaction progress. This assumption follows from the observation that there appear to be no other reactions occurring under the conditions of these experiments. Second, it is assumed that the maximum temperature rise corresponds to complete reaction (fraction conversion = 1.0). Thus, for any time $t < t_{max}$ the concentration of the limiting component is related to the temperature rise by the expression

$$C = C^o \ (1 - \ (T - T^o)/ \ (T^{max} - T^o)) \qquad \text{(II-6)}$$

From this concentration and the known stoichiometry, the concentration of the other reactant can be calculated. In order to minimize errors due to thermal losses, only temperature rises less than 90% of the maximum value were used for determining reaction-rate constants.

The preceeding analysis presupposes perfect mixing. Since these experiments require the injection of one of the reactant solutions into the other, a finite period (whose length depends on the sample volumes, solvent viscosity, and stirrer

speed) is needed for complete mixing of the components. Before mixing is complete, local inhomogeneities exist in which one of the reactants is in large excess. The other reactant is rapidly depleted there, effectively reducing the reaction rate in these regions to zero. Overall, the experimentally measured rate is lowered and the kinetics of the reaction under such conditions is of uncertain significance. However, during this period the temperature of the solution nevertheless rises in proportion to the reaction that does occur. Therefore, after the solutions are thoroughly mixed, the average concentrations of the two reactants (as determined from the temperature rise using Eq. II-6) become the homogeneous concentrations in the bulk liquid and the rate is that for the homogeneous reaction. The rate constant was determined from plots of the integrated rate expression (Eq. II-4) versus time by considering only the portions of the plots for which the time is greater than the mixing time.

3. *Heat of Reaction*

The heat of reaction for each of the experiments was calculated from the measured temperature rise and the thermal properties of the system. An enthalpy balance yields

$$H_{rxn} = \frac{[(mC_p)_s + (mC_p)_{app}]}{n_L} (T^{max} - T^o) \qquad \text{(II-7)}$$

where the subscripts "app" and "s" refer to the apparatus and solution respectively. The value n_L is the number of moles of the limiting reactant.

The heat capacity of the organic solution, including all solvents and products, was estimated by a group contribution method applied to the bulk solvent used in a given run. Calculations for Triglyme and DGM using Missenard's group contribution method, Reid et al. (1977), give a value of $C_p = 0.50 \pm 0.05$ cal/g-^{o}C.

D. Results and Discussion -- DMA as Catalyst

1. *Results for the Reaction of H_2S and SO_2 in Triglyme/DMA/H_2O*

Kinetic data were analyzed using equation Eq. II-4. Plots of the concentration expression on the righthand side of Eq. II-4 versus time yield a straight line (in the region $t > t_{mix}$) with a slope of value k_2 when the reaction follows second-order kinetics. Data pairs of potential/time obtained from the chart-recorder traces were converted into concentration/time pairs and plotted in the integrated-rate form.

A sample plot for the reaction of H_2S and SO_2 in a DMA/Triglyme mixture is shown in Fig. II-3. Values of k_2 were obtained from similar plots by drawing the best straight line through the data for $t > 1$ sec. The curvature at times less than 1 second results from the finite time required for injecting the SO_2 sample and mixing the solution. In nearly all of the runs the time intercept, as found by extrapolating the line drawn through the integrated rate data back to the time axis, was about 0.5 to 0.8 seconds. These values, which approximate quite well the mixing time in the HCl - NaOH experiments, showed little variation from run to run and thereby indicate a high degree of reproducibility in the experimental conditions and technique. Furthermore, since this result agrees with the mixing time obtained in the acid-base experiments, data points at longer times presumably represent the progress of the reaction in a well-mixed solution.

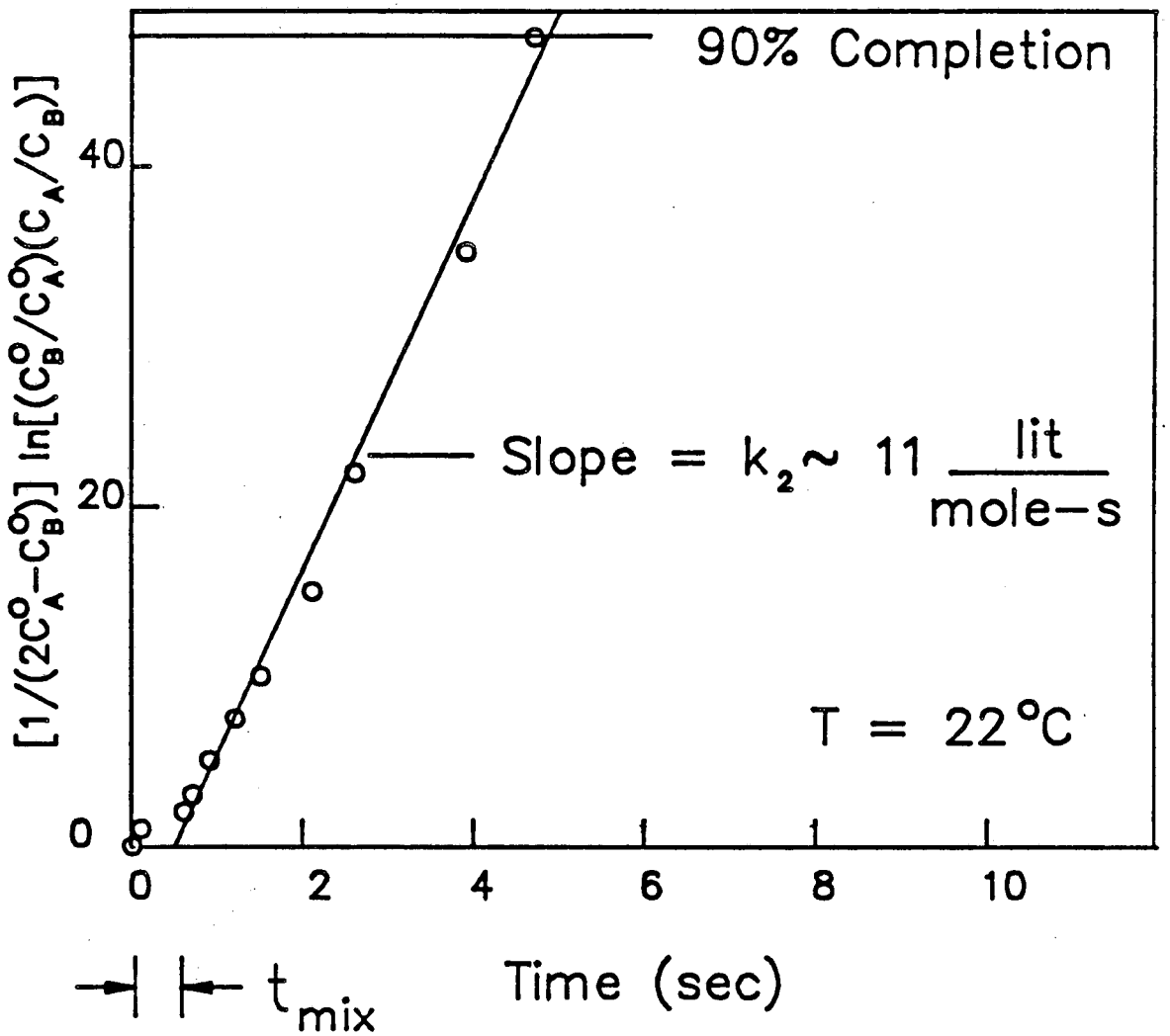
Most of the experiments presented here were performed with nearly stoichiometric equivalents of H_2S and SO_2. Additional experiments were carried out with the ratio of H_2S to SO_2 varying from about 2:1 to 1:2. In all cases the second-order rate expression provided the best straight-line fit to the data in spite of a four-fold change in SO_2 concentration. Attempts to analyze the data using a rate expression that is second order in H_2S and first order in SO_2 yielded non-linear integrated-rate plots.

Results for the reaction in Triglyme with DMA and water present are

Figure II-3

Integrated Rate Plot

(10 wt% DMA, 2 wt% Water in Triglyme)



summarized in Fig. II-4, where the second-order rate constant is plotted as a function of wt% DMA for various H_2O concentrations. With pure Triglyme as the solvent, plots of the integrated rate expression versus time are non-linear, increasing in slope as time progresses. Tangents to these curves correspond to k_2 values of 0.5 lit/mole-s or lower. The curvature, which reflects an increase of the rate constant with time, may be due to an autocatalytic effect of water formed during the reaction or possibly to a change in the reaction mechanism. Note that while adding small amounts of water to Triglyme seems to speed the reaction, it is unclear whether the same reaction pathway is followed since some of the integrated rate plots are non-linear. Perhaps other products, such as the sulfoxy acids that frequently form in aqueous media, are being created in these cases.

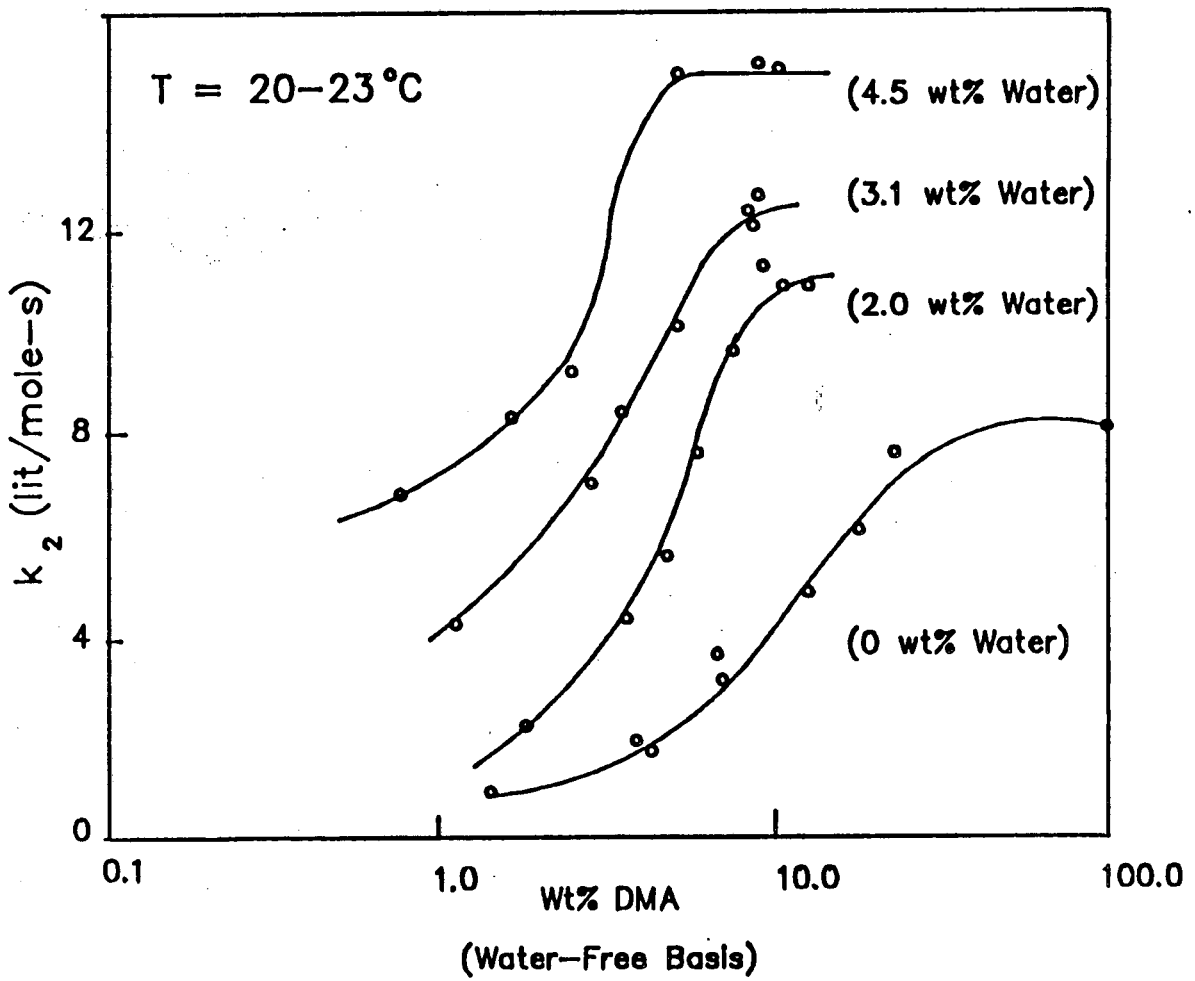
When DMA is added to Triglyme, the integrated rate plots become linear, i.e., there appears to be a first-order dependence on both H_2S and SO_2. Furthermore, addition of only 2 wt% DMA more than doubles the reaction-rate constant over that observed for Triglyme alone. Increasing the DMA concentration still further continues to increase the rate constant, k_2, which asymptotically approaches a value of 8 lit/mole-s.

The presence of water in DMA/Triglyme greatly accelerates the rate as evidenced in Fig. II-4. Addition of 4 to 5 wt% H_2O doubles the observed rate constant. As the wt% water is increased in a mixture containing 10 wt% DMA or more, the value of k_2 attains a maximum value of about 20 to 22 lit/mole-s. Note that the value of k_2 for the mixed solvents exceeds the sum of those for the reaction carried out with only one of the catalysts (DMA or H_2O) present. This suggests a synergism between the effects of the water and DMA. The mechanism for this synergism has not been determined.

2. *Results for the Reaction in DMA/DGM/Water*

The kinetic behavior of the reaction in DGM alone is similar to that observed for Triglyme. DGM has little or no catalytic effect, and plots of the integrated

Figure II-4
Variation of Second-Order Rate Constant with DMA and Water Content in Triglyme



rate expression exhibit a slope that increases with time. However, values for k_2 are considerably higher when a given wt% DMA is added to DGM than for a similar DMA/Triglyme mixture. The rate constants for the reaction of H_2S and SO_2 in DMA/DGM mixtures are shown in Fig. II-5. The DMA/DGM combination seems to present a more favorable environment for the reaction than DMA alone; when the DMA content of the mixture exceeds 4 wt% the values of k_2 exceed those for pure DMA. The maximum value of k_2, which is approximately 20 lit/mole-s, is more than twice as large as that obtained in the Triglyme/DMA cases with no water present.

The addition of water to DMA/DGM mixtures has little or no effect on the rate, as is reflected in the data presented in Table II-1. At 10 wt% DMA the rate constant has attained its maximum value and the addition of water does not increase the rate constant further.

A few runs with 2.5 wt% DMA in DGM, performed in a cold box at 7 to 10 ^{o}C, yielded an average k_2 of 4.5 lit/mole-s. On the basis of these data and those in Fig. II-5, the estimated activation energy for the reaction is 7.4 kcal/mole, which corresponds to a doubling of the rate every 20 ^{o}C. The limitations of the experimental technique prevented a more thorough investigation of the effect of temperature on the reaction kinetics. However, the fact that the activation energy is low tends to support the assumption that rate constants obtained from experiments with small temperatures variations can be used to approximate the true isothermal values at the average temperature of an experiment.

3. *Results for the Reaction in DMA/Triglyme/Methanol*

A reasonable hypothesis to explain the fact that both H_2O and DGM accelerate the reaction between H_2S and SO_2 in the presence of DMA is that hydroxyl groups are responsible for the synergistic catalytic effect. In order to substantiate this idea, a few experiments were performed in which methanol was added to DMA/Triglyme. The data are summarized in Table II-2. As the methanol

Figure II-5

Variation of Second-Order Rate Constant with wt% DMA for DMA in DGM at 17-23 °C

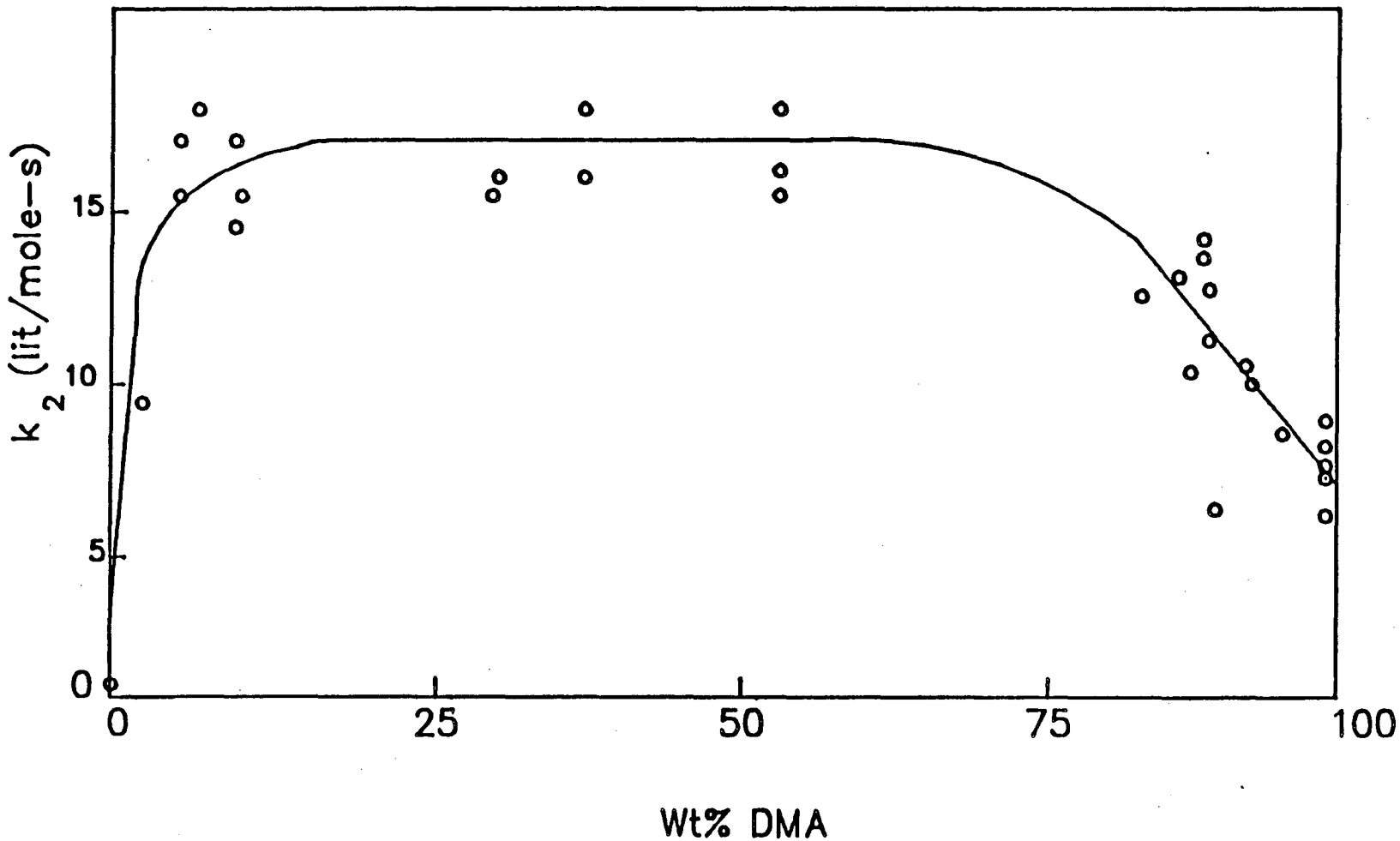


Table II-1

Rate Constants for the Reaction of H_2S and SO_2 in DMA/DGM with Water Added

(T = 23 °C)

Wt% H_2O	Wt% DMA	k_2 (lit/mole-s)
0.5	9.8	20.4
0.5	9.8	21.6
1.4	9.7	22.8
1.6	9.7	22.1
2.4	9.6	21.3
2.6	9.6	23.5

Table II-2

Rate Constants for the Reaction of H_2S and SO_2 in DMA/Triglyme/Methanol Mixtures at 10 Wt% DMA

(T = 22 °C)

WT% Methanol	k_2(lit/mole-s)
3.7	16.2
8.3	23.2
45.1	23.4

concentration is increased, the rate constant increases towards an asymptotic value which is a factor of 5 greater than than that for the same concentration of DMA alone in dry Triglyme. Adding methanol, a substance containing hydroxyl groups, to the mixture enhances the reaction rate constant. Also, the maximum rate constant is approximately equal to the maximum k_2 obtained for the reaction carried out with the same wt% DMA in Triglyme with water present and to the rate constant for the same concentration of DMA in DGM.

4. *Heat of Reaction*

The heat of reaction between H_2S and SO_2, taken as the average for a large number of data points, is 28 to 29 kcal/mole of SO_2 reacted. This value seems to apply equally well to all mixtures of the solvents used in these experiments. Evidently, effects such as heats of solution in the various solvents are small or similar in magnitude for the range of cases considered.

5. *Summary of DMA Results*

From the experimental results it is readily apparent that the presence of DMA greatly accelerates the reaction between H_2S and SO_2 in polyglycol ethers. The reaction appears to be first order with respect to both reactants over the range of concentrations tested. The nitrogen of the amine, which is noted for its ability to form complexes with sulfur dioxide (Hill and Fitzgerald, 1935), may provide a favorable site at which the reaction can occur. This effect seems to be dependent on both the concentration of catalytic agent and its amount relative to the concentration of SO_2. When the mole ratio of DMA to SO_2 exceeds one, the number of active reaction sites appears to be proportional to the SO_2 concentration.

For the reaction in DGM an additional factor is found to be important. The DGM appears to enhance the effect of DMA in accelerating the reaction, producing a maximum value of the rate constant when intermediate

concentrations of both solvents are present. This enhancement seems to be related to a catalytic effect of the OH group of the DGM that is synergistic with DMA. Further evidence of the hydroxyl effect can be seen in the results for the reaction in DMA/Triglyme with water present; indeed this effect may be expected for alcohols in general since the experiments performed with DMA/Triglyme/Methanol yielded similar results. It is possible, as suggested by Albertson and McReynolds (1943), that the solvent promotes the reaction by forming a hydrogen bridge compound with H_2S which then falls apart to yield a reactive hydrosulfide ion. Determination of whether the hydroxyl provides an alternate reaction site or speeds an intermediate or parallel step or causes some other enhancement will require further investigation.

E. Catalyst Survey

1. *Catalyst Structure*

At this point only the catalytic activity of DMA has been discussed. However, other tertiary amines can also catalyze this reaction. To determine which other compounds are good catalysts, the investigation of the reaction kinetics was extended to a variety of amines. Rate experiments were conducted as outlined above and the data analyzed using Eqs. II-4 and II-6. The results of these studies are given in Table II-3. An examination of the data reveals that catalytic activity is a strong function of certain chemical properties of the catalyst.

The most striking feature of the data is the difference in the rate constants for the heterocyclic amines compared to those for non-heterocycles. In general a non-heterocycle must be present in 10 to 100 times greater concentration than a similar heterocycle to produce an equivalent rate constant. For example, compare DMA to 3-pyridyl carbinol (3PC). As can be seen in Table II-4, a major structural difference between the two molecules is in the location of the nitrogen. 3-pyridyl carbinol, which has a nitrogen in the aromatic ring, produced the same level of catalysis (i.e. rate constant) at a 30-fold lower concentration than DMA. Whether

Table II-3

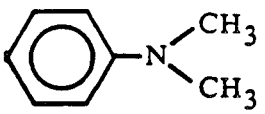
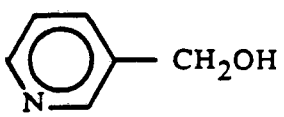
Rate Constant Summary for the Reaction of H_2S and SO_2 in DGM

CATALYST	k_2 (L/MOLE-S)	B.P. (^oC)
Non-heterocyclic amines		
N,N-diethyl toluamide (0.79M)	0.49	111 (1 mm)
Phenyl diethanol amine (0.44M)	0.71	228 (15 mm)
Triethyl amine (0.83M)	13.6	89
N,N-Dimethyl aniline (0.44M)	16.0	194
Heterocyclic Amines (All at 0.07M)		
Acridine	Reacted with SO_2	346
2,2'-Bipyridine	0.38	273
2-Phenyl pyridine	0.72	268
Quinoline	2.80	238
3-Phenyl pyridine	3.40	270
2-Benzyl pyridine	4.80	276
4,4'-Bipyridine hydrate	6.34	305
3-Pyridyl carbinol	6.41	266
4-Benzyl pyridine	8.33	287
Pyridine	8.40	115

Table II-4

Effect of Heterocyclic Nitrogen on Reaction Rate Constant

Concentration of Catalyst to Achieve k_2 = 16 L/MOLE-S

Catalyst	[CAT] (MOLE/LIT)	Structure
N,N-Dimethyl Aniline	0.440	
3-Pyridyl Carbinol	0.015	

this effect is due to the electronic environment of the nitrogen or to the structural arrangement of the molecule is not known.

However, it is clear that in some cases steric effects are important in determining how well a given catalyst performs. There are several examples of this property in the systems studied. Among the heterocyclic compounds, the accessibility of the nitrogen in 3- phenyl pyridine, 4,4'-bipyridine, and 4-benzyl pyridine is unobstructed while the approach to the analogous nitrogen in 2-phenyl pyridine, 2,2'-pipyridine, or 2-benzyl pyridine is partially hindered by a large adjacent molecular group. The structures are detailed in Table II-5 where the rate constants obtained for the reaction in DGM are also given. When the aromatic nitrogen is next to a carbon containing a bulky phenyl or pyridyl group the rate constant is greatly reduced. However, if the environment of the nitrogen is unobstructed, then the second order rate constant is increased by a factor of 5 to 20.

A further example of the effect of bulky substituents on the catalytic ability of tertiary amines is evidenced by the order-of-magnitude difference in the rate constant obtained for the reaction in DMA/DGM versus that in Phenyl diethanol amine/DGM. Apparently the presence of the bulky ethanol substituents reduces the accessibility of the reactive site relative to the case where there are methyl groups on the nitrogen.

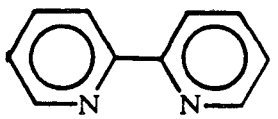
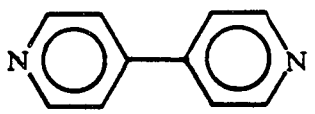
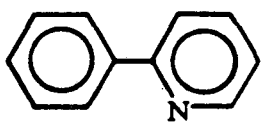
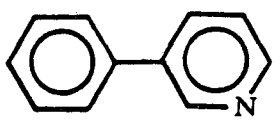
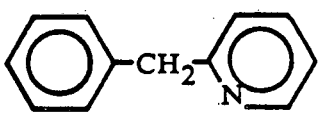
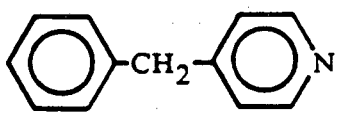
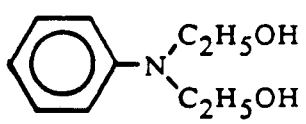
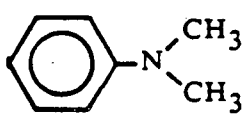
2. *Parametric Studies of the Reaction using 3PC in DGM and Triglyme*

Since the properties and catalytic ability of 3PC make it a promising candidate for use in a process for the reactive removal of H_2S from industrial gas streams, more detailed studies of its catalytic activity were needed. In order to determine the effect of solvent composition on the reaction rate constant, a series of runs with varying concentrations of 3PC in DGM and Triglyme were performed. The results for the reaction of H_2S and SO_2 in Triglyme are summarized in Fig. II-6, where the results obtained in DMA/DGM have also been

Table II-5

Steric Effects on the Reaction Rate Constant

([CAT] = 0.07 M except as noted)

Catalyst	k_2	Structure
2,2'-bipyridine	0.38	
4,4'-bipyridine	6.34	
2-phenyl pyridine	0.72	
3-phenyl pyridine	3.40	
2-benzyl pyridine	4.80	
4-benyzl pyridine	8.33	
Phenyl diethanol amine (0.44M)	0.71	
N,N-Dimethyl aniline (0.44M)	16.0	

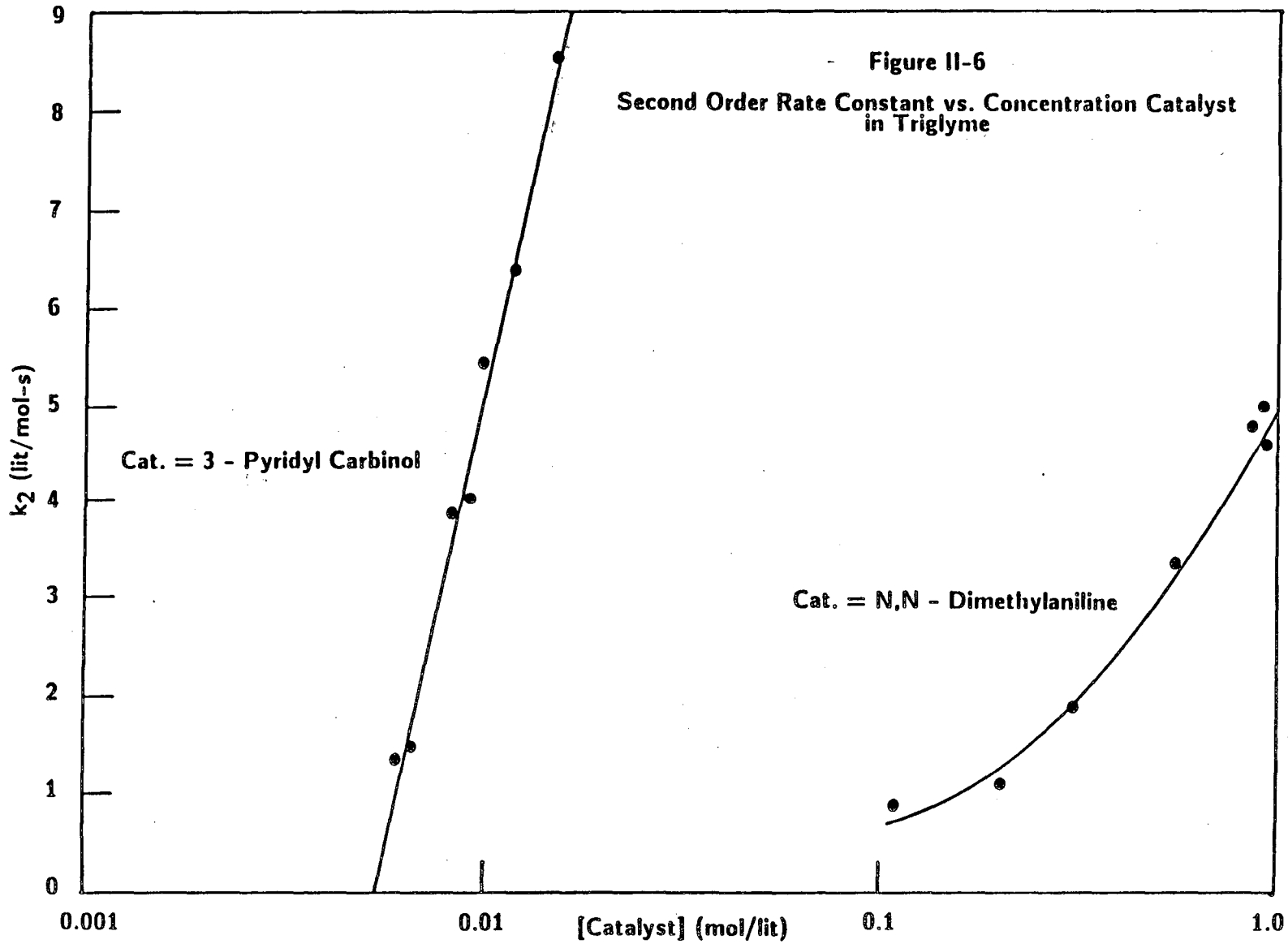


Figure II-6

Second Order Rate Constant vs. Concentration Catalyst in Triglyme

reproduced.

Evidently 3PC is a considerably better catalyst than DMA for the reaction carried out in Triglyme. A concentration of DMA one to two orders of magnitude higher than that of 3PC is required to obtain the same second-order rate constant. Furthermore, the rate of increase of k_2 with concentration of catalyst is greater when 3PC is used.

A few data points for the reaction carried out in 3PC/DGM are given in Table II-6. Comparing these results to those for 3PC/Triglyme, it is apparent that at these low concentrations of catalyst the rate constants are somewhat higher and that the increase in rate constant with concentration is faster. This is similar to the results obtained for DMA in DGM and Triglyme where the presence of the hydroxyl group of the monoether accelerated the rate over that obtained in the diether.

Whether the rate constant would continue to increase monotonically as indicated in Fig. II-6 or reach a maximum value as in the DMA/Triglyme or DMA/DGM runs is not clear at present. An accurate assessment of the potential for accelerating the reaction by further addition of 3PC will require a different experimental technique, since the present one is limited by mixing time and the response of the chart recorder to second-order rate constants of less than 100. However, it is apparent that rate constants of at least 100 are obtainable with relatively low concentrations of 3PC.

3. *Water Effect*

When water is added to mixtures of DMA/Triglyme the reaction rate constant was found to increase markedly. The addition of water to 3PC in Triglyme seems to have a similar effect (see Table II-7). Water accelerates the reaction by a factor of 4 to 5 over that obtained for the same concentration of 3PC in Triglyme alone.

F. Selection of Catalyst

Having determined the rate constants for several tertiary aromatic amines, it

Table II-6

Second-Order Rate Constant for the Reaction in Mixtures of 3-pyridyl carbinol (3PC) and DGM

[3PC] (MOLE/L)	k_2 (L/MOLE-S)
0.0105	6.4
0.0144	17.7
0.0500	>100.0

Table II-7

Effect of Water on the Reaction in Mixtures of 3-pyridyl carbinol (3PC) and Triglyme

[3PC] (MOLE/L)	[Water] (MOLE/L)	k_2
0.0061	0.0	1.3
0.0061	1.1	4.2
0.0150	0.0	8.5
0.0150	2.1	39.3

is possible to select a catalyst to use in a process for the removal of H_2S by reaction with SO_2 in glycol ethers. A suitable catalyst from a process standpoint should exhibit the following characteristics:

1. It must catalyze the reaction, producing a sufficiently high rate of reaction to permit nearly complete removal of H_2S in a reasonable size physical operation.

2. The vapor pressure of the catalyst at the temperatures likely to be present in the process (usually $T < 120\ ^{o}C$) should be low, thus preventing volatile loss of the chemical.

3. Since polyglycol ethers exhibit properties which make them attractive as bulk solvents for the process, the catalyst should be soluble in the mono and diethers of di and higher glycol ether derivatives.

4. The catalyst must not react with any of the reactants or products. This generally eliminates primary and secondary amines, which react with product sulfur, and some tertiary amines which complex irreversibly with one or both of the reactants.

5. The catalyst should be readily available at reasonable cost.

As shown above, sterically hindered amines can be eliminated from consideration because of the relatively low reaction rate constants obtained with their use. Additionally, acridine, which forms an insoluble adduct with SO_2, and pyridine, which exhibits excellent catalytic properties but boils at a low temperature, are rejected as possible catalysts.

Of the remaining catalysts tested, quinoline and 3PC are promising choices. Quinoline is relatively cheap and moderately high boiling, but is only slightly soluble in water. However, 3PC has a low vapor pressure at the temperatures of interest and is miscible with water. This last property facilitates its recovery from stack gases by aqueous scrubbing and reduces the possibility of a stream splitting into separate organic and aqueous phases. Thus, 3-pyridyl carbinol is the catalyst chosen for subsequent development of the present process for H_2S removal.

III. Process Development

A. Background

A major goal of the work for this thesis was to develop a process scheme which demonstrates the feasibility of using the reaction between H_2S and SO_2 in organic solvents for sweetening a sour gas stream from a coal gasifier. Since the intended use for this gas is power generation in a gas turbine, the total sulfur content of the gas must be reduced to the ppm level to protect the equipment from corrosion as well as to meet environmental specifications.

The removal of H_2S from gas streams has been the subject of intense study in recent years. As noted in the introduction, many methods employing a variety of different techniques have been proposed as solutions to this problem. The process considered here relies on the reaction between H_2S and SO_2 carried out in the liquid phase at moderate temperatures (< 150 ^oC). Under these conditions the reaction is irreversible and with a suitable catalyst/solvent system can be made to go essentially to completion in a short time. Quantitative removal of H_2S is possible, thereby satisfying stringent environmental requirements. Although processes of this type have been mentioned in the patent literature (e.g. Townsend, 1959; Deal et al., 1968; Renault, 1969; Tanimura, 1976), none has been successfully commercialized in an application where an exit H_2S concentration of a few ppm is needed.

B. Simulation

1. *Main Program*

Since a detailed analysis of the potential flow configurations for the process was desired, a computer simulation capable of performing energy and material balances was written. Such calculations permit rapid evaluation of different arrangements of unit operations and make it possible to identify important process requirements and constraints. An overview of the simulation is presented here; for

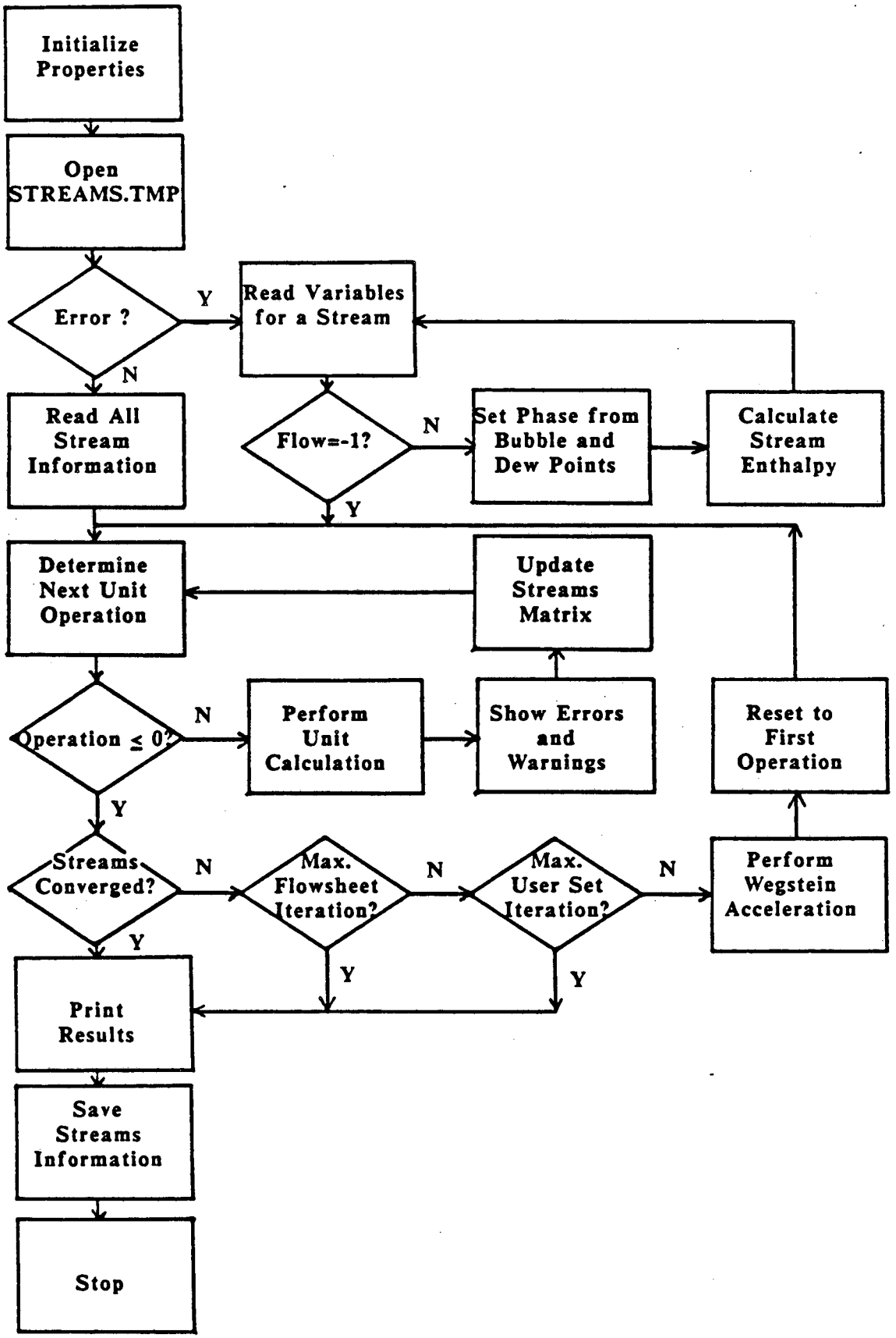
an in-depth discussion of calculational methods the reader is referred to the Appendix.

A flowchart for the simulation main program is given in Fig. III-1. The external framework of the simulation is fairly simple. The parameters for all streams including composition, flow, temperature, pressure, enthalpy, and phase are stored in a single array in the main program. This stream matrix is initially filled with values read from a disk file which contains user-specified feeds and initial guesses where needed. For streams with non-zero flows the program determines the phase by calculating the bubble and dew points. Stream enthalpies are also calculated. If the results from a prior run of the simulation are available, they may be used as the initial values for the present calculations. This is particularly advantageous when minor changes have been made to a flowsheet, since the solution to the new problem will probably not be greatly different from that obtained previously.

Unit operations in the flowsheet are calculated in sequence from a data file with the stream variables and other parameters passed to the unit subprograms as needed. The outlet streams from each operation are used to update the global stream matrix.

After all of the unit calculations have been performed, the program tests for convergence by comparing the current stream variables with those from the previous iteration. If the variables are not converged, then Wegstein acceleration (Perry, 1984, pg. 2-62) is applied to the component flows. Usually the accelerated variables are closer to the correct solution than those obtained by simple direct substitution. The results of the previous iteration are combined with the Wegstein-accelerated variables where appropriate to form a new set of guesses for the next iteration. The sequence of calculations is repeated until component flows and stream temperatures are converged to one part in 1000 or until a user-specified maximum allowable iteration count is reached.

Fig. III-1 Program Flowchart



2. *Unit Operations*

The flowsheet simulation program supports a variety of unit operations (see Table III-1). A brief description of the function and purpose of the available operations of the major types of units is given below.

Absorber/Stripper/Distillation

The absorber/stripper/distillation operation is used to perform stage-to-stage column calculations. This routine has been designed to handle a number of different process conditions including specifications for a reboiler, a condenser, and interstage coolers. Calculations are carried out using a multivariate Newton-Raphson algorithm modified to account for the presence of chemical reaction between H_2S and SO_2. The subroutine returns exit liquid and vapor streams and the overhead condensate if one is obtained. The internal temperature, flow, and composition profiles for the column are saved in a separate data file.

Materials Handling Routines

Several units have been provided to simulate materials handling and transfer operations. In the compressor and pump operations the inlet stream is transferred to the outlet at a specified pressure. The exit stream enthalpy and phase are determined by an isenthalpic flash calculation. An estimate of the power requirement is made by assuming ideal gas behavior in the compressor and that the fluid being pumped are incompressible.

Mixer and splitter routines are provided for combining and separating streams respectively. The mixer operation also performs an isenthalpic flash to determine the exit enthalpy and phase of the mixture. A warning is issued if a mixed phase (i.e. liquid and vapor) is formed at the exit conditions.

A make-up unit is included to deal with those places in the process where a constant flow of a component or stream is desired. The three types of operation supported are:

Table III-1

Major Unit Operations in the Simulation

ABSORBER/STRIPPER/DISTILLATION

COMPRESSOR

CRYSTALLIZER

FURNACE

HEATER

HEAT EXCHANGER

MAKE-UP

MELTER/DECANTER

MIXER

PARTIAL CONDENSER

PHFLASH

PUMP

REACTOR

SPLITTER

SULFUR CENTRIFUGE

SULFUR SETTLER

Utility Subroutines

BUBBLE AND DEW POINTS

THERMODYNAMICS

SULFUR SATURATOR

1. Set flow of component I, other component flows unchanged.

2. Set flow of component I, other component flows scaled based on inlet composition.

3. Set total stream flow.

Sulfur Handling Operations

There are three major units that simulate the necessary sulfur handling operations in the process. The crystallizer calculates the amount of sulfur that precipitates under specified conditions and modifies the exit solid and dissolved sulfur concentrations appropriately. The slurry from this unit is generally fed to a sulfur settler in which the bulk of the solvent stream is separated from an exit slurry with a predetermined weight fraction of solid sulfur.

The operation of a centrifuge is performed by a subroutine in which a cake of a fixed wt% sulfur is formed and subsequently washed and settled to form a new slurry. The liquid obtained from the centrifugation and washing steps and the exit slurry are available for further processing.

Reaction Vessels

In the furnace sulfur, H_2S, and hydrocarbons are burned to form SO_2 and CO_2. The routine determines the amount of O_2 (air) required by assuming complete combustion. The quantity of sulfur needed to produce the desired SO_2 outlet flow is calculated taking into account any H_2S and SO_2 in the inlet. The total heat released in the furnace is recorded.

The liquid-phase reaction between H_2S and SO_2 is carried out in bulk in the reactor. This unit calculates the quantities of H_2S and SO_2 removed and sulfur and H_2O formed given the reaction rate constant and the fraction conversion. The temperature and enthalpy of the outlet stream are adjusted for heat changes associated with externally applied cooling or the with crystallization of sulfur. A reactor size for carrying out this second order reaction is estimated for plug-flow operation.

Heat Transfer and Equilibrium Operations

The four major heat transfer and equilibrium operations are a heat exchanger, a melter/decanter, a partial condenser (or isothermal flash), and an adiabatic or isenthalpic flash. The heat exchanger routine calculates the exit cold and hot stream temperatures and enthalpies given the exchanger area and heat transfer coefficient. Phase changes, such as flashing of light gases from solution upon heating are taken into account.

In the melter/decanter sulfur is melted and separated from the other inlet components (usually water with small amounts of solvent and dissolved gas). The heat required for this operation is returned by the unit.

The two flash calculations, isothermal and isenthalpic, return the exit liquid and vapor streams obtained under specified conditions. These routines not only provide a procedure for performing desired flash operations, but also are used by several other unit subroutines to determine the phase(s) and enthalpy for a given stream.

Utility Routines

A few subroutines included in the simulation perform support calculations for the other units. For example bubble and dew point temperatures for a stream are calculated in one subroutine. Another determines the mole fraction of sulfur in a stream at saturation (sulfur saturator). The thermodynamics subroutine returns values for and derivatives of enthalpies and equilibrium constants for a specified stream input.

C. Process Description

1. *Problem Specification*

The basis of the design is the gasification of 50,000 kg/hr of Illinois #6 bituminous coal using air and steam at a pressure of 2500 kPa. This rate of fuel use would yield about 120 MW of electric power assuming an overall thermal efficiency of 40%. The total quantity of gas to be treated is 8,950 kmole/hr with an H_2S content of about 0.62%. A complete specification of the gas composition is given in Table III-2. The fuel gas is assumed to have been scrubbed with water at 45 ^oC to eliminate particulates and to condense excess steam ahead of the H_2S-removal process.

The design specification for the exit concentration of H_2S is 1 part per million. Furthermore, since the gas is to be burned in a gas turbine, it is desired to remove H_2S selectively, leaving the flows of other components and thus the volume of the gas unchanged.

The computer simulation was used to evaluate a number of different process configurations for this application. Figure III-2 is the current flowsheet for the process. The component balances, temperatures, and pressures are given in Table III-3. Tetraglyme and 3-pyridyl carbinol were chosen as the bulk solvent and reaction catalyst respectively. Gas solubilities were obtained from Demyanovich (1984) and Sciamanna (1986). A description of the process follows.

2. *Processing Steps*

Primary Absorber

Sour gas is fed to the bottom of the primary absorber where the bulk of the H_2S is removed by physical absorption in solvent recycled from the reactor/cystallizer. The H_2S level in the gas is reduced to about 600 ppm in 15 theoretical trays (L/mV ~ 1.5). In the section immediately above this point residual H_2S is absorbed and reacts with a solution of SO_2. The liquid fed to this

Table III-2

Design Basis for Process

(120 Megawatt Power Plant (Gas Turbine))

Sour Gas Characteristics

Composition: (mole fractions)
(Denn et al. 1979)

H_2	= 0.2729	H_2O	= 0.00290
N_2	= 0.4130	H_2S	= 0.00614
CO_2	= 0.1902	CH_4	= 0.00145
CO	= 0.1133		

Flow = 2.488 kmole/s

T = 368 K

P = 2500 kPa

Process Solvent/Catalyst

Tetraglyme/ 3-pyridyl carbinol

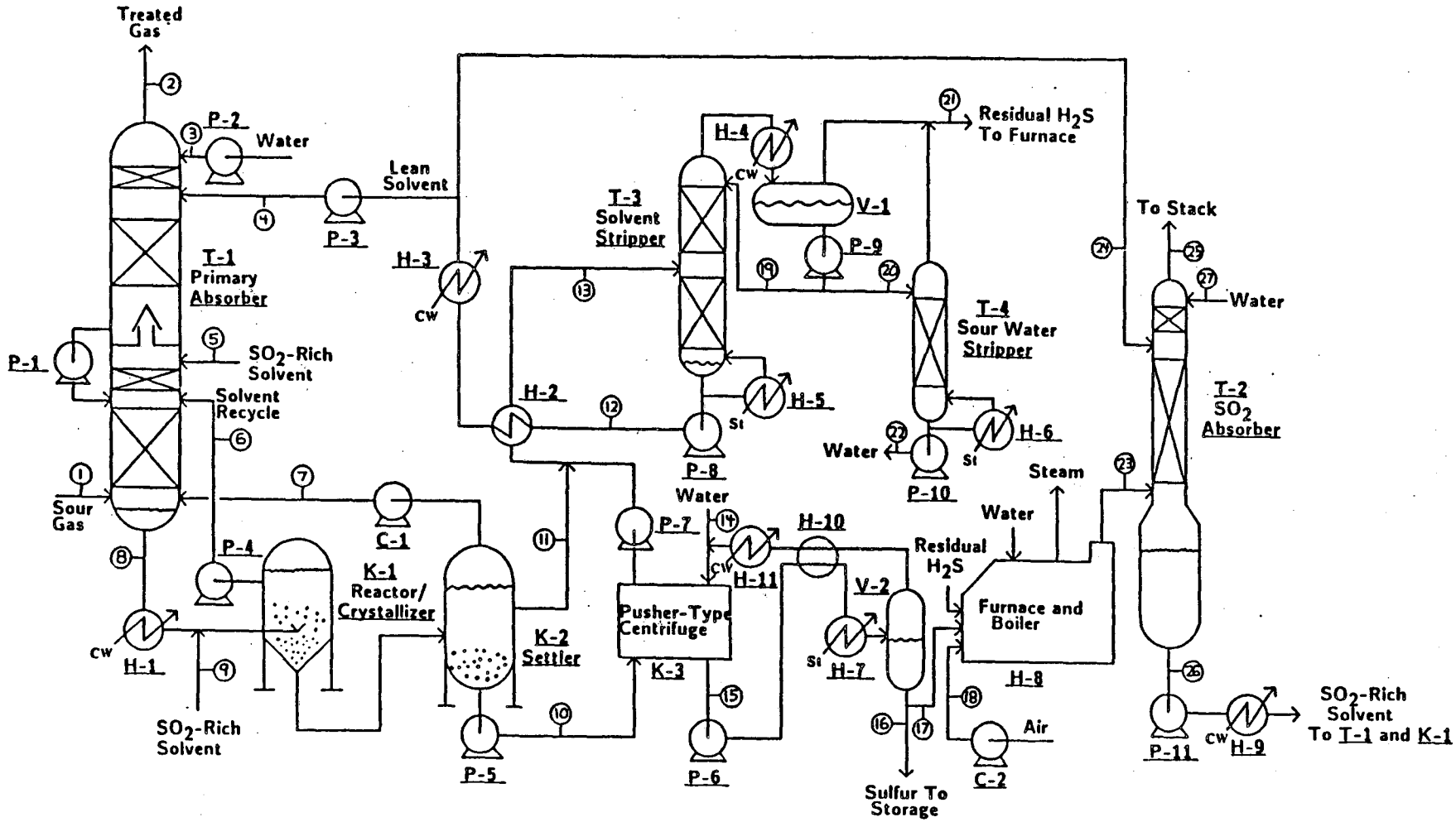


Figure III-2

Flowsheet for Process

Table III-3

Stream Flows and Conditions for Process

	Stream:	1	2	3	4	5	6	7	8	9
H_2	kmol/hr	2444.3	2444.3	--	--	--	4.3	0.6	4.9	--
CO_2	'	1703.6	1688.7	--	--	0.04	280.8	26.4	322.1	0.36
N_2	'	3699.2	3699.2	--	--	--	5.4	0.8	6.2	0.02
CH_4	'	13.0	13.0	--	--	--	0.09	0.01	0.1	--
CO	'	1014.8	1014.8	--	--	--	3.1	0.4	3.5	--
H_2S	'	55.0	--	--	--	--	8.4	0.2	58.2	--
SO_2	'	--	--	--	--	2.7	--	--	--	24.3
H_2O	'	26.0	17.5	36.0	13.2	1.9	887.1	0.5	952.6	17.2
S (dissolved)	'	--	--	--	3.0	0.2	47.2	--	58.5	2.5
S (solid or liquid)	'	--	--	--	--	--	--	--	--	--
Total	kmol/hr	8955.9	8877.5	36.0	16.2	4.84	1236.39	28.91	1406.1	44.38
Solvent Flow	kmol/hr	--	--	--	140.4	13.0	1823.8	--	1977.2	116.6
Pressure (kPa)		2510	2500	2510	2510	2510	2510	2510	2500	2510
Temperature (deg C)		95	33	40	30	30	30	54	55	30

Table III-3
(Continued)

Stream Flows and Conditions for Process

Stream:	10	11	12	13	14	15	16	17	18
CO_2 '	0.6	14.6	--	15.2	--	--	--	--	--
N_2 '	--	--	--	--	--	--	--	--	103.4
O_2 '	--	--	--	--	--	--	--	--	27.5
H_2S '	0.05	0.95	--	1.00	--	--	--	--	--
SO_2 '	--	--	--	--	--	--	--	--	--
H_2O '	5.4	125.4	25.4	145.2	14.4	287.8	--	--	--
S (dissolved)'	0.3	5.4	5.7	5.7	--	--	--	--	--
S (solid or liquid) '	81.0	--	--	--	--	81.0	55.0	26.0	--
Total kmol/hr	87.35	146.35	31.1	167.1	14.4	368.8	55.0	26.0	130.9
Solvent Flow kmol/hr	11.2	258.8	270.0	270.0	--	--	--	--	--
Pressure (kPa)	120	120	120	110	500	100	120	120	110
Temperature (deg C)	27	27	153	117	40	38	120	120	40

Table III-3
(Continued)

Stream Flows and Conditions for Process

Stream:	19	20	21	22	23	24	25	26	27
CO_2 '	0.02	0.05	15.2	--	15.2	--	14.8	0.4	--
N_2 '	--	--	--	--	103.4	--	103.4	0.02	--
H_2S '	--	0.01	1.00	--	--	--	--	--	--
SO_2 '	--	--	--	--	27.0	--	--	27.0	--
H_2O '	39.5	118.5	1.5	118.3	2.5	12.2	2.7	19.2	7.2
S (dissolved)'	--	--	--	--	--	2.7	--	2.7	--
S (solid or liquid) '	--	--	--	--	--	--	--	--	--
Total kmol/hr	39.52	118.56	17.7	118.3	148.1	14.9	120.9	49.32	7.2
Solvent Flow kmol/hr	0.01	0.02	--	0.02	--	129.6	--	129.6	--
Pressure (kPa)	110	110	110	110	110	110	100	100	110
Temperature (deg C)	40	40	42	100	150	30	20	61	40

section contains about 10% of the total stoichiometric requirement of SO_2 for the process. A chimney tray permits the vapor to leave this section, whereas the liquid from the section above is bypassed to the section below in order to maintain a concentration of SO_2 high enough to remove most of the remaining H_2S from the gas in 5 stages.

Residual H_2S and desorbed sulfur dioxide are recovered from the gas in the top section by scrubbing with lean cool solvent in 6 to 8 trays (L/mV for SO_2 ~ 2). Two trays are provided at the top of the absorber for solvent recovery by a water wash. The treated gas leaving the absorber contains less than 1 ppm H_2 and SO_2.

Reactor/Crystallizer

The effluent liquid from the absorber is cooled and mixed with a solvent stream containing sufficient SO_2 to react with slightly less than 90% of the H_2S entering the process. This mixture is forwarded to a vessel where the liquid-phase reaction between H_2S and SO_2 is allowed to continue nearly to completion. Driving force for the reaction is provided by a slight excess of H_2S. The sulfur formed by the reaction is allowed to settle and is removed as a slurry.

Approximately 85% of the solvent leaving the reactor is pumped back to the absorber to provide the liquid for bulk absorption of H_2S. By recycling most of the solvent required in the absorber without reducing the pressure, pumping costs associated with solvent circulation are kept low.

Settler and Solvent Surge Vessel

The sulfur slurry leaving the reactor is cooled to 35 ^{o}C and fed to the settler and solvent surge vessel where the mixture is flashed to atmospheric pressure. The flash gas, containing mostly CO_2 with small amounts of H_2S and lighter components (H_2, N_2, CO), is compressed and recycled to the bottom of the primary absorber. A slurry of sulfur is withdrawn from the bottom of the tank and treated to separate the sulfur from the solvent. Sufficent solvent is withdrawn from the

storage tank for use in the SO_2 removal steps in the primary absorber and the SO_2 absorber.

Solvent Stripper

The solvent from the settler is preheated to 120 ^oC and fed to a solvent stripper operating at atmospheric pressure. In this column the liquid is reboiled to reduce the water content of the solvent to about 1.0 wt%. The water provides a stripping vapor for residual H_2S and hydrocarbon (mostly CO_2). About 5 stages are needed in the bottom of the stripper to reduce the H_2S content of the solvent product to less than 1 ppm. The overhead is condensed with cooling water. Part of the condensed water is refluxed at the top of the column to minimize solvent loss. The slightly sour water (1 to 20 ppm H_2S) is stripped of H_2S and used in the centrifuge and in the water wash sections in the absorbers. The net water of reaction is sent to disposal.

The bottoms stream leaving the stripper is cooled in exchange with the incoming wet solvent. This cool lean solvent is used in the SO_2 absorption steps in the primary and SO_2 absorbers.

Sulfur Separation

The sulfur slurry from the settler is fed to a pusher-type centrifuge. Solvent-free water from the sour water stripper is used first to rinse the sulfur cake and then to re-slurry the sulfur. The solvent stream and rinse water are pumped to the solvent stripper. The sulfur-water slurry is heated above the melting point of sulfur and the two liquids are decanted. Water from this separation is returned to the centrifuge. The net sulfur product is pumped to a sulfur pit for storage.

SO_2 Absorber

A gas containing about 18% SO_2 is formed by the combustion of part of the molten sulfur and the residual H_2S from the solvent and sour water strippers with air in a furnace. The heat generated is used to produce high pressure steam in a waste-heat boiler. SO_2 is absorbed from the gas by contacting it with cool lean

solvent in a 7- to 10-stage column operating at atmospheric pressure. The exit SO_2 concentration in the gas is less than 1 ppm. The solvent flow in this unit is set to provide cooling of the inlet gas stream and to remove the heat of absorption of SO_2. A water wash is provided at the top of the column to minimize solvent loss. The bottom of the scrubber serves as a tank to provide an inventory of SO_2 solution. Storage of SO_2 solution permits partial decoupling of the rate at which SO_2 is generated from the rate of H_2S absorption.

3. *Summary*

An equipment list and utilities estimates are shown in Tables III-4 and III-5 respectively. An economic analysis is not presented here; for a detailed comparison of this process with conventional technology see Lynn et al. (1986). However, the flowsheet and data presented illustrate the potential utility of the present technology for nearly complete removal of hydrogen sulfide from a sour gas stream.

The process is relatively simple and has modest equipment and utilities requirements. In its present configuration it has the capacity to reduce easily the effluent H_2S and SO_2 to less than 1 ppm, thus meeting environmental specifications. Furthermore, because the process is highly selective for H_2S, the total reduction in the molar flow of the gas being treated is less than 1%, which is of importance when the treated gas is to be burned in a combustion turbine. Of the other components in the gas only CO_2 (5.5% reduction) and H_2O (32% reduction) are removed to any extent. This loss of water is not considered significant because the entering gas contains only 0.29% H_2O.

The value of using the reaction between H_2S and SO_2 in organic solvents has been demonstrated. Bulk formation of sulfur is facilitated by the reaction and reactive absorption permits quantitative removal of H_2S. This combination is put to good advantage in the present process; solvent recycle for physical absorption of most of the hydrogen sulfide is made possible by the homogeneous reaction

Table III-4

Equipment List

T-1	Primary Absorber
T-2	SO_2 Absorber
T-3	Solvent Stripper
T-4	Sour Water Stripper
K-1	Reactor/Crystallizer
K-2	Settler/Solvent Surge Tank
K-3	Pusher-Type Centrifuge
V-1	Reflux Drum
V-2	Decanter
H-1	Absorber Bottoms Cooler
H-2	Solvent Preheater
H-3	Solvent Cooler
H-4	Stripper Condenser
H-5	Solvent Stripper Reboiler
H-6	Sour Water Stripper Reboiler
H-7	Sulfur Melter
H-8	Furnace and Boiler
H-9	SO_2 Solution Cooler
H-10	Melter Preheater
H-11	Wash Water Cooler

Table III-4 (continued)

P-1	Absorber Bypass Pump
P-2	Primary Absorber Water Wash Pump
P-3	Primary Absorber Lean Solvent Pump
P-4	Primary Absorber Solvent Recycle Pump
P-5	Sulfur Slurry to Centrifuge Pump
P-6	Sulfur/Water Slurry to Decanter Pump
P-7	Solvent to Stripper Pump
P-8	Lean Solvent Pump
P-9	Condensate Pump
P-10	Waste Water Pump
P-11	SO_2 Solution Pump
C-1	Flash Gas Compressor
C-2	Air Compressor

Table III-5

UTILITIES USE AND COSTS

Electric Power		
Flash Gas Compressor, C-1	92	kW
Air Blower, C-2	11	
Lean Solvent Pump, P-3	33	
Sulfur Slurry Pump, P-6	13	
Solvent Forwarding Pumps; P-4, P-5, P-8, P-11	4	
Small Pumps; P-1, P-2, P-7, P-9, P-10	1	
Total	154	kW
Annual* Cost @ $0.07/kW-hr	**$86,000**	
Steam produced, 4200 kPa (615 psia)		
Waste heat boiler	< 4,300 >	kg/hr
Annual* Credit @ $11.50/10^3 kg	**< $395,000 >**	
Steam consumed, 1150 kPa (165 psia)		
Solvent Stripper Reboiler, H-5	3650	kg/hr
Sour Water Stripper Reboiler, H-6	140	
Sulfur Melter, H-7	270	
Total steam consumed	4060	kg/hr
Annual* Cost @ $9.90/10^3 kg	**$321,500**	
Cooling Water **		
Solvent Coolers; H-1, H-3, H-9	1440	gal/min
Water Cooler, H-11	23	
Solvent Stripper Condenser, H-4	380	
Total cooling water usage	1843	gal/min
Annual* Cost @ $0.10/10^3 gal	**$88,500**	
Total Net Annual* Cost for Utilities	**$101,000**	

* Annual costs based on an operating rate of 8,000 hrs/yr.

** Cooling Water available at 25 ^{o}C. Exit Temp. = 45 ^{o}C.

occurring in the reactor/cystallizer, while reactive absorption is employed to reduce the exit concentration of sulfurous compounds to desired levels. Since most of the liquid is recycled to the absorber without a reduction in pressure, solvent pumping power requirements are relatively low.

The processing scheme also takes advantage of the 30-fold greater solubility of SO_2 relative to H_2S in tetraglyme and similar solvents. Because the exit vapor from the reaction section of the absorber contains small amounts of desorbed SO_2 and little or no H_2S, the solvent flow needed for complete removal of all sulfurous compounds is much lower than would be necessary for removal of H_2S to a comparable level. This configuration keeps the flow of lean solvent low. Where a sulfurous compound must be removed essentially quantitatively by physical absorption, that compound is always SO_2.

Conversely, the fact that the effluent from the reactor/crystallizer is slightly H_2S-rich means that less soluble H_2S rather than SO_2 needs to be stripped from the solvent. The coupling of a low solvent flow with the presence of an easily removed component makes it possible to keep stripping costs to a minimum.

Although an economic analysis is beyond the scope of the present investigation, a significant reduction in utilities cost could be achieved by lowering the power requirements of the process, represented chiefly by the lean solvent pump (P-3) and the flash gas compressor (C-1). The flow of lean solvent to the top of the primary absorber, which is set to obtain a desired level of SO_2 removal, could be reduced by perhaps 20 to 25% (making L/mV ~ 1.4 for SO_2). The resultant reduction in pump size and power requirement would have to be weighed against the increase in capital cost of the added absorption trays.

A lower power (and capital) expense would result by eliminating the compressor and directing the flow of the flash gas to either the furnace or to the SO_2 absorber. The feasibility of this option depends in the former case on whether a stable flame can be maintained in the furnace in the presence of additional

inerts (i.e. CO_2), and in the latter case on restrictions on the emission of CO and H_2. Both methods yield low concentrations of H_2S in the exit stack gas, while reducing power comsumption by about 30%.

References

Albertson, N.F. and J.P. McReynolds, *J. Am. Chem. Soc.*, *65*, 1690 (1943).

Crowe, C.M., A.E. Hamielec, T.W. Hoffman, A.I. Johnson, and D.R. Woods, "Chemical Plant Simulation, An Introduction to Computer-Aided Steady-State Process Analysis", Prentice-Hall (1971).

Deal, C.H., and M.N. Papadopoulos, U.S. Patent 3,363,989 (1968).

Demyanovich, R.J., "The Use of Polar Organic Solvents to Control SO_2 Emissions: A Study of Solubilities and Conceptual Designs", Ph.D. Thesis in Chemical Engineering, University of California, Berkeley (1984).

Denn, M.M., W.C. Yu, and J. Wei, *Ind. Eng. Chem. Fund.*, *18*, 286 (1979).

Edmister, W.C., *Hydrocarbon Proc.*, *47 (9)*, 239 (1968).

Ferguson, P.A., "Hydrogen Sulfide Removal from Gases, Air, and Liquids", Noyes Data Corp. (1975).

Grant Chemical, "Glymes, The Grant Family of Glycol Diethers", Grant Chemical Division, Ferro Corp. (1982).

Hill, A.E., and T.B. Fitzgerald, *J. Am. Chem. Soc.*, *57*, 250 (1935).

King, C.J., "Separation Processes", McGraw-Hill Book Co., Second Edition (1980).

Kohl, A.L. and F.C. Riesenfeld, "Gas Purification", Gulf Publishing Company, Third Edition (1979).

Lynn, S., D.W. Neumann, S.F. Sciamanna, and F.H. Vorhis, "A Comparision of the UCB Sulfur Recovery Process with Conventional Sulfur Recovery Technology for Treating Recycle Gas from a Crude Residuum Hydrotreater", Lawrence Berkeley Laboratory, Report No. LBL-21574 (1986).

Newman, J.S., *Ind. Eng. Chem. Fund.*, *7*, 514 (1968).

Perry, R.H. and D. Green, Editors, "Perry's Chemical Engineers' Handbook", McGraw-Hill Book Co., Sixth Edition (1984).

Reid, R.C., J.M. Prausnitz, T.K. Sherwood, "The Properties of Gases and Liquids", McGraw-Hill Book Co., Third Edition (1977).

Renault, P., U.S. Patent 3,170,766 (1969).

Sciamanna, S.F., Ph.D. Thesis in Chemical Engineering, University of California, Berkeley (1986).

Smith, J.M. and H.C. Van Ness, "Introduction to Chemical Engineeering Thermodynamics", McGraw-Hill Book Co., Third Edition (1975).

Sweny, J.W., "High CO_2- High H_2S Removal with SELEXOL Solvent", 59th Annual GPA Convention (1980).

Tanimura, W., U.S. Patent 3,953,586 (1976).

Townsend, F.M., U.S. Patent 2,881,047 (1959).

Urban, P., U.S. Patent 2,987,379 (1961).

Weast, R.C., Editor, "CRC Handbook of Chemistry and Physics", CRC Press, 65th Edition (1984).

Wilhelm, E., R. Battino, and R.J. Wilcock, *Chem. Rev., 77 (2)*, 219 (1977).

List of Variables

A	=	Heat Exchanger Area (m^2)
a_i	=	Stoichiometric Coefficient for Reaction
	=	-2 for H_2S
	=	-1 for SO_2
	=	2 for H_2O
	=	3 for Sulfur
	=	0 for all other components
C	=	Concentration ($kmol/m^3$)
C_p	=	Heat Capacity (kJ/kmol-K)
D_L	=	Average Molar Liquid Density ($kmol/m^3$)
$DELT_{LM}$	=	Log Mean Temperature (K)
F	=	Feed Flow (kmol/s)
h	=	Liquid Enthalpy (kJ/kmol)
H	=	Vapor Enthalpy (kJ/kmol)
H_{RXN}	=	Heat of Reaction (kJ/kmol SO_2)
k_i	=	Equilibrium Constant for Component i
k_2	=	Second Order Reaction Rate Constant (m^3/kmol-s)
L	=	Liquid Flow (kmol/s)
m	=	Flow Rate (kmol/s)
n_L	=	Moles of Limiting Reactant (Eqn. II-7)
QH_j	=	Heat Added or Removed on Stage j (kJ/s)
RS	=	Fraction of Liquid to Sidedraw
T	=	Temperature (K)
t	=	Time (s)
U	=	Overall Heat Transfer Coefficient (kJ/m^2-s-K)
V	=	Vapor Flow (kmol/s)
V_L	=	Volume of Liquid on a Tray (m^3)
x	=	Liquid Mole Fraction
z	=	Feed Mole Fraction

Appendix

Description of the Process Simulation Program

A-I. Simulation Programs

This appendix contains listings of the Fortran-77 and Knowledge Manager Database code (sections A-III and A-IV respectively) for the computer simulation used in the evaluation of the University of California Sulfur Recovery Process. A brief explanation of how to use the simulation program to perform the calculations for a particular process configuration is given in section A-II.

Most of the unit operations are simulated using well known chemical engineering algorithms. A few of the routines requiring more explanation are described below.

A. Main Program - BERKSULF.F77

The main program is actually composed of three routines, BERKRUN, BERKSULF, and BERKSTOP. A normal run of the simulation would begin by invoking BERKRUN, an assembly language program whose sole purpose is to repeatedly start-up BERKSULF. Typically BERKSULF will execute for 5 iterations, terminate, and then be restarted by BERKRUN. This somewhat awkward program chaining is necessitated by memory limitations of the Digital Research Fortran compiler used to compile the simulation.

BERKRUN is aborted when BERKSULF calls BERKSTOP, another assembly language routine, which forces the operating system to terminate execution of BERKRUN. The stopping point is determined by convergence criteria set at compile-time in BERKSULF.

The calculations of the simulation are directed by BERKSULF. After initialization of internal variables, values for composition, flow, temperature, pressure, and phase are read a disk file, either STREAMS.DAT or STREAMS.TMP,

into the array STREAMS. The file STREAMS.DAT contains the set values and initial guesses for the streams, whereas STREAMS.TMP consists of data saved from a previous invocation of BERKSULF. The primary purpose of STREAMS.TMP is to pass variables between restarts of BERKSULF by BERKRUN, although the contents can be used as a set of guesses for a new flow configuration which does not differ greatly from a previous one.

If the data are read from STREAMS.DAT, then the enthalpy for each non-zero stream is calculated. The bubble and dew points are also determined and a warning is issued if the phase does not match that set in the input. Enthalpies and phases are read from the data file when STREAMS.TMP serves as the source.

After STREAMS has been initialized, the program begins to perform the series of unit operations which constitute the flow configuration of interest. The sequence of events is as follows:

1. Read data for the next operation from the disk file STREAMIO.DAT. A line in the file consists of the operation code number (operation to perform) followed by the unit number and a list of the parameters needed and streams involved in the calculation.

2. Vector to the appropriate unit set-up code. Call the indicated subroutine, passing parameters as needed.

3. Update the STREAMS array with data returned from the unit subroutine. Also save any unit information such as power requirement, heat duty, and equipment size.

4. Check for and report potential error conditions. The most common errors are non-convergence in a unit subroutine or that a calculation is not performed due to a zero feed. Exit stream phase conditions (mixed or single) are also noted.

5. Repeat steps 1 through 4 until all operations have been performed. This is indicated by a zero or negative value for the operation code.

6. The normalized values for composition, flow and temperature for all the

streams are compared with those of the previous iteration. If these variables are converged to 1 part in 100, jump to step 9.

7. Apply Wegstein acceleration to the component flows. This is performed by extrapolation (or interpolation) using the results of several direct substitution calculations. If $x_{k+1} = F(x_k)$ is replaced by $x'_{k+1} = F(x_k)$, where F(x) is a function representing the direct substitution calculation, then the accelerated value for x to insert in the equations on the next iteration can be obtained from (Perry, 1984, 2-62)

$$x_{k+1} = q\, x_k + (1-q)\, x'_{k+1} \qquad \text{(A-1)}$$

$$q = (x'_{k+1} - x'_k)/(x'_{k+1} - 2x'_k + x'_{k-1})$$

where the subscript indicates the iteration number and a superscript ' refers to a variable derived from a direct substitution calculation. The accelerated value will generally lead to convergence when $q < 0$.

8. Increment the iteration counter. If the count exceeds the maximum permissible value (either the internal count or the absolute value of a negative operation code obtained in step 5), then jump to step 9. If the iteration count is 5 greater than when the current run of BERKSULF started, write all data to STREAMS.TMP, stop, and wait to be restarted by BERKRUN.

9. At convergence or when the maximum number of iterations has been reached, write all data from STREAMS to the files STREAMS.OLD and BERKOUT.DAT. STREAMS.OLD contains data in the same form as STREAMS.TMP for possible later use, while BERKOUT.DAT is the formatted output detailing all of the stream information. A unit summary consisting of calculated powers, heat duties, and unit sizes is also written.

10. Terminate BERKRUN and stop.

B. Tray Absorber/Stripper/Distillation Routine

A general procedure for a staged absorber is to write material, heat, and

equilibrium equations which are linearized by the Newton-Raphson method. The resulting set of coupled equations is in block tridiagonal form and can be solved by Newman's method using the BAND subroutine (Newman, 1968). Updated variables for the next iteration are obtained from the solution. Initial guesses for the component mole fractions can be made by applying the Thomas method (King, 1980, pgs. 466-472) to an assumed temperature and flow profile based on feed conditions. Convergence of the component mole fractions to one part in 1000 on all stages is usually obtained in fewer than 10 iterations. For the case where H_2S is absorbed from a gas stream into a solution containing SO_2, as can occur in this process, the calculations must account for the reaction between two simultaneously absorbing/desorbing species. Under these conditions a reaction term is added to the energy and material balances to account for the disappearance of H_2S and SO_2 and the production of H_2O and sulfur. This is implemented by treating each stage as a CSTR with the reaction evaluated at outlet conditions.

Including the reaction terms from the mass and energy balances explicitly in the linearization of the equations yields a problem which is calculationally unstable and often non-convergent. This difficulty is overcome by treating the reaction terms in the equations as constants (i.e. as pseudo-sidedraws for H_2S and SO_2 and as pseudo-feeds for H_2O and Sulfur) for the solution of the general problem at each iteration. To account explicitly for the presence of the reaction, the mass balance equations for H_2S and SO_2 are linearized in mole fraction <u>only</u> and a converged solution for these components is obtained on all stages. These values then are used to calculate the constant reaction terms in the general problem. The rest of the calculations proceed as for the solution to the problem in which no reaction is present.

The equations describing the staged column calculations for absorption, stripping, and distillation are given in Table A-I. Stepwise, a solution to the

Table A-I

Equations Describing an Equilibrium Stage j

Component Mass Balances (A-2): (1 Eqn. for each component i)

$$-V_{j-1}k_{j-1,i}x_{j-1,i} + (V_j k_{j,i} + (L_j/(1-RS_j)))x_{j,i} - L_{j+1,i}$$
$$= (F(z_i)_F)_j + a_i k_2 V_L (x_{H2S} D_L)(x_{SO2} D_L)$$

Heat Balance (A-3):

$$-V_{j-1}H_{j-1} + V_j H_j + L_j h_j - L_{j+1} h_{j+1}$$
$$= (Fh_F)_j - \sum_i H_{RXN} a_i k_2 V_L (x_{H2S} D_L)(x_{SO2} D_L) + QH_j$$

Equilibrium Expression (A-4):

$$\sum_i [(k_{j,i} - 1)x_{j,i}] = 0$$

Mole Fraction (A-5):

$$\sum_i [x_{j,i}] = 0$$

where

a_i	= Stoichiometric Coefficient for Reaction
	= -2 for H_2S
	= -1 for SO_2
	= 2 for H_2O
	= 3 for Sulfur
	= 0 for all other components
D_L	= Average Molar Liquid Density (kmol/m^3)
F	= Feed Flow (kmol/s)
h	= Liquid Enthalpy (kJ/kmol)
H	= Vapor Enthalpy (kJ/kmol)
H_{RXN}	= Heat of Reaction (kJ/kmol SO_2)
k_i	= Equilibrium Constant for Component i
k_2	= Second Order Reaction Rate Constant (m^3/kmol-s)
L	= Liquid Flow (kmol/s)
QH_j	= Heat Added or Removed on Stage j (kJ/s)
RS	= Fraction of Liquid to Sidedraw
V	= Vapor Flow (kmol/s)
V_L	= Volume of Liquid on a Tray (m^3)
x	= Liquid Mole Fraction
z	= Feed Mole Fraction

problem is obtained as follows:

1. Set initial guesses for the column. The liquid and vapor flows, the temperatures, and the compositions on all stages are estimated from the cummulative effects of the feeds to the column.

2. Perform property calculations (equilibrium constants, derivatives, etc.).

3. Calculate x_i, y_i on all stages by the Thomas method.

4. Perform property calculations under the new conditions.

5. The component mass balance equations for H_2S and SO_2 (A-2) are linearized in mole fraction and solved. The term $a_i k_2 V_L (x_{H2S} D_L)(x_{SO2} D_L)$, which represents the effect of the chemical reaction, is included in the linearization. Convergence of the mole fractions on all stages to 1 part in 1000 is generally obtained in less than 30 iterations.

6. Equations A-2, A-3, A-4, and A-5 are linearized by Newton-Raphson in T, V, L and all x_i and solved by Newman's method. The reaction term is treated as a constant value at the conditions obtained in step 5.

7. Iterate to step 4 until all x_i are converged to 1 part in 1000.

C. Heat Exchanger Subroutine

The heat exchanger subroutine simulates the operation of a single pass countercurrent heat transfer device. Since the possibility of flashing of light gasses from solvent upon heating exists in the process under consideration, an algorithm was devised which correctly accounts for fluid phase changes. The equations describing this unit are given in Table A-II. Equations A-6 and A-7, which are linearized in the exit temperatures of the hot and cold streams, are solved by Newton-Raphson as follows:

1. Estimate $T_{c\text{-}exit}$ and $T_{h\text{-}exit}$ using the exact solution to the heat exchanger problem for the case of no phase change.

2. Perform isothermal flash calculations on the two fluids at their exit

Table A-II

Equations Describing a Heat Exchanger

Heat Balance Equations (A-6 and A-7):

$$F_1 = (H_{h-in} - H_{h-exit}) - UA\ DELT_{LM} = 0$$

$$F_2 = (H_{h-in} - H_{h-exit}) - (H_{c-exit} - H_{c-in}) = 0$$

Exit Temperatures (No Phase Change) (A-8 and A-9):

$$T_{c-exit} = \frac{T_{h-in}(1-B_{NPC}) + T_{c-in}B_{NPC}(1-A_{NPC})}{(1-A_{NPC}B_{NPC})}$$

$$T_{h-exit} = T_{h-in} + (T_{c-in} - T_{h-in})(A_{NPC}(1-B_{NPC}))/(1-A_{NPC}B_{NPC})$$

where

$$A_{NPC} = \frac{\sum_i (m_{h,i}C_{p-h,i})}{\sum_i (m_{c,i}C_{p-c,i})}$$

$$B_{NPC} = EXP[UA(1/\sum_i (m_{c,i}C_{p-c,i}) - 1/\sum_i (m_{h,i}C_{p-h,i})]$$

A	=	Heat Exchanger Area (m^2)
C_p	=	Heat Capacity (kJ/kmol-K)
$DELT_{LM}$	=	Log Mean Temperature (K)
m	=	Flow Rate (kmol/s)
T	=	Temperature (K)
U	=	Overall Heat Transfer Coefficient (kJ/m^2-s-K)

Subscripts:

c	=	cold stream
h	=	hot stream
in	=	inlet
exit	=	outlet
i	=	component i

temperatures to obtain the correct exit enthalpies, accounting for possible phase changes. Calculate values for F_1 and F_2.

3. Calculate the derivatives of F_1 and F_2 with respect to the outlet temperatures. The derivatives are determined numerically by performing isothermal flash calculations at incremental increases in temperature.

4. The F values and the derivatives are used to calculate new values for the exit temperatures.

5. Repeat steps 2 through 4 until convergence is obtained.

D. Thermodynamics Subroutine

The thermodynamics subroutine supplies the enthalpies, equilibrium constants, and derivatives of H and k with respect to temperature needed in the simulation calculations. For the implementation covered in section A-III, the bulk solvent is assumed to be tetraglyme. References for data used in the calculations can be found with the source code. A brief discussion of the methods of calculation is given below.

Vapor-phase enthalpies are calculated from heat capacity data referenced to 298 K. To determine liquid-phase enthalpies, the heat of vaporization is subtracted from the vapor-phase values. Heats of solution, when available, are used in place of the heats of vaporization when the solvent is present. The enthalpy of formation is included in all enthalpy calculations, since a chemical reaction is present in the process.

Equilibrium constants are estimated from temperature dependent Henry's law constants when solvent and/or water are present, and by Raoult's law using the Antoine equation for vapor pressure otherwise. Solvent properties are always estimated from vapor pressure data. Mixtures of solvent and water are dealt with by employing the empirical mixing "rule" of a volume fraction average of the Henry's law values in the two liquids. Since the mass densities of tetraglyme and

water are nearly equal, the average weight fraction can be used in place of the volume fraction. The expression describing the Henry's law constant for a component in the mixture is

$$H_{mix} = DEXP\ [(W_s\ LN(H_s + W_w\ LN(H_w)/(W_s + W_w)] \qquad (A\text{-}10)$$

Note that volume rather than mole fraction is used in the analysis because it better accounts for the difference in molecular size (see Sciamanna, 1986).

At pressures greater than 500 kPa, corrections for enthalpies and k values are derived by assuming that the Redlich-Kwong equation of state applies. Enthalpy departures and fugacity coefficients are calculated based on this model.

A-II. Simulation Usage

The simulation was created to permit rapid assessment of process alternatives by calculation of material and energy balances. To evaluate a particular configuration, a detailed flowsheet must be drawn. All units, even minor ones such as mixing and splitting vessels, need to be included in the drawing. The diagram establishes the unit operations to be performed and the interconnection between devices.

Since processes often contain one or more recycle streams, the ordering of unit calculations requires intelligent use of the process arrangement. In fact the flow of information used in calculations is sometimes different from the flow of mass. When a downstream unit produces an output required in a particular unit, it may be convenient to calculate the downstream unit first, even though this is not the order in which the unit operations would normally be thought of as occurring.

Frequently the user will need to "guess" the flow, composition, temperature, etc. of a stream in order to provide an initial input to an operation. For example, if an input stream to unit one is a recycle from unit two, which derives its output from unit one, then the input to one of the units must be specified before the first iteration is performed. Often the variables representing these streams are set

based on knowledge of the process conditions. For minor input streams such variables often can be initialized to zero.

The process considered in the present work is sufficiently simple that ordering of units could be performed by inspection. For more complex flow configurations detailed procedures for determining calculation order and stream specifications, referred to as decomposition analysis, are discussed elsewhere (see for example Crowe et al., 1971).

Figure A-I is a presentation of the type of information needed to set up the simulation. On this block flow diagram, which is a schematic representation of the process of Fig. III-2, the unit numbers (below or inside a unit block) reflect the order in which the unit calculations are to occur. Set and guessed streams have also been indicated. Although stream numbers have been omitted to reduce clutter in the drawing, in practice all the streams would be numbered.

The block flow diagram, which specifies the order of calculations, interconnection of units, and the "torn" streams in the process simulation, combined with detailed information about process units (i.e number of stages in an absorber, set stream flows, set temperatures, etc.) and with a knowledge of process requirements, serves as a guide to creating the simulation input. Two disk files, STREAMS.DAT and STREAMIO.DAT, already described above, provide the data needed to perform the process calculations.

Although these files can be produced using any text editor by applying the specifications given in the program code (see A-III), the procedure is extremely tedious and time consuming. To ease data entry, a set of programs, written in the KnowledgeMan Data Base Management System, has been provided. The program code is in section A-IV. The routines are entirely menu driven and produce the files required to run the simulation.

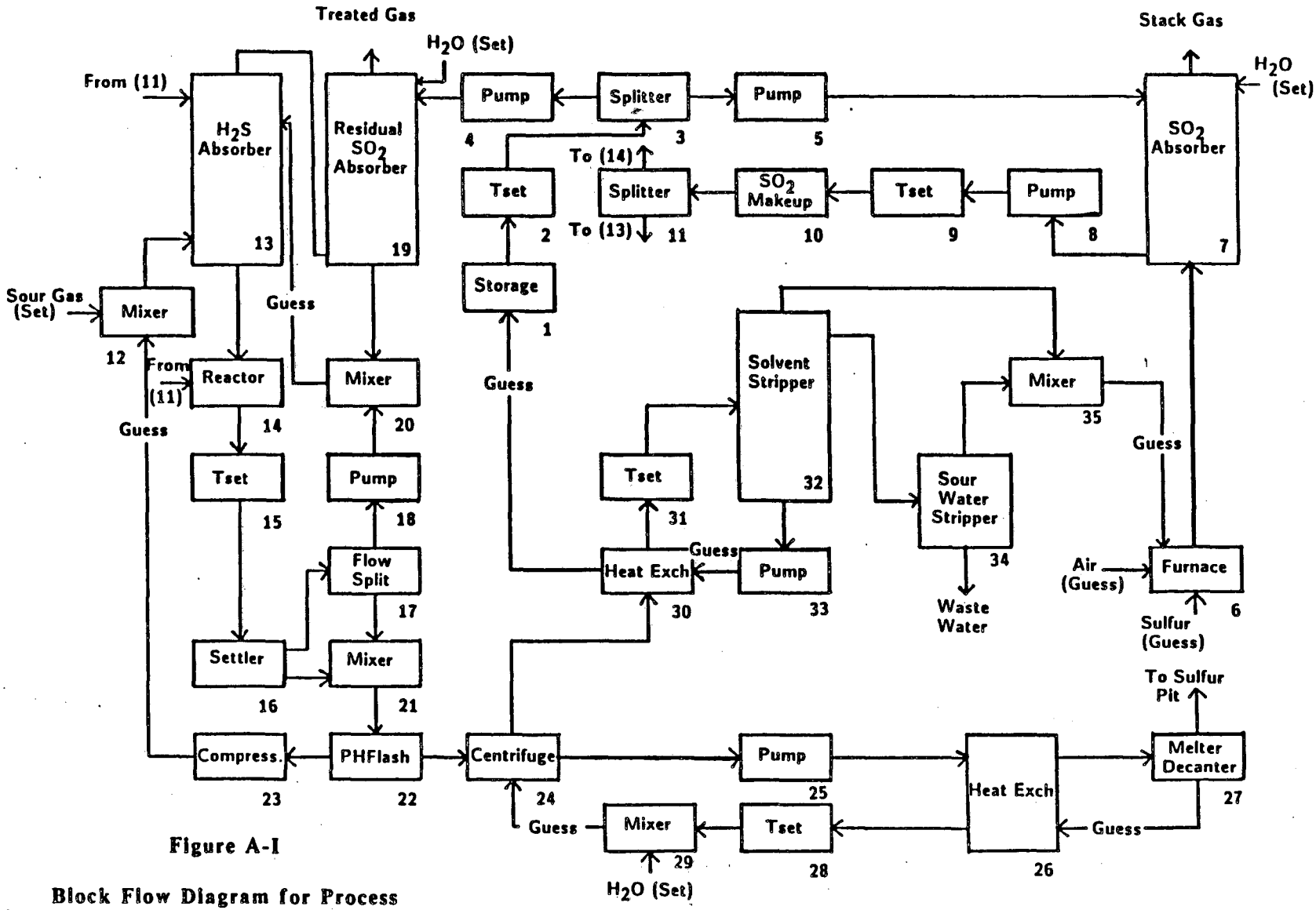


Figure A-I

Block Flow Diagram for Process

A-III. Simulation Code

This section contains the computer code for the UCBSRP simulation. All of the routines, except the assembler coded BERKRUN and BERKSTOP were written in Digital Research Fortran-77 Version 4.1 for Concurrent CPM. The simulation was executed on a Compupro computer system with a 10 Mhz. 286/287 processor and 768k of main memory. A typical run of a flow configuration of the complexity indicated in section A-II required 5 to 8 hours to produce a converged solution.

Numerous comments have been included in the code to make it more readable. For detailed descriptions of the main program, the absorber/stripper/distillation routine, the heat exchanger and the thermodynamics calculations see section A-I. The remainder of the routines should be self explanatory. Cited literature references can be consulted for further information on some of the algorithms.

```
;SIMULATION DRIVER PROGRAM
;CONTINUALLY EXECUTES BERKSULF.CMD
;WHEN BERKSULF DETERMINES THAT CONVERGENCE HAS OCCURRED
; IT ABORTS THIS ROUTINE BY A CALL TO THE BERKSTOP
; SUBROUTINE
;
;THIS PROGRAM IS INVOKED BY:
; BERKRUN
;   OR
; BERKRUN X
;  WHERE X IS AN OPTIONALLY SPECIFIED DRIVE ON WHICH
;   BERKSULF CAN BE FOUND

;WRITTEN FOR DRI ASM86 UNDER CONCURRENT CPM 4.1
; LAST UPDATE: 04/21/86
;
;EQUATES
GETDRV          EQU     025             ;RETURN DEFAULT DRIVE
SETPRIOR        EQU     145             ;SET PRIORITY
CONATTACH       EQU     146             ;ATTACH CONSOLE
CLICMD          EQU     150             ;SEND CLI COMMAND

DBUF            EQU     080H            ;DEFAULT COMMAND TAIL BUFFER

CCPM            EQU     224             ;SYSTEM ENTRY

        CSEG
        ORG     0

START:  MOV     BX,OFFSET DBUF          ;POINT TO DEFAULT BUFFER
        CMP BYTE PTR [BX],0             ;CHECK FOR DRIVE SPECIFICATION
        JZ      NO_TAIL
START1: INC     BX                      ;SKIP OVER NULLS, SPACES AND TABS
        CMP BYTE PTR [BX],20H
        JE      START1
        CMP BYTE PTR [BX],09H
        JE      START1
```

```
        CMP BYTE PTR [BX],00
        JE      START1
        MOV     AL,[BX]                 ;GET DRIVE DESIGNATION
        MOV     DRV,AL
        JMPS    SEND_COM
NO_TAIL:MOV     CL,GETDRV               ;GET DEFAULT DRIVE IF NONE SPECIFIED
        INT     CCPM
        ADD     AL,41H                  ;MAKE DRIVE ASCII
        MOV     DRV,AL
SEND_COM:
        MOV     CL,CLICMD
        MOV     DX,OFFSET BERKSULF      ;EXECUTE BERKSULF
        INT     CCPM
        MOV     CL,SETPRIOR             ;SET PRIORITY HIGHER THAN SHELL
        MOV     DL,190                  ; SO WE REGAIN CONSOLE CONTROL
        INT     CCPM                    ; AFTER BERKSULF STOPS
        MOV     CL,CONATTACH            ;ATTACH TO CONSOLE
        INT     CCPM
        JMPS    START                   ;DO FOREVER

        DSEG
        ORG     100H
BERKSULF DB     0
DRV     DB      'A'
        DB      ':BERKSULF',0

        END
```

```
      PROGRAM BERKSULF

C  LAST UPDATE 6/23/86

C*****************************************************************
C  DESCRIPTION OF MAJOR VARIABLES:
C
C  STREAM = COMPOSITION, FLOW, TEMPERATURE, PRESSURE, AND
C    ENTHALPY, PHASE FOR ALL STREAMS IN THE PROCESS. (CURRENT ITERATION)
C  STROLD = COMPOSITION, FLOW, TEMPERATURE FOR ALL STREAMS
C    FROM THE PREVIOUS ITERATION
C
C  UNIT   = UNIT NUMBER, POWER, HEAT DUTY, AND SIZE
C   POWER IN KW, HEATS IN KW/S, SIZE IN GAL.
C
C  FLOW   = COMPONENT FLOWS FOR THE PREVIOUS TWO AND CURRENT
C    ITERATION, USED IN WEGSTEIN ACCELERATION (SEE BELOW)
C
C  X, Y, AND HX ARE USED IN STREAM INITIALIZATION
C   X     = LIQUID MOLE FRACTIONS
C   Y     = VAPOR MOLE FRACTIONS
C   HX    = COMPONENT ENTHALPIES
C
C  FEED, ZFEED, AND QI ARE USED FOR PASSING FEED CONDITIONS TO THE
C   ABSORBER ROUTINE
C   FEED  = STAGE #, FLOW, TEMPERATURE, AND ENTHALPY OF FEED
C   ZFEED = COMPOSITION
C   QI    = INTERCOOLER STAGE NUMBER(S) AND HEAT REMOVED
C
C  MOLWT  = MOLECULAR WEIGHTS FOR ALL COMPONENTS (FOR COMPONENT
C   DESIGNATIONS SEE THERMODYNAMICS SUBROUTINE)
C
C  ALPHA  = ARRAY OF STOICHIOMETRIC COEFFICIENTS FOR THE REACTION
C   BETWEEN H2S AND SO2 TO FORM SULFUR AND WATER
C
C  ABND, BBND, ETC. ARE THE ARRAYS THAT APPEAR IN THE BAND
```

```
C   SUBROUTINE USED IN ABSORBER CALCULATIONS
C
C  STRI AND STRO ARRAYS ARE USED TO PASS STREAM INFORMATION TO
C   AND FROM VARIOUS SUBROUTINES
C   THE CONTENTS = COMPOSITION, FLOW,T,P,ENTHALPY,PHASE
C
C*****************************************************************
C NOTES: THROUGHOUT THE SIMULATION THE FOLLOWING VALUES APPLY FOR
C  THE PHASE TYPE
C    1 = LIQUID
C    2 = VAPOR
C    3 = MIXED LIQUID AND VAPOR
C    4 = LIQUID WITH SOLID SULFUR PRESENT
C    5 = VAPOR WITH SOLID SULFUR PRESENT
C    6 = MIXED LIQUID AND VAPOR WITH SOLID SULFUR PRESENT
C
C FLOWS IN KMOLE/S
C TEMPERATURES IN K
C PRESSURE IN KPA
C ENTHALPIES IN KJ/KMOLE
C
C*****************************************************************
C DESCRIPTION OF OPCODE OPERATION MATRIX
C
C THE DRIVER INFORMATION FOR THE SIMULATION PROGRAM IS KEPT AS A
C   SERIES OF DATA SETS IN THE FILE STREAMIO.DAT
C EACH ENTRY IN THE DRIVER FILE CONSISTS OF THE FOLLOWING ENTRIES:
C       OPCODE#, UNIT#, DATA1, DATA2, ....., DATAN WHERE N CAN
C          RANGE TO 30 ITEMS
C MOST OF THE DATA ITEMS ARE EITHER STREAM NUMBERS OR PROPERTY VALUES
C   SUCH AS PRESSURE, ETC.

C THE PROGRAM READS THE ENTRIES SEQUENTIALLY, CARRYING OUT EACH
C   OPERATION AS REQUESTED
C WHEN ALL OPERATIONS HAVE BEEN COMPLETED (OPCODE=0) THEN A
C   CONVERGENCE CHECK IS PERFORMED.
C THIS CONTINUES UNTIL CONVERGENCE IS OBTAINED OR A MAXIMUM
```

```
C   NUMBER OF ITERATIONS HAS OCCURRED.
C NOTE THAT THIS AMOUNTS TO A DIRECT SUBSTITUTION CONVERGENCE SCHEME.

C THE AVAILABLE OPERATION CODES AND THE ASSOCIATED DATA VALUES
C   ARE LISTED BELOW -- NOTE THAT ALL VALUES FOR A GIVEN OPERATION
C   MUST BE SUPPLIED!
C   ABBREVIATIONS USED:

C       FLOW    = TOTAL STREAM FLOW (KMOLE/S)
C       FLOWI   = FLOW OF COMPONENT I (KMOLE/S)
C       FS      = FRACTION OF LIQUID TO SIDE DRAW FROM CONDENSER IN
C                   DISTILLATION
C       ICOMP   = COMPONENT NUMBER
C       IOFLAG  = IF EQUALS 1 USE OLD PROFILE ON FIRST ITERATION
C                 IF EQUALS 0 GENERATE NEW INITIAL GUESSES
C       P       = PRESSURE (IN UNIT OR AT OUTLET) (KPA)
C       QR      = HEAT REMOVED (KJ)
C       QRE     = HEAT ADDED TO REBOILER (KJ/S)
C       QI#     = HEAT ADDED TO STAGE (KJ/S)
C       QIN     = HEAT ADDED (KJ)
C       RHOM    = AVERAGE MOLAR LIQUID DENSITY (KMOLE/M^3)
C       R2CONST = SECOND ORDER RATE CONSTANT (H2S - SO2 RXN) (M^3/KMOLE-S)
C       STI     = STREAM  IN
C       STO     = STREAM  OUT
C       T(OUT)  = TEMPERATURE (IN UNIT OR AT OUTLET) (K)
C       TC      = TEMPERATURE OF CONDENSER (K)
C       UA      = HEAT TRANSFER COEFFICIENT * AREA (KJ/S-K)
C       VLIQT   = VOLUME OF LIQUID ON A TRAY (M^3)
```

OPERATION CODE#	OPERATION	DATA I/O (I.E. DATA IN OPCODE MATRIX)
1	PLATE ABSORBER/ DISTILLATION	UNIT, NSTAGES, P, VLIQT, RHOM, R2CONST, FEED1, STAGEF1, FEED2, STAGEF2,..,FEED5, STAGEF5, STRO VAPOR, STRO LIQUID, QRE, QI1, STAGEQI1, QI2, STAGEQI2, TC, FS, STRO LIQUID CONDENSER, IOFLAG
2	FSPLIT	UNIT, STRI, STRO1, STRO2, FLOW TO 1
3	PHFLASH	UNIT, STRI, P, STRO LIQUID, STRO VAPOR
4	HEAT EXCHANGER	UNIT, STRI1 (HOT), STRI2 (COLD), STRO1, STRO2, UA
5	REBOILER	UNIT, STRI, STRO LIQUID, STRO VAPOR, P, T
6	CONDENSER	UNIT, STRI, STRO, P
7	MIXER	UNIT, STRI1, STRI2, STRO, P
8	SPLITTER	UNIT, STRI, STRO1, STRO2, FRACT TO STRO1
9	PUMP	UNIT, STRI, STRO, P AT OUTLET, RHOM
10	REACTOR	UNIT, STRI1, STRI2, STRO, P, Q REMOVED, FRACT. CONVERSION, R2CONST
11	SULFUR SATURATOR	UNIT, STRI STRO, T, P
12	SULFUR SETTLER	UNIT, STRI, STRO1 (SULFUR), STRO2 (SOLVENT), WT% S IN SULFUR OUT, P
13	SULFUR WASHER	UNIT, STRI1 (SULFUR), STRI2 (WATER), STRO1 (WASH), STRO2 (SULFUR), WT% S IN SULFUR, P
14	MELTER/DECANTER	UNIT, STRI1, STRO1 (WATER), STRO2 (SULFUR), T, P
15	FURNACE	UNIT, STRI1 (SULFUR), STRI2 ("JUNK"), STRI3 (AIR), SO2REQ, STRO1, TOUT, POUT (NOTE THAT THE COMPOSITION,TEMPERATURE, PRESSURE AND PHASE MUST BE SET FOR THE AIR STREAM)

16	COMPRESSOR	UNIT, STRI1, STRO1, POUT
17	SULFUR CENTRIFUGE	UNIT, STRI1, STRI2, STRO1 (SULFUR SLURRY), STRO2 (SOLVENT/WATER), P, WT%1, WT%2 (WHERE WT% VALUES ARE FRACTION OF SULFUR IN SULFUR STREAMS IN EACH SETTLING STEP)
18	PARTIAL CONDENSER	UNIT, STRI1, PCOND, TCOND, STRO LIQUID, STRO VAPOR
19	HEATER	UNIT, STRI, STRO, QIN
20	STORAGE	UNIT, STRI, STRO, FLOWI, ICOMP, TOUT (FLOWI SET, OTHER FLOWS SCALED AT SAME COMPOSITION)
21	CRYSTAL	UNIT, STRI, STRO
22	SULSPLIT	UNIT, STRI, STRO1, STRO2, FRACT1
23	MAKEUP	UNIT, STRI, STRO, FLOWI, ICOMP (FLOWI SET, OTHER FLOWS UNCHANGED)
24	TSET	UNIT, STRI, STRO, TSET
25	HCSPLIT	UNIT, STRI, STRO1 (HYDROCARBONS), STRO2 (OTHER)

```
IMPLICIT REAL*8(A-H,O-Z)

REAL*8 STREAM(100,25), STROLD(100,23), UNIT(80,4), OPCODE(30)
REAL*8 FLOW(100,20,3)
REAL*8 FEED(5,4),ZFEED(5,20)
REAL*8 X(20),Y(20),HX(20)
REAL*8 MOLWT(20)
REAL*8 STRI1(25),STRI2(25),STRO1(25),STRO2(25),STRF(25)
REAL*8 STRI3(25),STRI4(25),STRO3(25),STRO4(25)

REAL*8 QI(2,2)

LOGICAL INEW
```

```
      REAL*8 ABND(18,18),BBND(18,18),CBND(18,25),DBND(18,37)
      REAL*8 GBND(18),XBND(18,18),YBND(18,18)

      REAL*8 EBND(18,19,25)

      INTEGER*2 ALPHA(20)

      INTEGER*2 ERROR

      COMMON/ALPH/ALPHA
      COMMON/MOLEC/MOLWT
      COMMON/BND/ABND,BBND,CBND,DBND,GBND,XBND,YBND,IT,NJ
      COMMON/EEE/EBND
      COMMON/WEGS/FLOW

C SET TEMPORARY INTERNAL ITERATION LIMIT
      ITERMAX=10

C NUMBER OF COMPONENTS IS 20
      NC=20

C SET VALUES FOR CONVERGENCE PRECISION AND ITERATION COUNT
C MAXIMUM NUMBER OF ITERATIONS = IMAX IN MAIN, IMAXS FOR SUBROUTINES
C                              = ICMAX FOR A SINGLE EXECUTION OF BERKSULF
C                                (UNDER CONTROL OF BERKRUN)
C CONVERGENCE CRITERION = EPSILON IN MAIN
C                       = EPS FOR SUBROUTINES
      IMAX=50
      IMAXS=30
      ICMAX=5
      EPSILON=1.0D-2
      EPS=1.D-3
C READ REACTION STOICHIOMETRY VALUES
      OPEN(2,FILE='ALPHA.DAT')
      READ(2,*) (ALPHA(I),I=1,NC)
```

```
C READ ALL MOLECULAR WEIGHT VALUES
      OPEN(3,FILE='MOLEC.DAT')
      READ(3,*) (MOLWT(I),I=1,NC)

C TRY TO READ AN OLD STREAMS MATRIX. THIS WOULD EXIST EITHER BECAUSE
C WE RAN THE CALCULATIONS BEFORE OR ARE SOMEWHERE IN THE MIDDLE OF A
C CALCULATION REQUIRING MULTIPLE INVOCATIONS OF THIS PROGRAM BY BERKRUN
      INEW=.TRUE.
      OPEN(4,FILE='STREAMS.TMP',STATUS='OLD',ERR=8)
C READ OLD ITERATION COUNT AND SHOW THAT WE FOUND THIS FILE
      READ(4,*) ITER
      I=1
5     READ(4,*) (STREAM(I,J),J=1,NC)
      READ(4,*) (STREAM(I,J),J=NC+1,NC+5)

      IF(STREAM(I,NC+1).NE.-1.0D0) THEN
              I=I+1
              GOTO 5
      ENDIF

      INEW=.FALSE.

      GOTO 6

C READ ALL INITIAL STREAM VALUES. READ UNTIL FLOW=-1.0
C ALSO INITIALIZE THE ITERATION COUNTER
8     OPEN(4,FILE='STREAMS.DAT')
      ITER=1

9     I=1
10    READ(4,*) (STREAM(I,J),J=1,NC+5)

      IF(STREAM(I,NC+1).NE.-1.0D0) THEN
              I=I+1
              GOTO 10
      ENDIF
```

```
6       NSTREAMS=I-1

        WRITE(6,43333) NSTREAMS
43333   FORMAT(X,' NSTREAMS=',I5)

C ZERO ACCELERATOR MATRIX AND SHOW THAT FRESH VALUES MUST BE GENERATED
        ICWEGS=1
        DO 11 I=1,NSTREAMS
        DO 11 J=1,NC
        DO 11 K=1,3
11      FLOW(I,J,K)=0.D0

C READ OLD FLOW MATRIX FOR ACCELERATOR IF NEEDED
        IF(.NOT.INEW) THEN
         OPEN(9,FILE='WEGST',ERR=14)
         DO 12 I=1,NSTREAMS
         DO 12 J=1,NC
12       READ(9,*,ERR=14,END=14) (FLOW(I,J,K), K=1,3)
C SHOW THAT AN ACCEPTABLE OLD FLOW MATRIX HAS BEEN READ
        ICWEGS=5
        ENDIF

C CLOSE FILES
14      CLOSE(9,STATUS='DELETE')
        CLOSE(1)
        CLOSE(2)
        CLOSE(3)
        CLOSE(4)

C COPY DATA TO CONVERGENCE MATRIX
        DO 15 I=1,NSTREAMS
        DO 15 J=1,NC+3
15      STROLD(I,J)=STREAM(I,J)

C OPEN OUTPUT DATA FILE
        OPEN(1,FILE='BERKOUT.DAT')
C NOW DELETE OLD DATA FILE
```

```
      CLOSE(1,STATUS='DELETE')

C NOW OPEN IT FOR THIS RUN
      OPEN(1,FILE='BERKOUT.DAT')

C CALCULATE ALL INITIAL ENTHALPIES FOR STREAMS WITH NON-ZERO FLOW
C ALSO CALCULATE BUBBLE AND DEW POINTS AND MAKE SURE ENTERED PHASE
C IS CORRECT. PRINT WARNINGS FOR INCORRECT PHASES AND FOR MIXTURES
C OF LIQUID AND VAPOR
C IF THIS IS AN OLD STREAMS MATRIX (SEE ABOVE) THEN ALREADY HAVE
C ENTHALPIES, ETC.

      IF(INEW) THEN
      DO 20 I=1,NSTREAMS
      WRITE(6,19950) I
      IF(STREAM(I,NC+1).EQ.0.0D0) GOTO 20
      DO 30 J=1,NC
      X(J)=STREAM(I,J)
30    Y(J)=X(J)
      IPHASE=INT(STREAM(I,NC+5))
      TFEED=STREAM(I,NC+2)
      CALL HEAT(0,IPHASE,NC,TFEED,STREAM(I,NC+3),X,HX,STREAM(I,NC+4))
      CALL BUBDEW(NC,STREAM(I,NC+2),X,Y,TBUB,TDEW,STREAM(I,NC+3),
     1 EPS,2)

      IF(TFEED.LT.TBUB) THEN
C ALL LIQUID
              IF(IPHASE.NE.1) THEN
                      STREAM(I,NC+5)=1
                      WRITE(1,20010) I
                      WRITE(6,20010) I
              ENDIF
      GOTO 20
      ENDIF

      IF(TFEED.GT.TDEW) THEN
C ALL VAPOR
```

```
            IF(IPHASE.NE.2) THEN
                    STREAM(I,NC+5)=2
                    WRITE(1,20020) I
                    WRITE(6,20020) I
            ENDIF
      GOTO 20
      ENDIF

C MUST BE MIXED VAPOR AND LIQUID, ISSUE A WARNING, SET PHASE TO MIXED
      WRITE(1,20030) I
      WRITE(6,20030) I
      STREAM(I,NC+5)=3.0

20    CONTINUE
     ENDIF

C OPEN OPERATION CODE FILE
      OPEN(2,FILE='STREAMIO.DAT')

C ZERO LOCAL ITERATION COUNTER
      ICOUNT=1

C MAIN PROGRAM LOOP -- CONTINUE UNTIL CONVERGED OR ITER>IMAX
40    REWIND 2
      NUNITS=0

C OPERATION CODE LOOP
50    READ(2,*) (OPCODE(I),I=1,30)

C QUIT IF ALL OPERATIONS HAVE BEEN PERFORMED
      IF(OPCODE(1).LE.0.0D0) GOTO 15000
      NUNITS=NUNITS+1

C SET CURRENT UNIT NUMBER INTO UNIT MATRIX (UNIT NUMBER CAN BE FRACTIONAL)
      UNIT(NUNITS,1)=OPCODE(2)

C SHOW THAT AN OPERATION IS IN PROGRESS
```

```
      WRITE(6,20000) ITER,OPCODE(2)

C SEND TO CORRECT UNIT OPERATION
      GOTO (100,200,300,400,500,600,700,800,900,1000,1100,1200,
     1 1300,1400,1500,1600,1700,1800,1900,2000,2100,2200,2300,2400,
     2 2500) INT(OPCODE(1))

C IF ERROR
      GOTO 10000

C*********************************************************************
C PERFORM PLATE ABSORBER CALCULATION
C FIRST ZERO THE FEED MATRICES
100   DO 130 I=1,5
      DO 120 J=1,4
120   FEED(I,J)=0.000
      DO 130 J=1,NC
130   ZFEED(I,J)=0.000

C SET NUMBER OF STAGES AND PRESSURE
      NSTAGES=INT(OPCODE(3))
      P=OPCODE(4)

C SET LIQUID VOLUME ON A TRAY, AVERAGE MOLAR LIQ. DENSITY, AND RATE CONSTANT
      VLIQT=OPCODE(5)
      RHOM=OPCODE(6)
      R2CONST=OPCODE(7)

C SET UP THE FEED MATRICES FOR NON-ZERO FEED STREAMS
      J=-2

      DO 140 I=1,5
      J=J+2
      IF(OPCODE(J+8).EQ.0.0D0) GOTO 140

C SET INDEX FOR CURRENT FEED
      ICODE=INT(OPCODE(J+8))
```

```
C GET VALUES FOR STAGE#, FLOW, TEMP, AND ENTHALPY
        FEED(I,1)=OPCODE(J+9)
        FEED(I,2)=STREAM(ICODE,NC+1)
        FEED(I,3)=STREAM(ICODE,NC+2)
        FEED(I,4)=STREAM(ICODE,NC+4)
C SET ALL Z VALUES
        DO 140 K=1,NC
        ZFEED(I,K)=STREAM(ICODE,K)
140     CONTINUE

C SET REBOILER DUTY, INTERSTAGE HEATER/COOLER VALUES, CONDENSER TEMP
C AND FRACTION SIDEDRAW (IF PRESENT)

        QR=OPCODE(20)
        QI(1,1)=OPCODE(22)
        QI(1,2)=OPCODE(21)
        QI(2,1)=OPCODE(24)
        QI(2,2)=OPCODE(23)
        TC=OPCODE(25)
        FS=OPCODE(26)

C CHECK FOR USE OF OLD PROFILE ON FIRST ITERATION
        IOFLAG=INT(OPCODE(28))

C CHECK FOR ALL ZERO ENTRY FLOWS
C IF NO FLOW IN GAS FEED AND NO FLOW IN ANY LIQUID FEED THEN ABORT
C THE CALCULATION
        DO 145 I=1,5
        IF(FEED(I,2).NE.0.0D0) GOTO 147
145     CONTINUE
        GOTO 99910

C SET NUMBER OF COMPONENTS FOR SUBROUTINE BASED ON MAX. NUMBER OF NON-ZERO
C ENTRIES FOR THE FEED MOLE FRACTIONS
C MAX NUMBER THAT THE SUBROUTINE CAN HANDLE IS 15
147     NCSUB=0
```

```
      DO 148 I=1,5
      DO 148 IX=1,15
      IF(ZFEED(I,IX).NE.0.0D0) THEN
              IF(NCSUB.LT.IX) THEN
              NCSUB=IX
              ENDIF
      ENDIF
148   CONTINUE


      WRITE(6,12457) NCSUB
12457 FORMAT(X,'NCSUB=',I3)


C CALL THE ABSORBER ROUTINE
      CALL ABSORBER(NSTAGES,NC,P,FEED,ZFEED,VLIQT,RHOM,R2CONST,
    1 QR,QI,TC,FS,STRO1,STRO2,STRO3,IMAXS,NCSUB,OPCODE(2),IOFLAG,ITER)

C SET RETURN VALUES INTO APPROPRIATE MATRIX
C SET INDEXES FOR VAPOR AND LIQUID RETURN
      IRET1=INT(OPCODE(18))
      IRET2=INT(OPCODE(19))
      DO 150 I=1,NC+5
      STREAM(IRET1,I)=STRO1(I)
150   STREAM(IRET2,I)=STRO2(I)

C SET SIDEDRAW IF PRESENT
      IF(FS.NE.0.D0) THEN
       IRET3=INT(OPCODE(27))
       DO 152 I=1,NC+5
152    STREAM(IRET3,I)=STRO3(I)
      ENDIF

C JUMP TO ERROR CHECK ROUTINE
      GOTO 10000
```

```
C*********************************************************************
C FSPLIT - SPLIT THE INLET STREAM BASED ON FLOW
C
C SET INLET STREAM
200     IN1=INT(OPCODE(3))

        DO 210 I=1,NC+5
210     STRI1(I)=STREAM(IN1,I)

C SET FLOW TO SEND TO OUTLET STREAM 1
        FLOWO=OPCODE(6)

C CHECK FOR ZERO FEED FLOW
        IF(STRI1(NC+1).EQ.0.0D0) GOTO 99910

C DO THE OPERATION
        CALL FSPLIT(NC,STRI1,STRO1,STRO2,FLOWO)

C GET RETURN STREAMS
        IRET1=INT(OPCODE(4))
        IRET2=INT(OPCODE(5))

        DO 220 I=1,NC+5
        STREAM(IRET2,I)=STRO2(I)
220     STREAM(IRET1,I)=STRO1(I)

C DO ERROR CHECKING
        GOTO 10000


C*********************************************************************
C ISENTHALPIC FLASH
C
C SET FEED STREAM
300     IN1=INT(OPCODE(3))
        DO 310 I=1,NC+5
310     STRF(I)=STREAM(IN1,I)
```

```
C CHECK FOR ZERO FEED FLOW
      IF(STRF(NC+1).EQ.0.0D0) GOTO 99910

C DO THE FLASH
      CALL PHFLASH(NC,STRF,OPCODE(4),STRO1,STRO2,EPS,IPHASE,
     1 IMAXS,ERROR)

C TRANSFER RETURN VALUES TO STREAM MATRIX
      IRET1=INT(OPCODE(5))
      IRET2=INT(OPCODE(6))

      DO 320 I=1,NC+5
      STREAM(IRET1,I)=STRO1(I)
320   STREAM(IRET2,I)=STRO2(I)

C DO ERROR CHECKING
      GOTO 10000


C*********************************************************************
C HEAT EXCHANGER
C
C SET INLET STREAM VALUES
400   IN1=INT(OPCODE(3))
      IN2=INT(OPCODE(4))

      DO 410 I=1,NC+5
      STRI1(I)=STREAM(IN1,I)
410   STRI2(I)=STREAM(IN2,I)

C SET UA
      UA=OPCODE(7)

C CHECK FOR ZERO FEED VALUES
C IF SO THEN PASS STREAMS THROUGH WITH NO HEAT EXCHANGE
      IF((STRI1(NC+1).EQ.0.0D0).OR.(STRI2(NC+1).EQ.0.0D0)) THEN
```

```
                IRET1=INT(OPCODE(5))
                IRET2=INT(OPCODE(6))
                DO 415 I=1,NC+5
                STREAM(IRET1,I)=STRI1(I)
415             STREAM(IRET2,I)=STRI2(I)
                GOTO 99910
        ENDIF

C CALL THE SUBROUTINE
        CALL XCHG(NC,STRI1,STRI2,STRO1,STRO2,UA,EPS,IMAXS,ERROR)

C SAVE RETURN VALUES IN STREAM MATRIX
        IRET1=INT(OPCODE(5))
        IRET2=INT(OPCODE(6))

        DO 420 I=1,NC+5
        STREAM(IRET1,I)=STRO1(I)
420     STREAM(IRET2,I)=STRO2(I)

C DO ERROR CHECK
        GOTO 10000

C*********************************************************************
C REBOILER
C
C SET VALUES FOR INLET STREAM
500      IN1=INT(OPCODE(3))

        DO 510 I=1,NC+5
510     STRI1(I)=STREAM(IN1,I)

C SET VALUE FOR PRESSURE
        P=OPCODE(6)

C SET TEMPERATURE
        TRB=OPCODE(7)
```

```
C CHECK FOR ZERO FEED FLOW
      IF(STRI1(NC+1).EQ.0.0D0) GOTO 99910

      CALL REBOIL(NC,STRI1,STRO1,STRO2,P,TRB,QIN,IMAXS,EPS,ERROR)

C SAVE RETURN STREAMS
      IRET1=INT(OPCODE(4))
      IRET2=INT(OPCODE(5))

      DO 520 I=1,NC+5
      STREAM(IRET2,I)=STRO2(I)
520   STREAM(IRET1,I)=STRO1(I)

C SAVE THE HEAT REQUIREMENT
      UNIT(NUNITS,3)=QIN

C DO ERROR CHECK
      GOTO 10000

C***********************************************************************
C CONDENSER
C
C SET INLET STREAM
600   IN1=INT(OPCODE(3))

      DO 610 I=1,NC+5
610   STRI1(I)=STREAM(IN1,I)

C SET THE PRESSURE
      P=OPCODE(5)

C CHECK FOR ZERO FEED FLOW
      IF(STRI1(NC+1).EQ.0.0D0) GOTO 99910

C CALL THE ROUTINE
      CALL CONDENSE(NC,P,STRI1,STRO1,QOUT,EPS,ERROR)
```

```
C SAVE RETURN STREAM
      IRET1=INT(OPCODE(4))

      DO 620 I=1,NC+5
620   STREAM(IRET1,I)=STRO1(I)

C SAVE HEAT REMOVAL REQUIREMENT
      UNIT(NUNITS,3)=QOUT

C CHECK FOR ERRORS

      GOTO 10000

C*********************************************************************
C MIXER
C
C SET INLET STREAMS
700   IN1=INT(OPCODE(3))
      IN2=INT(OPCODE(4))

      DO 710 I=1,NC+5
      STRI1(I)=STREAM(IN1,I)
710   STRI2(I)=STREAM(IN2,I)

C SET PRESSURE
      P=OPCODE(6)

C CHECK FOR BOTH STREAMS WITH ZERO FLOW
      IF((STRI1(NC+1).EQ.0.0D0).AND.(STRI2(NC+1).EQ.0.0D0)) GOTO 99910

      CALL MIXER(NC,STRI1,STRI2,STRO1,P,IPHASE,EPS,ERROR)

C SAVE RETURN STREAM
      IRET1=INT(OPCODE(5))

      DO 720 I=1,NC+5
720   STREAM(IRET1,I)=STRO1(I)
```

```
C CHECK FOR ERRORS
      GOTO 10000


C*********************************************************************
C SPLITTER
C
C GET INLET STREAM
800   IN1=INT(OPCODE(3))

      DO 810 I=1,NC+5
810   STRI1(I)=STREAM(IN1,I)

C SET FRACTION OF FEED SENT TO OUTLET STREAM 1
      FRAC1=OPCODE(6)

C CHECK FOR ZERO FEED FLOW
      IF(STRI1(NC+1).EQ.0.0D0) GOTO 99910

C CALL THE SPLITTER ROUTINE
      CALL SPLIT(NC,STRI1,STRO1,STRO2,FRAC1)

C SET RETURN STREAMS
      IRET1=INT(OPCODE(4))
      IRET2=INT(OPCODE(5))

      DO 820 I=1,NC+5
      STREAM(IRET1,I)=STRO1(I)
820   STREAM(IRET2,I)=STRO2(I)

C CHECK FOR ERRORS

      GOTO 10000

C*********************************************************************
C PUMP
```

```
C
C GET INLET STREAM
900     IN1=INT(OPCODE(3))

        DO 910 I=1,NC+5
910     STRI1(I)=STREAM(IN1,I)

C SET OUTLET PRESSURE AND LIQUID DENSITY
        POUT=OPCODE(5)
        RHOM=OPCODE(6)

C CHECK FOR ZERO FEED FLOW
        IF(STRI1(NC+1).EQ.0.0D0) GOTO 99910

        CALL PUMP(NC,STRI1,STRO1,POUT,RHOM,POWER,IMAXS,EPS,ERROR)

C GET RETURN STREAM
        IRET1=INT(OPCODE(4))

        DO 920 I=1,NC+5
920     STREAM(IRET1,I)=STRO1(I)

C SAVE POWER REQUIREMENT
        UNIT(NUNITS,2)=POWER

C CHECK FOR ERROR
        GOTO 10000

C*********************************************************************
C REACTOR
C
C SET INLET STREAMS
1000    IN1=INT(OPCODE(3))
        IN2=INT(OPCODE(4))

        DO 1010 I=1,NC+5
        STRI1(I)=STREAM(IN1,I)
```

```
1010    STRI2(I)=STREAM(IN2,I)

C SET REACTOR PRESSURE, HEAT REMOVAL, FRACTION CONVERSION, AND RATE CONSTANT
        PREACT=OPCODE(6)
        QOUT=OPCODE(7)
        FRACT=OPCODE(8)
        RK2=OPCODE(9)

C CHECK FOR ZERO FEED FLOWS
        IF((STRI1(NC+1).EQ.0.0D0).AND.(STRI2(NC+1).EQ.0.0D0)) GOTO 99910

        CALL REACTOR(NC,STRI1,STRI2,PREACT,QOUT,STRO1,FRACT,RK2,VREACT,
     1 IPHASE,EPS,ERROR)

C GET EXIT STREAM VALUES
        IRET1=INT(OPCODE(5))

        DO 1020 I=1,NC+5
1020    STREAM(IRET1,I)=STRO1(I)

C SAVE EXIT HEAT REMOVAL FOR THIS UNIT
        UNIT(NUNITS,3)=QOUT

C SAVE UNIT VOLUME
        UNIT(NUNITS,4)=VREACT

C CHECK FOR ERRORS
        GOTO 10000

C*********************************************************************
C SULFUR SATURATOR
C
C SET INLET STREAM
1100    IN1=INT(OPCODE(3))

        DO 1110 I=1,NC+5
1110    STRI1(I)=STREAM(IN1,I)
```

```
C SET TEMPERATURE AND PRESSURE IN THE UNIT
        TSOL=OPCODE(5)
        PSOL=OPCODE(6)

C CHECK FOR ZERO FEED FLOW
        IF(STRI1(NC+1).EQ.0.0D0) GOTO 99910


C CALL THE ROUTINE
        CALL SULFSOL(NC,STRI1,STRO1,TSOL,PSOL,EPS,ERROR)

C GET THE RETURN STREAM
        IRET1=INT(OPCODE(4))

        DO 1120 I=1,NC+5
1120    STREAM(IRET1,I)=STRO1(I)

C CHECK FOR ERRORS
        GOTO 10000

C*********************************************************************
C SULFUR SETTLER
C
C GET INLET STREAM
1200    IN1=INT(OPCODE(3))

        DO 1210 I=1,NC+5
1210    STRI1(I)=STREAM(IN1,I)

C SET WT% SULFUR IN SULFUR EFFLUENT
        WTS=OPCODE(6)

C SET PRESSURE
        PS=OPCODE(7)

C CHECK FOR ZERO FEED FLOW
```

```
      IF(STRI1(NC+1).EQ.0.0D0) GOTO 99910

      CALL SETTLER(NC,STRI1,STRO1,STRO2,WTS,PS,EPS,ERROR)

C GET RETURN STREAMS
      IRET1=INT(OPCODE(4))
      IRET2=INT(OPCODE(5))

      DO 1220 I=1,NC+5
      STREAM(IRET1,I)=STRO1(I)
1220  STREAM(IRET2,I)=STRO2(I)

C CHECK ERRORS
      GOTO 10000

C*********************************************************************
C SULFUR WASHER
C
C SET INLET SULFUR AND WATER STREAMS
1300  IN1=INT(OPCODE(3))
      IN2=INT(OPCODE(4))

      DO 1310 I=1,NC+5
      STRI1(I)=STREAM(IN1,I)
1310  STRI2(I)=STREAM(IN2,I)

C SET SULFUR FRACTION IN SULFUR STREAM AND SET PRESSURE
      WTS=OPCODE(7)
      PS=OPCODE(8)

C CHECK FOR ZERO FEED FLOWS
      IF((STRI1(NC+1).EQ.0.0D0).AND.(STRI2(NC+1).EQ.0.0D0)) GOTO 99910


      CALL WASHER(NC,STRI1,STRI2,STRO1,STRO2,PS,WTS,EPS,ERROR)

C GET RETURN WASH AND SULFUR STREAMS
```

```
      IRET1=INT(OPCODE(5))
      IRET2=INT(OPCODE(6))

      DO 1320 I=1,NC+5
      STREAM(IRET1,I)=STRO1(I)
1320  STREAM(IRET2,I)=STRO2(I)

C CHECK FOR ERRORS
      GOTO 10000

C*********************************************************************
C MELTER/DECANTER
C
C SET INLET STREAM
1400  IN1=INT(OPCODE(3))

      DO 1410 I=1,NC+5
1410  STRI1(I)=STREAM(IN1,I)

C SET MELT TEMPERATURE AND PRESSURE
      TMELT=OPCODE(6)
      PMELT=OPCODE(7)

C CHECK FOR ZERO FLOW IN FEED
      IF(STRI1(NC+1).EQ.0.0D0) GOTO 99910

      CALL MELTDEC(NC,STRI1,TMELT,PMELT,STRO1,STRO2,QIN,EPS
     1 ,ERROR)

C GET RETURN STREAMS
      IRET1=INT(OPCODE(4))
      IRET2=INT(OPCODE(5))

      DO 1420 I=1,NC+5
      STREAM(IRET1,I)=STRO1(I)
1420  STREAM(IRET2,I)=STRO2(I)
```

```
C SAVE HEAT REQUIREMENT
        UNIT(NUNITS,3)=QIN

C CHECK FOR ERRORS
        GOTO 10000

C*********************************************************************
C FURNACE
C
C SET INLET STREAMS
1500    IN1=INT(OPCODE(3))
        IN2=INT(OPCODE(4))
        IN3=INT(OPCODE(5))

        DO 1510 I=1,NC+5
        STRI1(I)=STREAM(IN1,I)
        STRI2(I)=STREAM(IN2,I)
1510    STRI3(I)=STREAM(IN3,I)

C SET SO2 OUTLET REQUIREMENT
        SO2REQ=OPCODE(6)

C SET EXIT TEMPERATURE AND PRESSURE
        TOUT=OPCODE(8)
        POUT=OPCODE(9)

        CALL FURNACE(NC,SO2REQ,STRI1,STRI2,STRI3,STRO1,TOUT,POUT,QOUT,
     1 EPS,ERROR)

C GET RETURN VALUES (INCLUDING THE MAIN SULFUR AND AIR STREAM FLOWS)
        IRET1=INT(OPCODE(7))

        DO 1520 I=1,NC+5
        STREAM(IN1,I)=STRI1(I)
        STREAM(IN3,I)=STRI3(I)
1520    STREAM(IRET1,I)=STRO1(I)
```

```
C SAVE HEAT REMOVED
      UNIT(NUNITS,3)=QOUT

C CHECK FOR ERRORS
      GOTO 10000

C*********************************************************************
C COMPRESSOR
C
C SET INLET STREAM
1600  IN1=INT(OPCODE(3))

      DO 1610 I=1,NC+5
1610  STRI1(I)=STREAM(IN1,I)

C SET OUTLET PRESSURE
      POUT=OPCODE(5)

C CHECK FOR ZERO FEED FLOW
      IF(STRI1(NC+1).EQ.0.0D0) GOTO 99910

      CALL COMPRESS(NC,STRI1,STRO1,POUT,POWER,IMAXS,EPS,ERROR)

C GET RETURN VALUES
      IRET1=INT(OPCODE(4))

      DO 1620 I=1,NC+5
1620  STREAM(IRET1,I)=STRO1(I)

C SAVE POWER INPUT REQUIRED
      UNIT(NUNITS,2)=POWER

C CHECK FOR ERROR
      GOTO 10000


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

```
C SULFUR CENTRIFUGE
C
C SET INLET STREAMS
1700    IN1=INT(OPCODE(3))
        IN2=INT(OPCODE(4))

        DO 1710 I=1,NC+5
        STRI1(I)=STREAM(IN1,I)
1710    STRI2(I)=STREAM(IN2,I)


C SET PRESSURE
        P=OPCODE(7)

C SET SULFUR WEIGHT PERCENTS IN SULFUR STREAMS IN CENTRIFUGE
        SP1=OPCODE(8)
        SP2=OPCODE(9)

C CHECK FOR STREAMS WITH ZERO FEED FLOW
        IF((STRI1(NC+1).EQ.0.0D0).OR.(STRI2(NC+1).EQ.0.D0)) GOTO 99910

        CALL SULCENT(NC,P,STRI1,STRI2,STRO1,STRO2,SP1,SP2,EPS,ERROR)

C GET RETURN VALUES FOR STREAMS
        IRET1=INT(OPCODE(5))
        IRET2=INT(OPCODE(6))

        DO 1720 I=1,NC+5
        STREAM(IRET1,I)=STRO1(I)
1720    STREAM(IRET2,I)=STRO2(I)

C CHECK FOR ERROR AND DO NEXT COMMAND
        GOTO 10000

C***********************************************************************
C PARTIAL CONDENSER
C
```

```
C SET INLET STREAM
1800    IN1=INT(OPCODE(3))

        DO 1810 I=1,NC+5
1810    STRI1(I)=STREAM(IN1,I)

C SET PRESSURE AND TEMPERATURE
        PCOND=OPCODE(4)
        TCOND=OPCODE(5)

C CHECK FOR ZERO FLOW FEED
        IF(STRI1(NC+1).EQ.0.0D0) GOTO 99910

        CALL PARCOND(NC,STRI1,PCOND,TCOND,STRO1,STRO2,QCHG,EPS,
     1 IMAXS,ERROR)

C GET RETURN STREAMS
        IRET1=INT(OPCODE(6))
        IRET2=INT(OPCODE(7))

        DO 1820 I=1,NC+5
        STREAM(IRET1,I)=STRO1(I)
1820    STREAM(IRET2,I)=STRO2(I)

C SAVE HEAT REMOVAL REQUIREMENT
        UNIT(NUNITS,3)=QCHG

C CHECK FOR ERROR AND DO NEXT OPERATION
        GOTO 10000


C*********************************************************************
C HEATER
C
C
C SET INLET STREAM
1900    IN1=INT(OPCODE(3))
```

```
      DO 1910 I=1,NC+5
1910  STRI1(I)=STREAM(IN1,I)

C SET HEAT ADDED
      QIN=OPCODE(5)

C CHECK FOR ZERO FLOW INLET STREAM
      IF(STRI1(NC+1).EQ.0.0D0) GOTO 99910

      CALL HEATER(NC,STRI1,STRO1,QIN,EPS,ERROR)

C GET RETURN STREAM
      IRET1=INT(OPCODE(4))

      DO 1920 I=1,NC+5
1920  STREAM(IRET1,I)=STRO1(I)

C DO ERROR CHECKING
      GOTO 10000

C*********************************************************************
C MAKUP/STORAGE TANK
C FLOW OF COMPONENT I IS SET, OTHER FLOWS ARE SCALED WHILE
C MAINTAINING THE SAME COMPOSITION
C
C SET INLET STREAM
2000  IN1=INT(OPCODE(3))

      DO 2010 I=1,NC+5
2010  STRI1(I)=STREAM(IN1,I)

C SET FLOW AND COMPONENT NUMBER FOR FIXED FLOW COMPONENT
      FLOWIO=OPCODE(5)
      ICOMP=INT(OPCODE(6))
      TOUT=OPCODE(7)
```

```
      CALL STORAGE(NC,STRI1,STRO1,FLOWIO,ICOMP,TOUT)

C GET RETURN STREAM
      IRET1=INT(OPCODE(4))

      DO 2020 I=1,NC+5
2020  STREAM(IRET1,I)=STRO1(I)

C DO ERROR CHECKING
      GOTO 10000

C*********************************************************************
C CRYSTALLIZER
C SET INLET STREAM
2100  IN1=INT(OPCODE(3))

      DO 2110 I=1,NC+5
2110  STRI1(I)=STREAM(IN1,I)

C CHECK FOR ZERO FLOW INLET STREAM
      IF(STRI1(NC+1).EQ.0.0D0) GOTO 99910

      CALL CRYSTAL(NC,STRI1,STRO1,EPS,ERROR)

C GET RETURN STREAM
      IRET1=INT(OPCODE(4))

      DO 2120 I=1,NC+5
2120  STREAM(IRET1,I)=STRO1(I)

C DO ERROR CHECKING
      GOTO 10000

C*********************************************************************
C SULFUR SPLITTER
C SET INLET STREAM
2200  IN1=INT(OPCODE(3))
```

```
      DO 2210 I=1,NC+5
2210  STRI1(I)=STREAM(IN1,I)

C SET FRACTION OF SOLUTION TO TRANSFER TO EXIT STREAM 1 (ALL THE SULFUR
C GOES TO EXIT STREAM 1)

      FRACT1=OPCODE(6)

C CHECK FOR ZERO FEED FLOW
      IF(STRI1(NC+1).EQ.0.0D0) GOTO 99910

      CALL SULSPLIT(NC,STRI1,STRO1,STRO2,FRACT1)

C GET RETURN STREAMS
      IRET1=INT(OPCODE(4))
      IRET2=INT(OPCODE(5))

      DO 2220 I=1,NC+5
      STREAM(IRET2,I)=STRO2(I)
2220  STREAM(IRET1,I)=STRO1(I)

C DO ERROR CHECKING
      GOTO 10000

C*********************************************************************
C MAKEUP = WITH FIXED FLOW OF ONE COMPONENT -- OTHER FLOWS UNCHANGED
C
C SET INLET STREAM
2300  IN1=INT(OPCODE(3))

      DO 2310 I=1,NC+5
2310  STRI1(I)=STREAM(IN1,I)

C SET FLOW AND COMPONENT NUMBER FOR FIXED FLOW COMPONENT
      FLOWIO=OPCODE(5)
      ICOMP=INT(OPCODE(6))
```

```
      CALL MAKEUP(NC,STRI1,STRO1,FLOWIO,ICOMP,IMAXS,EPS,ERROR)

C GET RETURN STREAM
      IRET1=INT(OPCODE(4))

      DO 2320 I=1,NC+5
2320  STREAM(IRET1,I)=STRO1(I)

C DO ERROR CHECKING
      GOTO 10000

C*********************************************************************
C TSET - SET TEMPERATURE OF A STREAM AND CALCULATE THE ENTHALPY CHANGE
C
C SET INLET STREAM
2400  IN1=INT(OPCODE(3))

      DO 2410 I=1,NC+5
2410  STRI1(I)=STREAM(IN1,I)

C CHECK FOR ZERO FLOW IN FEED
      IF(STRI1(NC+1).EQ.0.0D0) GOTO 99910

C GET DESIRED TEMPERATURE
      T=OPCODE(5)

C DO THE OPERATION
      CALL TSET(NC,STRI1,STRO1,T,QCHG,IMAXS,EPS,ERROR)

C GET RETURN STREAM
      IRET1=INT(OPCODE(4))

      DO 2420 I=1,NC+5
2420  STREAM(IRET1,I)=STRO1(I)

C SAVE HEAT CHANGE REQUIREMENT
```

```
      UNIT(NUNITS,3)=QCHG

C DO ERROR CHECKING
      GOTO 10000


C*****************************************************************
C HCSPLIT - SPLIT HYDROCARBONS FROM A STREAM
C

C SET INLET STREAM
2500  IN1=INT(OPCODE(3))


      DO 2510 I=1,NC+5
2510  STRI1(I)=STREAM(IN1,I)

C CHECK FOR ZERO INLET FLOW
      IF(STRI1(NC+1).EQ.0.D0) GOTO 99910

C PERFORM THE SPLIT
      CALL HCSPLIT(NC,STRI1,STRO1,STRO2)

C GET RETURNED STREAMS
      IRET1=INT(OPCODE(4))
      IRET2=INT(OPCODE(5))

      DO 2520 I=1,NC+5
      STREAM(IRET1,I)=STRO1(I)
2520  STREAM(IRET2,I)=STRO2(I)

C PERFORM ERROR CHECKING
      GOTO 10000

C*********************************************************************
C UNIT NOT EXECUTED DUE TO ZERO FLOW IN ONE OR MORE INLET FEEDS
C FLAG THE ERROR FOR THE OPERATOR
```

```
99910   ERROR=10

C ******************************************************************
C BOTTOM OF OPERATION CODE LOOP
C CHECK FOR AN ERROR ON THE PRESENT OPERATION
C
C CHECK FOR CONVERGENCE ERRORS

10000   IF(ERROR.EQ.1) THEN
                WRITE(6,20100) OPCODE(2)
                WRITE(1,20100) OPCODE(2)
        ENDIF

C CHECK FOR MIXED VAPOR AND LIQUID STREAM
        IF(ERROR.EQ.2) THEN
                WRITE(6,20200) OPCODE(2)
                WRITE(1,20200) OPCODE(2)
        ENDIF

C CHECK FOR SINGLE PHASE STREAM RETURN FROM A FLASH CALCULATION
        IF(ERROR.EQ.3) THEN
                WRITE(6,20300) OPCODE(2)
                WRITE(1,20300) OPCODE(2)
        ENDIF


C CHECK FOR UNIT NOT EXECUTED ERROR
        IF(ERROR.EQ.10) THEN
                WRITE(6,20325) OPCODE(2)
                WRITE(1,20325) OPCODE(2)
        ENDIF

C CLEAR ERROR
        ERROR=0

C GO ON TO NEXT UNIT OPERATION
```

```
      WRITE(6,90000)
90000 FORMAT(X,'NOW LEAVING ERROR CHECKER')

      GOTO 50


C***********************************************************************
C BOTTOM OF MAIN PROGRAM LOOP
C CHECK FOR CONVERGENCE

15000 ITER=ITER+1
      ICWEGS=ICWEGS+1
      ICOUNT=ICOUNT+1
      WRITE(6,25020) ITER
      WRITE(1,25020) ITER
25020 FORMAT(' NOW INCREMENTING THE ITERATION COUNT, ITER=',I3)

C CHECK CONVERGENCE ON ALL STREAMS MOLE FRACTIONS, FLOW, AND TEMPERATURE
      DO 15100 I=1,NSTREAMS
      DO 15100 J=1,NC+3
      IF(STREAM(I,J).LT.1.0D-8) GOTO 15100
      IF(DABS((STREAM(I,J)-STROLD(I,J))/STREAM(I,J)).GT.EPSILON) THEN
       GOTO 15200
      ENDIF
15100 CONTINUE

C FELL THROUGH LOOP, MUST BE CONVERGED
      GOTO 15500

C NOT CONVERGED, SET UP NEXT ITERATION
C PRINT CURRENT RESULTS

C SHOW USER WHICH STREAM VARIABLE IS NOT CONVERGED
15200 WRITE(6,20515) I,J
      WRITE(1,20515) I,J

C PRINT CURRENT RESULTS ON FIRST ITERATION AND EVERY FIFTH THEREAFTER
```

```
      IREM=MOD(ITER,5)
      IF((IREM.EQ.0).OR.(ITER.EQ.1)) THEN

C     WRITE(6,20500)
      WRITE(1,20500)
      DO 15205 I=1,NSTREAMS
C     WRITE(6,20510) I,(STREAM(I,J),J=1,NC+5)
15205 WRITE(1,20510) I,(STREAM(I,J),J=1,NC+5)

      ENDIF

C  SAVE ALL STREAM VARIABLES
      IF(ITER.LT.IMAX) THEN
              DO 15210 I=1,NSTREAMS
              DO 15210 J=1,NC+3
15210         STROLD(I,J)=STREAM(I,J)

C SAVE COMPONENT FLOWS FOR WEGSTEIN ACCELERATION
      IF((ITER.EQ.2).OR.(ICWEGS.EQ.2)) THEN
              DO 15215 I=1,NSTREAMS
              DO 15215 J=1,NC
15215         FLOW(I,J,1)=STREAM(I,J)*STREAM(I,NC+1)
      ENDIF


      IF((ITER.EQ.3).OR.(ICWEGS.EQ.3)) THEN
              DO 15220 I=1,NSTREAMS
              DO 15220 J=1,NC
15220         FLOW(I,J,2)=STREAM(I,J)*STREAM(I,NC+1)
      ENDIF

C AT THE END OF THE THIRD AND SUBSEQUENT ITERATIONS WEGSTEIN
C ACCELERATION IS APPLIED TO ALL OF THE COMPONENT FLOWS
C THIS IS ACCOMPLISHED BY:
C    x_{k+1} = q x_k + (1-q) x'_{k+1}
C
C    q = (x'_{k+1} - x'_k)/(x'_{k+1} - 2x'_k + x'_{k-1})
```

```
C WHERE:
C THE SUBSCRIPT INDICATES THE ITERATION NUMBER
C A SUPERSCRIPT ' INDICATES A VALUE OBTAINED FROM A DIRECT SUBSTITUTION
C   CALCULATION OVER THE FLOWSHEET
C
C ACCELERATION GENERALLY PRODUCES CONVERGENCE WHEN Q<0.5
C ON ENTRY TO THE ACCELERATOR ROUTINES
C FLOW(I,J,1) = X'K-1 (VALUE OBTAINED FROM THE K-1 TH ITERATION) (XPM1)

C FLOW(I,J,2) = X'K   (VALUE OBTAINED FROM THE K TH ITERATION)     (XP1)

C FLOW(I,J,3) = XK          (INPUT VALUE TO THE K TH ITERATION)    (X1)

C STREAM(I,J) = X'K+1 (VALUE OUTPUT FROM THE K+1 TH OR CURRENT ITERATION) (XPP1)

C CONVERT STREAM MOLE FRACTIONS TO COMPONENT FLOWS (THIS SIMPLIFIES THE
C  WEGSTEIN CALCULATIONS AND IS NEEDED FOR RE-NORMALIZATION AFTER ACCELERATION)
        DO 15225 I=1,NSTREAMS
        DO 15225 J=1,NC
15225   STREAM(I,J)=STREAM(I,J)*STREAM(I,NC+1)

C PERFORM ACCELERATION

        IF((ITER.GE.4).OR.(ICWEGS.GE.4)) THEN
                DO 15230 I=1,NSTREAMS
                DO 15230 J=1,NC
                X1=FLOW(I,J,3)
                IF((ITER.EQ.4).OR.(ICWEGS.EQ.4)) THEN
                        X1=FLOW(I,J,2)
                ENDIF
                XPM1=FLOW(I,J,1)
                XP1=FLOW(I,J,2)
                XPP1=STREAM(I,J)
                DNOMQ=XPP1-2.D0*XP1+XPM1
                IF(DNOMQ.LT.1.D-10) GOTO 15235
                Q=(XPP1-XP1)/DNOMQ
                IF(Q.GT.0.5D0) GOTO 15235
```

```
                STREAM(I,J)=Q*X1+(1.D0-Q)*XPP1
                IF(STREAM(I,J).LT.0.0D0) THEN
                        STREAM(I,J)=STROLD(I,J)*STROLD(I,NC+1)
                ENDIF
15235           FLOW(I,J,1)=XP1
                FLOW(I,J,2)=XPP1
                FLOW(I,J,3)=STREAM(I,J)
15230           CONTINUE
        ENDIF


C CONVERT WEGSTEIN COMPONENT FLOW UPDATES BACK INTO MOLE FRACTIONS
C RECALCULATE TOTAL FLOW

        DO 15221 I=1,NSTREAMS
        STREAM(I,NC+1)=0.0D0
        DO 15221 J=1,NC
15221   STREAM(I,NC+1)=STREAM(I,NC+1)+STREAM(I,J)

        DO 15222 I=1,NSTREAMS
        IF(STREAM(I,NC+1).EQ.0.D0) GOTO 15222
        DO 15222 J=1,NC
        STREAM(I,J)=STREAM(I,J)/STREAM(I,NC+1)
15222   CONTINUE

        ENDIF


C CHECK FOR ITERATION COUNT STOP CODE, I.E. OPCODE(1)=-N.0
15219   IF((OPCODE(1).LT.0.0D0).AND.(IABS(INT(OPCODE(1))).LE.ITER))
     1 GOTO 15500

        ENDFILE(1)
        BACKSPACE(1)

C CHECK FOR MAXIMUM ITERATION FOR THIS INVOCATION OF BERKSULF
        IF((ICOUNT.GT.ICMAX).AND.(ITER.LT.IMAX))  THEN
```

```
C CLOSE THE NORMAL OUTPUT FILE
        CLOSE(1)
C SET END OF FILE MARKER FOR STREAMS
       STREAM(NSTREAMS+1,NC+1)=-1.D0
C OPEN THE TEMPORARY STORAGE FILE AND WRITE OUT THE STREAMS DATA
        OPEN(4,FILE='STREAMS.TMP')
        WRITE(4,*) ITER
        DO 15300 I=1,NSTREAMS+1
        WRITE(4,20505) (STREAM(I,J),J=1,NC)
15300   WRITE(4,20507) (STREAM(I,J),J=NC+1,NC+5)
        CLOSE(4)

C SAVE THE ACCELERATION ARRAY
        OPEN(9,FILE='WEGST')
        DO 15310 I=1,NSTREAMS
        DO 15310 J=1,NC
15310   WRITE(9,*) (FLOW(I,J,K),K=1,3)
        CLOSE(9)
        STOP
       ENDIF

       IF(ITER.LT.IMAX) GOTO 40


C SHOW THAT IT DID NOT CONVERGE
       WRITE(6,20400)
       WRITE(1,20400)

C ALL DONE, CONVERGED OR NOT
C PRINT ALL THE RESULTS

C PRINT OUT ALL THE STREAMS
C ALSO SAVE DATA IN A DISK FILE
15500  OPEN(3,FILE='UNITOUT.DAT')

       WRITE(6,20500)
       WRITE(1,20500)
```

```
      DO 15510 I=1,NSTREAMS
      WRITE(1,20510) I,(STREAM(I,J),J=1,NC+5)
15510 WRITE(6,20510) I,(STREAM(I,J),J=1,NC+5)

      DO 15515 I=1,NSTREAMS
15515 WRITE(1,20525) I,(STREAM(I,NC+1)-STROLD(I,NC+1))

      WRITE(1,20520) ITER
      WRITE(6,20520) ITER

C PRINT OUT THE POWER AND HEAT VALUES ASSOCIATED WITH UNIT OPERATIONS
      WRITE(6,20600)
      WRITE(3,20600)
      DO 15520 I=1,NUNITS
      WRITE(3,20610) (UNIT(I,J),J=1,4)
15520 WRITE(6,20610) (UNIT(I,J),J=1,4)

      CLOSE(1)
      CLOSE(3)

C WRITE OUT THE FINAL STREAMS VALUES IN CASE WE WISH TO USE THEM
C AGAIN
C ALSO DELETE TMP FILE
      OPEN(4,FILE='STREAMS.TMP')
      CLOSE(4,STATUS='DELETE')

C GET RID OF EXISTING FILE
      OPEN(4,FILE='STREAMS.OLD')
      CLOSE(4,STATUS='DELETE')
      OPEN(4,FILE='STREAMS.OLD')

C SET UP PHONY ITERATION COUNT AND SAVE THE STREAMS VALUES
       ITER=1
       WRITE(4,*) ITER
       DO 15600 I=1,NSTREAMS+1
       WRITE(4,20505) (STREAM(I,J),J=1,NC)
15600  WRITE(4,20507) (STREAM(I,J),J=NC+1,NC+5)
```

```
      CLOSE(4)

C ABORT BERKRUN
      CALL BERKSTOP

      STOP

C VARIOUS FORMAT STATEMENTS
19950 FORMAT(X,'NOW INITIALIZING STREAM NUMBER ',I2)
20000 FORMAT(1X,'FLOWSHEET ITERATION = ',I3,'   OPERATION NOW = ',
     1 F5.2)
20010 FORMAT(X,'ERROR! STREAM ',I3,' SHOULD BE LIQUID')
20020 FORMAT(X,'ERROR! STREAM ',I3,' SHOULD BE A VAPOR')
20030 FORMAT(X,'WARNING! STREAM ',I3,' CONTAINS BOTH VAPOR AND
     1 LIQUID')
20100 FORMAT(X,'ERROR! NON-CONVERGENCE ON UNIT ',F10.2)
20200 FORMAT(X,'WARNING! MIXED LIQUID AND VAPOR STREAM RETURNED BY
     1 UNIT ',F10.2)
20300 FORMAT(X,'WARNING! SINGLE PHASE RETURNED BY A FLASH
     1 CALCULATION ON UNIT ',F10.2)
20325 FORMAT(X,'WARNING! OPERATION NOT PERFORMED DUE TO ZERO FEED
     1 ON UNIT ',F10.2)
20400 FORMAT(1X,'THE SIMULATION DID NOT CONVERGE')
20500 FORMAT(1X,'ST#',X,3X,'SOLV',3X,3X,'H2O',4X,3X,'H2S',4X,3X,'SO2',
     1 4X,3X,'SULD',3X,3X,'CO2',4X,3X,'N2',5X,3X,'C1',5X,3X,'H2',5X,
     2 3X,'CO',5X,3X,'C2',5X,3X,'C3',5X,3X,'C4',5X,3X,'C5+',4X,
     3 3X,'C1SH',3X,3X,'C2SH',3X,3X,'SULP',3X,3X,'H2OP',3X,3X,'O2',
     4 5X,3X,'UNKN',3X,3X,'FLOW',3X,3X,'TEMP',3X,3X,'PRES',3X,3X,
     5 'ENTH',5X,4X,'PHAS')
20505 FORMAT(1X,20(D11.6,X))
20507 FORMAT(1X,5(D20.10,X))
20510 FORMAT(1X,I3,X,23(D9.4,X),D15.9,X,F4.1)
20515 FORMAT(X,'NOT CONVERGED, STREAM=',I3,' VARIABLE=',I3)
20520 FORMAT(X,' ITERATION COUNT=',I3)
20525 FORMAT(X,'STREAM=',I3,' DELTA FLOW=',D20.10)
20600 FORMAT(1X,'UNIT',17X,'POWER',24X,'HEAT',24X,'SIZE')
20610 FORMAT(1X,F4.1,7X,D20.10,7X,D20.10,7X,D20.10)
```

END

```
;BERKSTOP
;
;FORTRAN CALLABLE SUBROUTINE TO ABORT THE PROCESS BERKRUN.CMD
;THIS ROUTINE IS CALLED WHEN BERKSULF HAS CONVERGED
;IT STOPS THE REPEATED INVOCATIONS BY BERKRUN
;
; LAST UPDATE: 04/21/86
;
; WRITTEN IN DRI RASM86 VER 1.2
;
;EQUATES
CCPM    EQU     224             ;SYSTEM CALL ENTRY

P_PDADR EQU     156             ;GET PD ADDRESS
ABORT   EQU     157             ;ABORT SPECIFIED PROCESS

P_CNS   EQU BYTE PTR 20H        ;CONSOLE NUMBER IN PD

        CSEG
PUBLIC  BERKSTOP

BERKSTOP:
        PUSH    DS
        PUSH    ES
        MOV     AX,SEG APB          ;POINT TO ABORT PARAMETER BLOCK
        MOV     DS,AX               ; SEGMENT
        MOV     CL,P_PDADR          ;GET THE PD ADDRESS OF BERKSULF
        INT     CCPM
        MOV     BX,AX
        MOV     BL,ES:P_CNS[BX]     ;LOOK UP CURRENT CONSOLE NUMBER
        MOV     CNS,BL              ;SAVE IT IN BLOCK
        MOV     CL,ABORT            ;ABORT BERKRUN
        MOV     DX,OFFSET APB
        INT     CCPM
        POP     ES
        POP     DS
        RETF                        ;RETURN TO BERKSULF
```

```
        DSEG
APB     DW      0,0             ;ABORT PARAMETER BLOCK
CNS     DB      0,0             ;CONSOLE FIELD
        DB      'BERKRUN '

        END
```

```
      SUBROUTINE ABSORBER(NSTAGE,NCOMP,P,FEED,ZFEED,VLIQT,RHOM,
     1 R2CONST,QR,QI,TC,FS,STVAP,STLIQ,STLIQC,IMAX,NCSUB,UNIT,
     2 IOFLAG,ITKEY)

C LAST UPDATE:  6/10/86

C*****************************************************************
C THIS ROUTINE PERFORMS STAGE TO STAGE CALCULATIONS FOR ABSORBERS
C  STRIPPERS, AND DISTILLATION COLUMNS
C
C MAJOR VARIABLES:
C
C    FEED = STAGE #, FLOW, AND ENTHALPY OF THE FEEDS TO THE
C      COLUMN (MAX. NO. OF FEEDS = 5) FROM THE MAIN PROGRAM
C    ZFEED = FEED COMPOSITION
C
C  VARIABLES USED IN NEWTON-RAPHSON STAGE TO STAGE CALCULATIONS:
C    F       = FLOW AND ENTHALPY OF FEED TO EACH STAGE
C    ZF      = COMPOSITION OF FEED TO EACH STAGE
C    T       = TEMPERATURE ON STAGE J
C    V       = VAPOR FLOW FROM STAGE J
C    L       = LIQUID FLOW FROM STAGE J
C    X       = LIQUID PHASE COMPOSITION ON STAGE J
C    Y       = VAPOR PHASE COMPOSITION ON STAGE J
C    K       = EQUILIBRIUM CONSTANTS FOR STAGE J
C    HL      = LIQUID MOLAR ENTHALPIES FOR STAGE J
C    HV      = VAPOR MOLAR ENTHALPIES FOR STAGE J
C    DKDT    = DERIVATIVES OF K WITH RESPECT TO T
C    DHLDT   = DERIVS. OF HL WITH RESPECT TO T
C    DHVDT   = DERIVS. OF HV WITH RESPECT TO T
C    R       = MOLAR EXTENT OF REACTION
C              (=K2*CSO2*CH2S AT EXIT CONDITIONS)
C    QH      = HEAT ADDED OR REMOVED ON STAGE J
C    RS      = SIDE DRAW ON STAGE J
C    XH2SRXN = EXIT H2S MOLE FRACTION ON A STAGE
C    XSO2RXN = EXIT SO2 MOLE FRACTION ON A STAGE
C               (THESE LAST TWO ARE USED IN THE REACTION SOLVER)
```

```
C
C
C NOTES: ALTHOUGH THE MAIN PROGRAM SUPPORTS 20 COMPONENTS,
C  THE MAXIMUM HANDLED BY THIS SUBROUTINE IS 15. IF COMPONENTS
C  16-20 ARE TO CONSIDERED IN THE CALCULATIONS THIS CODE WILL
C  NEED TO BE CHANGED. HOWEVER, THE NUMBER OF STAGES WILL HAVE
C  TO BE REDUCED IN ORDER TO KEEP THE MAXIMUM NUMBER OF VARIABLES
C  IN EBND TO 8000 (SEE BAND)
C


      IMPLICIT REAL*8 (A-H,O-Z)

      REAL*8 FEED(5,4),ZFEED(5,20),F(23,3),ZF(23,15)
      REAL*8 T(23),V(23),L(23),X(23,15),Y(23,15)
      REAL*8 K(23,15),HL(23,15),HV(23,15)
      REAL*8 DKDT(23,15),DHLDT(23,15)
      REAL*8 DHVDT(23,15),R(23)
      REAL*8 XH2SRXN(23),XSO2RXN(23)

C THE FOLLOWING ARRAYS ARE USED FOR INITIALIZING ALL STAGE VARIABLES
      REAL*8 KEQ(20),DKEQDT(20),HLIQ(20),HVAP(20)
      REAL*8 DHLIQDT(20), DHVAPDT(20), XTEMP(20)
      REAL*8 YTEMP(20),FLTEMP,FT(20)

C THE FOLLOWING ARE USED IN THE THOMAS CALCULATION
      REAL*8 W(23),G(23),Q(23),AB(23,15),BB(23,15),DB(23,15)

      REAL*8 STVAP(25),STLIQ(25),STLIQC(25)

      REAL*8 QH(23),RS(23)

      REAL*8 QI(2,2)

      REAL*8 ABND(18,18),BBND(18,18),CBND(18,25),DBND(18,37)
      REAL*8 GBND(18),XBND(18,18),YBND(18,18)
```

```
      CHARACTER*11 FNAME,PNAME

      INTEGER*2 ALPHA(20)

      COMMON/ALPH/ALPHA

      COMMON/BND/ ABND,BBND,CBND,DBND,GBND,XBND,YBND,
     1 NVAR,NJ

      EXTERNAL SULFUR

C SHOW NO ERROR ON DISK IO
      IOERR=0

C SET UP AND OPEN OUTPUT FILES FOR THIS CALCULATION
C  ABSORBWX.YZ IS A HUMAN READABLE PRINTOUT OF THE RESULTS OF
C   THE CURRENT CALCULATION
C  PROFLEWX.YZ IS A COMPRESSED FORM OF THE SAME TO BE USED AS
C   A SET OF GUESSES FOR THIS UNIT ON THE NEXT ITERATION OF
C   THE FLOWSHEET

      I1=INT(UNIT/10.D0)
      I2=INT(UNIT-I1*10.D0)
      I3=INT(10.D0*(UNIT-I1*10.D0-I2))
      I4=INT(100.D0*(UNIT-I1*10.D0-I2-FLOAT(I3)/10.D0))
      FNAME='ABSORB'//CHAR(I1+48)//CHAR(I2+48)//'.'
     1 //CHAR(I3+48)//CHAR(I4+48)
      PNAME='PROFLE'//CHAR(I1+48)//CHAR(I2+48)//'.'
     1 //CHAR(I3+48)//CHAR(I4+48)

      OPEN(10,FILE=FNAME)
      CLOSE(10,STATUS='DELETE')

      OPEN(10,FILE=FNAME)

C SHOW ENTRY PARAMETERS
```

```
      WRITE(6,93456) ITKEY,NCSUB
93456 FORMAT(X,'ITKEY=',I3,' NCSUB=',I3)

      WRITE(10,50000) R2CONST,VLIQT,RHOM
50000 FORMAT(' K2=',F6.1,' VTRAY=',F6.4,' RHOL=',F5.2)

      WRITE(10,50001) QR,TC,FS
      WRITE(6,50001) QR,TC,FS
50001 FORMAT(X,'QR=',D12.5,' TC=',F6.2,' FS=',F5.3)
      IMAXA=100

C-INITIALIZE THE RETURN VALUE MATRICES
      DO 10 I=1,NCOMP+3
      STVAP(I)=0.0D0
      STLIQ(I)=0.0D0
10    STLIQC(I)=0.0D0

C-SET UP INTERSTAGE COOLER/HEATER VALUES AND SIDEDRAW FRACTIONS
      DO 15 I=1,NSTAGE
      QH(I)=0.0D0
15    RS(I)=1.0D0

C FIND INTERSTAGE COOLERS/HEATERS IN USE AND FILL THEM IN
C THE MATRIX QI CONTAINS STAGE#, HEAT REMOVED FOR EACH
C INTERCOOLER/HEATER IN USE

      DO 20 I=1,2
      ISTGE=INT(QI(I,1))
C CHECK FOR ILLEGAL OR NON-EXISTENT STAGE
      IF((ISTGE.EQ.0).OR.(ISTGE.GT.NSTAGE)) GOTO 20
      QH(ISTGE)=QI(I,2)
20    CONTINUE

C REBOILER
      QH(1)=QR

C SIDEDRAW AT CONDENSER
```

```
      RS(NSTAGE)=1.D0-FS

C SET MAXIMUM NUMBER OF COMPONENTS FOR SUBROUTINE
      NCOMPOLD=NCOMP
      NCOMP=NCSUB

C-SET NUMBER OF VARIABLES AND NUMBER OF INCREMENTS FOR BAND
      NVAR=NCOMP+3
      NJ=NSTAGE

C- NOTE THAT THIS PROGRAM DOES THE CALCULATION FROM THE BOTTOM UP
C- I.E. J=1 IS AT THE VAPOR INLET OF THE COLUMN


C- SET FEED VALUES FOR ALL STAGES. USE MATRIX FEED TO DETERMINE
C- WHICH STAGES ARE TO HAVE FEEDS. IF FOR A GIVEN I THE VALUE OF
C- FEED(I,2) IS NON-ZERO THEN THE STAGE NUMBER IS GIVEN BY FEED(I,1).
C- THE COMPOSITION OF THE FEED IS GIVEN IN ZFEED(I,IX).
C
C- THE FORM OF THE FEED MATRIX IS FEED# BY STAGE,FLOW,TEMP,ENTHALPY
C- THE FORM OF THE ZFEED MATRIX IS FEED# BY NCOMPONENTS
C
C- ALL THE FEEDS EXCEPT THE VAPOR ON TRAY 1 ARE ASSUMED TO BE LIQUIDS

C- ZERO THE FEED MATRICES
      DO 190 I=1,NSTAGE
      DO 180 J=1,3
180   F(I,J)=0.000
      DO 190 J=1,NCOMP
190   ZF(I,J)=0.000

C- FIND FEEDS ACTUALLY IN USE AND SET INTO FEED MATRICES
      DO 200 J=1,5
      IF (FEED(J,2).EQ.0) GOTO 200
C-    FOUND A FEED. TRANSFER ITS CONTENTS TO THE APPROPRIATE SPOT
C-    IN THE F MATRIX FOR THE COLUMN
```

```
      ISTAGE=INT(FEED(J,1))
      DO 210 I=2,4
210   F(ISTAGE,I-1)=FEED(J,I)
      DO 220 I=1,NCOMP
220   ZF(ISTAGE,I)=ZFEED(J,I)

200   CONTINUE

C WRITE OUT THE CURRENT FEEDS AND LOCATIONS
      DO 50010 I=1,NSTAGE
      WRITE(10,50020) (F(I,JK),JK=1,3)
50010 WRITE(6,50020) (F(I,JK),JK=1,3)
50020 FORMAT(X,'F(FLOW)= ',D12.5,'   F(TEMP)= ',F6.2,
     1 '   F(ENTHALPY)= ',D12.5)


C- FOR FIRST TIME THROUGH THE FLOWSHEET
C- SET INITIAL GUESSES FOR ALL T, V, L , AND X'S
C- SET ALL T'S TO AVERAGE OF ENTERING LIQUID T'S
C- SET ALL V'S TO ENTERING VAPOR FEED (ABSORPTION) OR L AT BOTTOM (DISTILLATION)
C- SET ALL L'S TO ENTERING LIQUID PLUS FEEDS
C- SET ALL X'S TO THOSE IN THE ENTERING LIQUID CORRECTING FOR ANY SIDE FEEDS

C-ON SUBSEQUENT TRIPS THROUGH A FLOWSHEET USE THE PROFILE OBTAINED IN
C-THE PREVIOUS ITERATION FOR THE INITIAL GUESSES

270   CONTINUE

      IF(IOERR.NE.0) THEN

C- ZERO TEMPORARY FLOW VALUES FOR EACH COMPONENT AND THE MAIN LIQUID STREAM

      FLTEMP=0.0D0
      DO 280 IX=1,NCOMP
280   FT(IX)=0.0D0

C- ITERATE FROM THE TOP DOWN
```

```
      DO 300 J=NSTAGE,2,-1
      V(J)=F(1,1)
      L(J)=FLTEMP+F(J,1)
      FLTEMP=L(J)
      DO 300 IX=1,NCOMP
C- UPDATE COMPONENT FLOWS
      FT(IX)=FT(IX)+F(J,1)*ZF(J,IX)
      X(J,IX)=FT(IX)/FLTEMP
300   Y(J,IX)=ZF(1,IX)

C-SET VALUES FOR STAGE 1, ADD A SMALL AMOUNT OF THE VAPOR TO THE LIQUID STREAM
      V(1)=F(1,1)
      L(1)=L(2)+0.1D0*F(1,1)
      DO 302 IX=1,NCOMP
      FT(IX)=FT(IX)+0.1D0*F(1,1)*ZF(1,IX)
      X(1,IX)=FT(IX)/L(1)
302   Y(1,IX)=Y(2,IX)

C-SET TEMPERATURE PROFILES BASED ON FEED TEMPS (ASSUME EQUAL HEAT CAPACITIES)
      T(NSTAGE)=F(NSTAGE,2)
      DO 303 J=NSTAGE-1,1,-1
303   T(J)=(L(J+1)*T(J+1)+F(J,1)*F(J,2))/(L(J+1)+F(J,1))


C-MODIFY GUESSES FOR A DISTILLATION COLUMN

C-SET ALL VAPOR FLOWS -- ASSUME TOTAL REBOILER
      IF(F(1,1).EQ.0.D0) THEN
       DO 304 J=1,NSTAGE
       V(J)=L(1)
       DO 304 IX=1,NCOMP
304    Y(J,IX)=X(1,IX)
      ENDIF

C-SET LIQUID FLOWS AND TEMPS FOR STAGES ABOVE THE FEED
C-PROPAGATE THE FLOWS UP THE COLUMN -- ASSUME INFINITE REFLUX
      IF(F(NSTAGE,1).EQ.0.D0) THEN
```

```
      DO 305 J=1,NSTAGE
      IF(L(J).NE.0.D0) GOTO 305
      L(J)=V(NSTAGE)
      DO 305 IX=1,NCOMP
      X(J,IX)=Y(NSTAGE,IX)
305   CONTINUE

C-FIX TEMPERATURE PROFILE
      T(NSTAGE)=F(NSTAGE,2)
      DO 306 J=NSTAGE-1,1,-1
      IF(T(J+1).EQ.0.0D0) THEN
      T(J)=F(J,2)
      ELSE
      T(J)=(L(J+1)*T(J+1)+F(J,1)*F(J,2))/(L(J+1)+F(J,1))
      ENDIF
306   CONTINUE
      DO 307 J=2,NSTAGE
      IF(T(J).NE.0.D0) GOTO 307
      T(J)=T(J-1)
307   CONTINUE
     ENDIF

     ELSE
C-READ PROFILE FROM PREVIOUS TRIP THROUGH THE FLOWSHEET AS GUESSES
C-FOR THIS ITERATION
C-IF AN ERROR OCCURS THEN GUESSES ARE MADE AS ON THE FIRST ITERATION

     IF((IOFLAG.EQ.0).AND.(ITKEY.EQ.1)) THEN
            IOERR=-1
            GOTO 270
     ENDIF

     OPEN(11,FILE=PNAME,IOSTAT=IOERR,ERR=270)
     DO 308 J=1,NSTAGE
     READ(11,70000,IOSTAT=IOERR,ERR=309,END=309) T(J),V(J),L(J)
     READ(11,70030,IOSTAT=IOERR,ERR=309,END=309) (X(J,IX),IX=1,NCOMP)
308  READ(11,70030,IOSTAT=IOERR,ERR=309,END=309) (Y(J,IX),IX=1,NCOMP)
```

```
C-DELETE THE OLD PROFILE
309     CLOSE(11,STATUS='DELETE')
        IF(IOERR.EQ.-1) GOTO 270
        ENDIF

C- SET ITERATION COUNTER (MAX. OF TEN ITERATIONS)
        ITER=0

C-SET CONDENSER TEMPERATURE IF SPECIFIED
        IF(TC.NE.0.0D0) THEN
         T(NSTAGE)=TC
        ENDIF

C- MAIN OUTSIDE ITERATION LOOP

1000    CONTINUE


C- DO CALCULATION FROM THE BOTTOM OF THE COLUMN UP
        DO 320 J=1,NSTAGE

C- CALCULATE ALL VALUES FOR K,HL,HV,DKDT,DHLDT,DHVDT
C- CALCULATE VALUES AT EACH J, USE HEAT AND KVAL SUBROUTINES
C- SEE SUBROUTINES FOR ARGUMENT USAGE. NOTE THAT HT IS THE TOTAL AVERAGE
C-MOLAR STREAM ENTHALPY.

C- SET TEMPORARY MATRIX OF ALL X'S AND Y'S FOR USE IN THE SUBROUTINES
C- USE CURRENT K VALUES TO CALCULATE Y'S. USE NORMALIZED MOLE FRACTS.

        XSUM=0.0D0
        YSUM=0.0D0
        DO 310 IX=1,NCOMP
        YSUM=YSUM+Y(J,IX)
310     XSUM=XSUM+X(J,IX)

        DO 330 IX=1,NCOMP
```

```
      XTEMP(IX)=X(J,IX)/XSUM
330   YTEMP(IX)=Y(J,IX)/YSUM

      CALL HEAT(0,1,NCOMP,T(J),P,XTEMP,HLIQ,HT)
      CALL HEAT(1,1,NCOMP,T(J),P,XTEMP,DHLIQDT,HT)
      CALL HEAT(0,2,NCOMP,T(J),P,YTEMP,HVAP,HT)
      CALL HEAT(1,2,NCOMP,T(J),P,YTEMP,DHVAPDT,HT)

      CALL KVAL(0,NCOMP,T(J),P,XTEMP,YTEMP,KEQ)
      CALL KVAL(1,NCOMP,T(J),P,XTEMP,YTEMP,DKEQDT)

      DO 320 IX=1,NCOMP
      HL(J,IX)=HLIQ(IX)
      HV(J,IX)=HVAP(IX)
      DHLDT(J,IX)=DHLIQDT(IX)
      DHVDT(J,IX)=DHVAPDT(IX)
      K(J,IX)=KEQ(IX)
320   DKDT(J,IX)=DKEQDT(IX)

      WRITE(6,51000)
51000 FORMAT(X,' PAST THERMO INIT',/)

C MODIFY INITIAL GUESSES FOR COMPONENT BALANCES USING THE THOMAS METHOD
      IF(ITER.EQ.0) THEN

      DO 450 J=2,NSTAGE
      DO 450 IX=1,NCOMP
      AB(J,IX)=-K(J-1,IX)*V(J-1)/L(J-1)
      BB(J,IX)=1.0D0+(K(J,IX)*V(J))/L(J)
450   DB(J,IX)=ZF(J,IX)*F(J,1)

      DO 455 IX=1,NCOMP
      DB(1,IX)=ZF(1,IX)*F(1,1)
      BB(1,IX)=1.0D0+(K(1,IX)*V(1))/L(1)
455   AB(1,IX)=0.0D0

C CARRY OUT THOMAS CALCULATION FOR EACH COMPONENT
```

```
      DO 490 IX=1,NCOMP
      W(1)=BB(1,IX)
      G(1)=DB(1,IX)/W(1)
      Q(1)=-1.0/W(1)

      DO 460 J=2,NSTAGE
      W(J)=BB(J,IX)-AB(J,IX)*Q(J-1)
      G(J)=(DB(J,IX)-AB(J,IX)*G(J-1))/W(J)
460   Q(J)=-1.0/W(J)
      Q(NSTAGE)=0.0D0
      X(NSTAGE,IX)=G(NSTAGE)/L(NSTAGE)
      FL=G(NSTAGE)
      DO 490 J=NSTAGE-1,1,-1
      FL=G(J)-Q(J)*FL
490   X(J,IX)=FL/L(J)

      ITER=ITER+1

      GOTO 1000
      ENDIF


C IF THE SECOND ORDER RATE CONSTANT IS NON-ZERO
C SOLVE MASS BALANCE EQUATIONS FOR H2S AND SO2 MOLE FRACTIONS ON EVERY
C STAGE. INCLUDE EXPLICIT REACTION DEPENDENCE.

    IF(R2CONST.EQ.0.D0) GOTO 501

C SET VARIABLE COUNT AND ZERO ITERATION NUMBER
      NVAR=2
      ITERRXN=0

C CALCULATE THE CONSTANT PART FOR ALL OF THE REACTION TERMS
      RXNTERM=RHOM*RHOM*VLIQT*R2CONST

      IRXN=0
```

```
      DO 2100 I=1,NSTAGE
      IF((X(I,3).GT.0.0D0).AND.(X(I,4).GT.0.0D0)) THEN
              IRXN=1
      ENDIF

      XH2SRXN(I)=X(I,3)
2100  XSO2RXN(I)=X(I,4)

      IF(IRXN.EQ.0) GOTO 501

C TOP OF RXN SOLVER LOOP
11000 CONTINUE

C ZERO THE BAND X AND Y MATRICES
      DO 2110 I=1,2
      DO 2110 J=1,2
      XBND(I,J)=0.0D0
2110  YBND(I,J)=0.0D0

C STEP THROUGH EQUATIONS FOR EACH STAGE
      DO 2150 J=1,NSTAGE

C ZERO THE COEFFICIENT MATRICES
      DO 2120 IR=1,2
      GBND(IR)=0.0D0
      DO 2120 IC=1,2
      ABND(IR,IC)=0.0D0
      BBND(IR,IC)=0.0D0
2120  DBND(IR,IC)=0.0D0

      IF(J.EQ.1) GOTO 2130

      ABND(1,1)=-V(J-1)*K(J-1,3)
      ABND(2,2)=-V(J-1)*K(J-1,4)
      GBND(1)=V(J-1)*K(J-1,3)*XH2SRXN(J-1)
      GBND(2)=V(J-1)*K(J-1,4)*XSO2RXN(J-1)
```

```
2130    BBND(1,1)=L(J)/RS(J)+V(J)*K(J,3)-ALPHA(3)*RXNTERM*XSO2RXN(J)
        BBND(1,2)=-ALPHA(3)*RXNTERM*XH2SRXN(J)
        BBND(2,2)=L(J)/RS(J)+V(J)*K(J,4)-ALPHA(4)*RXNTERM*XH2SRXN(J)
        BBND(2,1)=-ALPHA(4)*RXNTERM*XSO2RXN(J)

        IF(J.EQ.NSTAGE) GOTO 2140

        DBND(1,1)=-L(J+1)
        DBND(2,2)=-L(J+1)
        GBND(1)=GBND(1)+L(J+1)*XH2SRXN(J+1)
        GBND(2)=GBND(2)+L(J+1)*XSO2RXN(J+1)

2140    GBND(1)=GBND(1)+F(J,1)*ZF(J,3)-(V(J)*K(J,3)+L(J)/RS(J))
     1 *XH2SRXN(J)+ALPHA(3)*RXNTERM*XH2SRXN(J)*XSO2RXN(J)
        GBND(2)=GBND(2)+F(J,1)*ZF(J,4)-(V(J)*K(J,4)+L(J)/RS(J))
     1 *XSO2RXN(J)+ALPHA(4)*RXNTERM*XH2SRXN(J)*XSO2RXN(J)

        CALL BAND(J)

2150    CONTINUE

C UPDATE VALUES

        DO 2160 J=1,NSTAGE

        IF((XH2SRXN(J)+CBND(1,J)).LT.0.0D0) THEN
                XH2SRXN(J)=XH2SRXN(J)/2.0D0
        ELSE
                IF((XH2SRXN(J)+CBND(1,J)).GT.1.0D0) THEN
                        XH2SRXN(J)=(XH2SRXN(J)+1.0D0)/2.0D0
                ELSE
                        XH2SRXN(J)=XH2SRXN(J)+CBND(1,J)
                ENDIF
        ENDIF

        IF((XSO2RXN(J)+CBND(2,J)).LT.0.0D0) THEN
                XSO2RXN(J)=XH2SRXN(J)/2.0D0
```

```
      ELSE
            IF((XSO2RXN(J)+CBND(2,J)).GT.1.0D0) THEN
                  XSO2RXN(J)=(XSO2RXN(J)+1.0D0)/2.0D0
            ELSE
                  XSO2RXN(J)=XSO2RXN(J)+CBND(2,J)
            ENDIF
      ENDIF

2160  CONTINUE

C PERFORM CONVERGENCE CHECK
      DO 2170 J=1,NSTAGE
      IF(XH2SRXN(J).EQ.0.0D0) GOTO 2165
      IF(DABS(CBND(1,J)/XH2SRXN(J)).GT.1.0D-3) GOTO 2190
2165  IF(XSO2RXN(J).EQ.0.0D0) GOTO 2170
      IF(DABS(CBND(2,J)/XSO2RXN(J)).GT.1.0D-3) GOTO 2190
2170  CONTINUE

C RXN LOOP CONVERGED, UPDATE MAIN LOOP VALUES

      DO 2180 J=1,NSTAGE
      X(J,3)=XH2SRXN(J)
2180  X(J,4)=XSO2RXN(J)

C EXIT TO MAIN LOOP
      GOTO 501

C UPDATE ITERATION COUNT
2190  ITERRXN=ITERRXN+1

      WRITE(6,94000) ITERRXN
94000 FORMAT('+','IN RXN LOOP, ITERATION=',I3)

      IF(ITERRXN.LT.IMAXA) GOTO 11000
      WRITE(6,5250)
      WRITE(10,5250)
5250  FORMAT(X,'ERROR! NON-CONVERGENCE IN RXN LOOP IN ABSORBER')
```

```
C FALL THROUGH ERROR AND HOPE FOR THE BEST

C SET VARIABLE COUNT BACK TO MAIN LOOP VALUE
501     NVAR=NCOMP+3

C WRITE A BLANK LINE TO MAKE OUTPUT PRETTY
        WRITE(6,5260)
5260    FORMAT(X)

C SET RXN TERMS AS SIDE DRAW FOR EVERY STAGE

        DO 400 J=1,NSTAGE
400     R(J)=X(J,3)*X(J,4)*RHOM*RHOM*VLIQT*R2CONST

C- ZERO THE XBND AND YBND MATRICES AT THE START OF AN ITERATION.
        DO 500 I=1,NCOMP+3
        DO 500 J=1,NCOMP+3
        XBND(I,J)=0.0D0
500     YBND(I,J)=0.0D0

C- BEGIN INNER LOOP. PERFORM J CALLS TO BAND TO SOLVE THE COUPLED EQUATIONS

        DO 10000 J=1,NSTAGE

C- FIRST ZERO THE COEFFICIENT MATRICES
        DO 525 IR=1,NCOMP+3
        GBND(IR)=0.0D0
        DO 525 IC=1,NCOMP+3
        ABND(IR,IC)=0.0D0
        BBND(IR,IC)=0.0D0
525     DBND(IR,IC)=0.0D0

C- SEE IF WE ARE AT THE BOTTOM OF THE COLUMN, IF SO ABND IS ALL ZEROS

        IF(J.EQ.1) GOTO 2000
```

```
C- FILL ABND WITH J-1 VALUES OF COEFFICIENTS
C- THE ORDER OF VARIABLES IN A GIVEN ROW IS T,V,L,ALL X(IC)
C- THE ORDER OF EQUATIONS IS ALL MASS BAL., HEAT BAL., EQUILIB., MOLE FRACT.
C- VARIABLES: IR=ROW INDEX, IC=COLUMN INDEX, IX= COMPONENT INDEX


C- DO THE MASS BALANCES
      DO 530 IR=1,NCOMP
      IX=IR
      IC=IR+3
      ABND(IR,1)=-V(J-1)*X(J-1,IX)*DKDT(J-1,IX)
      ABND(IR,2)=-K(J-1,IX)*X(J-1,IX)
530   ABND(IR,IC)=-V(J-1)*K(J-1,IX)

C- DO THE HEAT BALANCE
      IR=NCOMP+1
      SUM0=0.000
      SUM1=0.000
      DO 540 IX=1,NCOMP
      SUM0=SUM0+X(J-1,IX)*(HV(J-1,IX)*DKDT(J-1,IX)+K(J-1,IX)*
    1 DHVDT(J-1,IX))
540   SUM1=SUM1+HV(J-1,IX)*K(J-1,IX)*X(J-1,IX)
      ABND(IR,1)=-V(J-1)*SUM0
      ABND(IR,2)=-SUM1
      DO 550 IX=1,NCOMP
      IC=IX+3
550   ABND(IR,IC)=-V(J-1)*HV(J-1,IX)*K(J-1,IX)

C- AT J-1 THE EQUILIBRIUM AND MOLE FRACTION EQNS. ARE ALL ZEROS

C- FILL IN THE BBND MATRIX AT J

C- MASS BALANCES
2000  DO 560 IR=1,NCOMP
      IX=IR
      IC=IR+3
      BBND(IR,1)=V(J)*X(J,IX)*DKDT(J,IX)
```

```
      BBND(IR,2)=K(J,IX)*X(J,IX)
      BBND(IR,3)=X(J,IX)/RS(J)
560   BBND(IR,IC)=V(J)*K(J,IX)+L(J)/RS(J)

C- HEAT BALANCE
      IR=NCOMP+1
      SUM0=0.0D0
      SUM1=0.0D0
      SUM2=0.0D0
      SUM3=0.0D0
      DO 570 IX=1,NCOMP
      SUM0=SUM0+X(J,IX)*(HV(J,IX)*DKDT(J,IX)+K(J,IX)*DHVDT(J,IX))
      SUM1=SUM1+X(J,IX)*DHLDT(J,IX)
      SUM2=SUM2+HV(J,IX)*K(J,IX)*X(J,IX)
570   SUM3=SUM3+HL(J,IX)*X(J,IX)
      BBND(IR,1)=V(J)*SUM0+L(J)*SUM1/RS(J)
      BBND(IR,2)=SUM2
      BBND(IR,3)=SUM3/RS(J)

      DO 580 IX=1,NCOMP
      IC=IX+3
580   BBND(IR,IC)=V(J)*HV(J,IX)*K(J,IX)+L(J)*HL(J,IX)/RS(J)

C- EQUILIB. EQN.
      IR=IR+1

      SUM0=0.0D0
      DO 590 IX=1,NCOMP
590   SUM0=SUM0+X(J,IX)*DKDT(J,IX)
      BBND(IR,1)=SUM0

      DO 600 IX=1,NCOMP
      IC=IX+3
600   BBND(IR,IC)=K(J,IX)-1.0D0

C- MOLE FRACTION
      IR=IR+1
```

```
      DO 610 IX=1,NCOMP
      IC=IX+3
610   BBND(IR,IC)=1.000


C- CHECK FOR LAST STAGE. IF SO THEN J+1 TERMS ARE ALL ZERO
      IF(J.EQ.NSTAGE) THEN
        GOTO 3000
      ENDIF

C- FILL IN THE DBND AT J+1

C- MASS BALANCES
      DO 620 IR=1,NCOMP
      IX=IR
      IC=IR+3
      DBND(IR,3)=-X(J+1,IX)
620   DBND(IR,IC)=-L(J+1)

C- HEAT BAL.
      IR=NCOMP+1
      SUM0=0.000
      SUM1=0.000
      DO 630 IX=1,NCOMP
      SUM0=SUM0+X(J+1,IX)*DHLDT(J+1,IX)
630   SUM1=SUM1+HL(J+1,IX)*X(J+1,IX)
      DBND(IR,1)=-L(J+1)*SUM0
      DBND(IR,3)=-SUM1
      DO 640 IX=1,NCOMP
      IC=IX+3
640   DBND(IR,IC)=-L(J+1)*HL(J+1,IX)

C***********************************************************************
C- FILL IN THE RIGHT HAND SIDE OF THE EQNS. (I.E. GBND)

C- CHECK FOR J=1 CONDITION, JUMP TO ALTERNATE GBND CALC IF SO
      IF(J.EQ.1) GOTO 3020
```

```
C- MASS BALANCES FOR 1<J<NJ
        DO 645 IR=1,NCOMP
        IX=IR
645     GBND(IR)=F(J,1)*ZF(J,IX)+ALPHA(IX)*R(J)
     1  +V(J-1)*K(J-1,IX)*X(J-1,IX)-(V(J)*K(J,IX)+L(J)/RS(J))*X(J,IX)
     2  +L(J+1)*X(J+1,IX)

C- HEAT BALANCE FOR 1<J<NJ
        IR=NCOMP+1
        SUM0=0.0D0
        SUM1=0.0D0
        SUM2=0.0D0
        SUM3=0.0D0
        DO 650 IX=1,NCOMP
        SUM0=SUM0+HV(J-1,IX)*K(J-1,IX)*X(J-1,IX)
        SUM1=SUM1+HV(J,IX)*K(J,IX)*X(J,IX)
        SUM2=SUM2+HL(J,IX)*X(J,IX)
650     SUM3=SUM3+HL(J+1,IX)*X(J+1,IX)

        GBND(IR)=F(J,1)*F(J,3)+V(J-1)*SUM0-V(J)*SUM1-L(J)*SUM2/RS(J)+
     1  L(J+1)*SUM3+QH(J)

C JUMP TO EQUILIBRIUM AND MOLE FRACTION BALANCES
        GOTO 3010

C- MASS BALANCES FOR J=NJ
3000    DO 655 IR=1,NCOMP
        IX=IR
655     GBND(IR)=F(J,1)*ZF(J,IX)+ALPHA(IX)*R(J)
     1  +V(J-1)*K(J-1,IX)*X(J-1,IX)-(V(J)*K(J,IX)+L(J)/RS(J))*X(J,IX)

C- HEAT BALANCE FOR J=NJ
        IR=NCOMP+1
        SUM0=0.0D0
        SUM1=0.0D0
        SUM2=0.0D0
```

```
      DO 660 IX=1,NCOMP
      SUM0=SUM0+HV(J-1,IX)*K(J-1,IX)*X(J-1,IX)
      SUM1=SUM1+HV(J,IX)*K(J,IX)*X(J,IX)
660   SUM2=SUM2+HL(J,IX)*X(J,IX)

      GBND(IR)=F(J,1)*F(J,3)+V(J-1)*SUM0-V(J)*SUM1-L(J)*SUM2/RS(J)+
     1 QH(J)

C JUMP TO EQUILIB AND MOLE FRACTION BALANCES

      GOTO 3010

C CALCULATE MASS AND HEAT BALANCES FOR J=1
3020  DO 665 IR=1,NCOMP
      IX=IR
665   GBND(IR)=F(J,1)*ZF(J,IX)+ALPHA(IX)*R(J)
     1 -(V(J)*K(J,IX)+L(J)/RS(J))*X(J,IX)+L(J+1)*X(J+1,IX)

C- HEAT BALANCE FOR J=NJ
      IR=NCOMP+1
      SUM0=0.0D0
      SUM1=0.0D0
      SUM2=0.0D0
      SUM3=0.0D0
      DO 670 IX=1,NCOMP
      SUM1=SUM1+HV(J,IX)*K(J,IX)*X(J,IX)
      SUM2=SUM2+HL(J,IX)*X(J,IX)
670   SUM3=SUM3+HL(J+1,IX)*X(J+1,IX)

      GBND(IR)=F(J,1)*F(J,3)-V(J)*SUM1-L(J)*SUM2/RS(J)+L(J+1)*SUM3+
     1 QH(J)

C- EQUILIB. EQN.
3010  IR=IR+1
      SUM0=0.0D0
      DO 675 IX=1,NCOMP
675   SUM0=SUM0+X(J,IX)*(1.0D0-K(J,IX))
```

```
      GBND(IR)=SUMO

C- MOLE FRACTION
      IR=IR+1
      SUMO=0.000
      DO 680 IX=1,NCOMP
680   SUMO=SUMO+X(J,IX)
      GBND(IR)=1.000-SUMO

C- IF A CONDENSER IS PRESENT AND THIS IS THE TOP STAGE THEN
C- THE EQUATION IS SIMPLY T(J)=TC
      IF((J.EQ.NSTAGE).AND.(TC.NE.0.D0)) THEN
        IR=NCOMP+1
        DO 685 IC=1,NCOMP+3
        ABND(IR,IC)=0.D0
        BBND(IR,IC)=0.D0
685     DBND(IR,IC)=0.D0
        BBND(IR,1)=1.000
        GBND(IR)=TC-T(J)
      ENDIF

C- MATRICES ARE SET, CALL BAND
      WRITE(6,53000) UNIT,ITKEY,ITER,J
53000 FORMAT('+','UNIT=',F5.2,'  FLOWSHEET ITER=',I3,'  ITER=',I3,
    1 '  STAGE = ',I2)
      CALL BAND(J)

C- DO NEXT J

10000 CONTINUE


C-EQNS HAVE BEEN SOLVED, UPDATE VARIABLES, SET NEW GUESSES
C-IF A NEGATIVE VALUE IS OBTAINED THE NEW VALUE BECOMES THE
C-PREVIOUS VALUE AVERAGED WITH 0.
C-IF A NEW MOLE FRACTION IS GREATER THAN 1.0 THEN THE NEW VALUE
C-IS THE OLD VALUE AVERAGED WITH 1.0
```

```
      WRITE(6,50500)
50500 FORMAT(' EQNS SOLVED, NOW UPDATING VALUES')

      DO 700 J=1,NSTAGE
      TTEMP=T(J)+CBND(1,J)
C LIMIT CHANGE IN T TO 20 PERCENT
      IF(DABS((TTEMP-T(J))/T(J)).GT.0.2D0) THEN
              CBND(1,J)=DSIGN(0.2D0*T(J),CBND(1,J))
      ENDIF

      IF((T(J)+CBND(1,J)).LT.0.0D0) THEN
              T(J)=T(J)/2.0D0
      ELSE
              T(J)=T(J)+CBND(1,J)
      ENDIF

      VTEMP=V(J)+CBND(2,J)

      IF(DABS((VTEMP-V(J))/V(J)).GT.0.2D0) THEN
              CBND(2,J)=DSIGN(0.2D0*V(J),CBND(2,J))
      ENDIF

      IF((V(J)+CBND(2,J)).LT.0.0D0) THEN
              V(J)=V(J)/2.0D0
      ELSE
              V(J)=V(J)+CBND(2,J)
      ENDIF

      RLTEMP=L(J)+CBND(3,J)

      IF(DABS((RLTEMP-L(J))/L(J)).GT.0.2D0) THEN
              CBND(3,J)=DSIGN(0.2D0*L(J),CBND(3,J))
      ENDIF

      IF((L(J)+CBND(3,J)).LT.0.0D0) THEN
              L(J)=L(J)/2.0D0
```

```
      ELSE
              L(J)=L(J)+CBND(3,J)
      ENDIF

      DO 700 IX=1,NCOMP
      IC=IX+3
      IF((X(J,IX)+CBND(IC,J)).LT.0.0D0) THEN
              X(J,IX)=X(J,IX)/2.0D0
      ELSE
              IF((X(J,IX)+CBND(IC,J)).GT.1.0D0) THEN
                      X(J,IX)=(X(J,IX)+1.0D0)/2.0D0
              ELSE
                      X(J,IX)=X(J,IX)+CBND(IC,J)
              ENDIF
      ENDIF

C CALCULATE NEW Y VALUES

      Y(J,IX)=X(J,IX)*K(J,IX)


C REMOVE COMPONENT 'NOISE'
      IF(X(J,IX).LT.1.0D-09) THEN
      X(J,IX)=0.0D0
      ENDIF

700   CONTINUE


      WRITE(6,50600)
50600 FORMAT(' PERFORMING CONVERGENCE CHECK')

      DO 40200 I=1,NSTAGE
40200 WRITE(6,40000) T(I),V(I),L(I),XH2SRXN(I),XSO2RXN(I)
40000 FORMAT(X,'T=',F10.4,' V=',D10.4,' L=',D10.4,' XRH2S=',D10.4,
     1 ' XRSO2=',D10.4)
```

```
C       DO 40300 I=1,NSTAGE
C40300  WRITE(6,40100) (X(I,IX),IX=1,NCOMP)
C40100  FORMAT(' ',4D20.10)

C- CHECK FOR CONVERGENCE -- CONVERGE LIQ. MOLE FRACTS
        DO 810 J=1,NSTAGE
        DO 810 IX=1,NCOMP
        IC=IX+3
        IF(X(J,IX).EQ.0.0D0) GOTO 810
        IF(DABS(CBND(IC,J)/X(J,IX)).GT.1.0D-3) THEN
                WRITE(6,54000) IX,J,X(J,IX)
54000   FORMAT(X,'NOT CONVERGED IX=',I4,' STAGE=',I4,' X=',D20.10)
                GOTO 4000
        ENDIF
810     CONTINUE

C- FELL THROUGH CHECK LOOP SO MUST HAVE CONVERGED
        WRITE(10,40810) UNIT,ITER
40810   FORMAT(X,' UNIT=',F5.2,' ITER=',I4,//)

        WRITE(10,60001)
        DO 60000 I=1,NSTAGE
60000   WRITE(10,40010) T(I),V(I),L(I)

        WRITE(10,40520)
        WRITE(10,60050)
        WRITE(10,40500)
        DO 60020 I=1,NSTAGE
60020   WRITE(10,40510) I,(X(I,IX),IX=1,NCOMP)

        WRITE(10,40520)
        WRITE(10,60060)
        WRITE(10,40500)
        DO 60090 I=1,NSTAGE
60090   WRITE(10,40510) I,(X(I,IX)*K(I,IX),IX=1,NCOMP)

        WRITE(10,40520)
```

```
      WRITE(10,60070)
      WRITE(10,40500)
      DO 60100 I=1,NSTAGE
60100 WRITE(10,40510) I,(X(I,IX)*L(I),IX=1,NCOMP)

      WRITE(10,40520)
      WRITE(10,60080)
      WRITE(10,40500)
      DO 60110 I=1,NSTAGE
60110 WRITE(10,40510) I,(X(I,IX)*K(I,IX)*V(I),IX=1,NCOMP)

60001 FORMAT(X,' ABSORBER OUTPUT DATA')
60050 FORMAT(X,' LIQUID COMPONENT MOLE FRACTIONS')
60060 FORMAT(X,' VAPOR COMPONENT MOLE FRACTIONS')
60070 FORMAT(X,' LIQUID COMPONENT FLOWS')
60080 FORMAT(X,' VAPOR COMPONENT FLOWS')
40010 FORMAT(X,'T=',F10.4,' V=',D10.4,' L=',D10.4)
40500 FORMAT(1X,'SG#',X,3X,'SOLV',3X,3X,'H2O',4X,3X,'H2S',4X,3X,'SO2',
    1 4X,3X,'SULD',3X,3X,'CO2',4X,3X,'N2',5X,3X,'C1',5X,3X,'H2',5X,
    2 3X,'CO',5X,3X,'C2',5X,3X,'C3',5X,3X,'C4',5X,3X,'C5+')
40510 FORMAT(1X,I3,X,23(D9.4,X),D15.9,X,F4.1)
40520 FORMAT(X,' ')

C WRITE THE COLUMN PROFILE TO THE .PRO FILE
      OPEN(11,FILE=PNAME)
      DO 61000 J=1,NSTAGE
      WRITE(11,70000) T(J),V(J),L(J)
      WRITE(11,70030) (X(J,IX),IX=1,NCOMP)
61000 WRITE(11,70030) (Y(J,IX),IX=1,NCOMP)


70000 FORMAT(3D20.10)
70030 FORMAT(14D17.10)

      GOTO 5000

C- NOT CONVERGED, DO NEXT ITERATION
```

```
4000    ITER=ITER+1
C PRETTY PRINT STATEMENT
        WRITE(6,48200)
48200   FORMAT(X)

        IF(ITER.LT.IMAXA) GOTO 1000


C CHECK TO SEE IF SULFUR PRECIPITATED ON ANY OF THE TRAYS
C IF SOLUBILITY IS EXCEEDED FLAG IT IN THE OUTPUT
5000    DO 5010 J=1,NSTAGE
        XS=SULFUR(T(J))
C       WRITE(6,60200) J,(X(J,5)/XS)*100.0D0,XS
        WRITE(10,60200) J,(X(J,5)/XS)*100.0D0,XS
60200   FORMAT(X,' % SULFUR SATURATION ON TRAY ',I4,' =',D20.10,
     1 ' (SOLMAX=',D20.10,')')
5010    CONTINUE

C SET VALUES FOR RETURN STREAMS
C SET COMPOSITIONS

        DO 910 I=1,NCOMP
        STVAP(I)=X(NSTAGE,I)*K(NSTAGE,I)
        STLIQ(I)=X(1,I)
910     STLIQC(I)=X(NSTAGE,I)

C-RESET NUMBER OF COMPONENTS TO MAIN LOOP VALUE
        NCOMP=NCOMPOLD


C SET FLOW, TEMP, PRESSURE, ENTHALPY, AND PHASE

        STVAP(NCOMP+1)=V(NSTAGE)
        STLIQ(NCOMP+1)=L(1)
        STLIQC(NCOMP+1)=L(NSTAGE)*FS/(1.D0-FS)
```

```
      STVAP(NCOMP+2)=T(NSTAGE)
      STLIQ(NCOMP+2)=T(1)
      STLIQC(NCOMP+2)=T(NSTAGE)

      STVAP(NCOMP+3)=P
      STLIQ(NCOMP+3)=P
      STLIQC(NCOMP+3)=P

      HTV=0.0D0
      HTL=0.0D0
      HTLC=0.0D0

      DO 920 I=1,NCSUB
      HTV=HTV+HV(NSTAGE,I)*X(NSTAGE,I)*K(NSTAGE,I)
      HTL=HTL+HL(1,I)*X(1,I)
920   HTLC=HTLC+HL(NSTAGE,I)*X(NSTAGE,I)

      STVAP(NCOMP+4)=HTV
      STLIQ(NCOMP+4)=HTL
      STLIQC(NCOMP+4)=HTLC

      STVAP(NCOMP+5)=2
      STLIQ(NCOMP+5)=1
      STLIQC(NCOMP+5)=1

C CLOSE OUTPUT DATA FILES
      CLOSE(10)
      CLOSE(11)

      RETURN
      END
```

```
      SUBROUTINE FSPLIT(NC,STREAMI,STREAM1,STREAM2,FLOW)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 STREAMI(25),STREAM1(25),STREAM2(25)

C FLOW SPLITTER
C
C LAST UPDATE: 04/21/86

C SET RETURN VALUES FOR ALL ZI,FLOW,TEMP,PRESS,ENTHALPY,PHASE
      DO 10 I=1,NC+5
      STREAM1(I)=STREAMI(I)
10    STREAM2(I)=STREAMI(I)

C SPLIT STREAM BASED ON FLOW - FLOW IS AMOUNT WHICH GOES TO STREAM1
C IF FLOW IS GREATER THAN FEED, SEND ALL TO STREAM1
      IF(FLOW.GT.STREAMI(NC+1)) THEN
              STREAM1(NC+1)=STREAMI(NC+1)
              STREAM2(NC+1)=0.0D0
      ELSE
              STREAM1(NC+1)=FLOW
              STREAM2(NC+1)=STREAMI(NC+1)-FLOW
      ENDIF

      RETURN
      END
```

```
      SUBROUTINE PHFLASH(NCOMP,STREAMF,PFLASH,STRL,STRV,
     1EPSILON,IPHASE,IMAX,ERROR)

C ISENTHALPIC OR ADIABATIC FLASH CALCULATION
C FOR MATHEMATICAL DETAILS SEE FOR EXAMPLE KING, PGS. 81-90.
C
C LAST UPDATE: 04/21/86

      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER*2 ERROR
      INTEGER*2 NCOMP,IMAX,ITER
      REAL*8 STREAMF(25),PFEED,TFEED,F,ZF(20),HF,PFLASH,
     1 TFLASH,V,L,X(20),Y(20),EPSILON
      REAL*8 STRL(25),STRV(25)
      REAL*8 TBUB,TDEW,VF
      REAL*8 K(20),DKDT(20)
      REAL*8 OLDF,OLDG,NEWF,NEWG,SUM0,SUM1,ADD0,ADD1,OGCHK,NGCHK
      REAL*8 FDREV,FDERT,GDERV,GDERT
      REAL*8 HL(20),HV(20),DHLDT(20),DHVDT(20),HTL,HTV,HT
      REAL*8 HFI(20),DHFIDT(20),HTD,HTB

C SET ERROR RETURN TO NO CONVERGENCE. SET IPHASE TO MIXED
C  LIQUID AND VAPOR
      ERROR=1
      IPHASE=3

C INITIALIZE ITERATION COUNT
      ITER=0

C SET VALUES FOR MOLE FRACTIONS - FIRST GUESS
      DO 5 I=1,NCOMP
      ZF(I)=STREAMF(I)
      X(I)=ZF(I)
5     Y(I)=ZF(I)

C SET LOCAL STREAM VARIABLES
      HF=STREAMF(NCOMP+4)
```

```
      PFEED=STREAMF(NCOMP+3)
      TFEED=STREAMF(NCOMP+2)
      F=STREAMF(NCOMP+1)

C CALCULATE BUBBLE AND DEW POINTS
      CALL BUBDEW(NCOMP,TFEED,X,Y,TBUB,TDEW,PFLASH,EPSILON,2)

C CALCULATE ENTHALPY OF THE FEED AT THE BUBBLE AND DEW POINTS
      CALL HEAT(0,1,NCOMP,TBUB,PFLASH,X,HL,HTB)
      CALL HEAT(0,2,NCOMP,TDEW,PFLASH,X,HV,HTD)


C SEE IF FEED IS ALL SUPERHEATED VAPOR OR SUBCOOLED LIQUID

      IF(HF.GT.HTD) THEN
       DO 10 I=1,NCOMP
10     X(I)=0.0D0
       V=F
       L=0.0D0
       IPHASE=2
       GOTO 175
      ENDIF

      IF(HF.LT.HTB) THEN
       DO 15 I=1,NCOMP
15     Y(I)=0.0D0
       L=F
       V=0.0D0
       IPHASE=1
       GOTO 175
      ENDIF

C*********************************************************
C HAVE MIXTURE, PERFORM FLASH CALCULATION
C*********************************************************
```

```
C MAKE FIRST GUESS FOR TFLASH AND V/F
      VF=(HF-HTB)/(HTD-HTB)
      TFLASH=TBUB+VF*(TDEW-TBUB)

C CALCULATE K
      CALL KVAL(0,NCOMP,TFLASH,PFLASH,X,Y,K)

C CALCULATE HL,HV
      CALL HEAT(0,1,NCOMP,TFLASH,PFLASH,X,HL,HTL)
      CALL HEAT(0,2,NCOMP,TFLASH,PFLASH,Y,HV,HTV)

C CALCULATE F AND G
      OLDF=0.0
      SUM0=0.0D0
      SUM1=0.0D0
      DO 20 I=1,NCOMP
      OLDF=OLDF+(ZF(I)*(K(I)-1.D0))/(VF*(K(I)-1.D0)+1.0D0)
      SUM0=SUM0+(HV(I)*ZF(I)*K(I))/(VF*(K(I)-1.D0)+1.0D0)
20    SUM1=SUM1+(HL(I)*ZF(I)/(VF*(K(I)-1.D0)+1.0D0))
      OLDG=VF*SUM0+(1.0D0-VF)*SUM1-HF

      OLDG=OLDG/HF

C CALCULATE DERIVATIVES DKDT,DHLDT,DHVDT
25    CALL KVAL(1,NCOMP,TFLASH,PFLASH,X,Y,DKDT)
      CALL HEAT(1,1,NCOMP,TFLASH,PFLASH,X,DHLDT,HT)
      CALL HEAT(1,2,NCOMP,TFLASH,PFLASH,Y,DHVDT,HT)

C CALCULATE DF/DV,DF/DT,DG/DV,DG/DT
      FDERV=0.0D0
      FDERT=0.0D0
      GDERV=0.0D0
      GDERT=0.0D0
      ADD0=0.0D0
      ADD1=0.0D0
      DO 30 I=1,NCOMP
      FDERV=FDERV-(ZF(I)*((K(I)-1.D0)**2))/((VF*(K(I)-1.D0)+1.0D0)**2)
```

```
      FDERT=FDERT+(ZF(I)*DKDT(I))/((VF*(K(I)-1.D0)+1.D0)**2)
      GDERV=GDERV+(ZF(I)*K(I)*(HV(I)-HL(I)))/((VF*(K(I)-1.D0)
     1 +1.D0)**2)
      ADD0=VF*K(I)*DHVDT(I)+(1.D0-VF)*DHLDT(I)
      ADD1=(VF*(1.D0-VF)*(HV(I)-HL(I))*DKDT(I))/(VF*(K(I)-1.D0)
     1 +1.D0)
      ADD2=ZF(I)*(ADD0+ADD1)/(VF*(K(I)-1.D0)+1.D0)
30    GDERT=GDERT+ADD2

      GDERV=GDERV/HF
      GDERT=GDERT/HF

C CALCULATE NEW TFLASH AND V
      TFLASH=TFLASH+(FDERV*OLDG-GDERV*OLDF)/(FDERT*GDERV-FDERV*GDERT)
      VFOLD=VF
      VF=VF+(GDERT*OLDF-FDERT*OLDG)/(FDERT*GDERV-FDERV*GDERT)

      IF(VF.LT.0.D0) THEN
              VF=VFOLD/2.D0
      ENDIF

      IF(VF.GT.1.D0) THEN
              VF=(VFOLD+1.D0)/2.D0
      ENDIF

      WRITE(6,20000) TFLASH,VF
20000 FORMAT(X,'TFLASH=',D20.10,' VF=',D20.10)


C CALCULATE NEW X,Y
      V=F*VF
      L=F-V

      DO 40 I=1,NCOMP
      X(I)=ZF(I)/((K(I)-1.D0)*VF+1.D0)
40    Y(I)=X(I)*K(I)
```

```
C CALCULATE NEW K VALUES
        CALL KVAL(0,NCOMP,TFLASH,PFLASH,X,Y,K)

C CALCULATE NEW ENTHALPIES
        CALL HEAT(0,1,NCOMP,TFLASH,PFLASH,X,HL,HTL)
        CALL HEAT(0,2,NCOMP,TFLASH,PFLASH,Y,HV,HTV)

C CALCULATE NEW F AND G FUNCTIONS
        NEWF=0.0
        SUM0=0.000
        SUM1=0.000
        DO 50 I=1,NCOMP
        NEWF=NEWF+(ZF(I)*(K(I)-1.D0))/(VF*(K(I)-1.D0)+1.000)
        SUM0=SUM0+(HV(I)*ZF(I)*K(I))/(VF*(K(I)-1.D0)+1.000)
50      SUM1=SUM1+(HL(I)*ZF(I)/(VF*(K(I)-1.D0)+1.000))
        NEWG=VF*SUM0+(1.000-VF)*SUM1-HF

        NEWG=NEWG/HF

        ITER=ITER+1

C NORMALIZE G FOR CONVERGENCE CHECK
        OGCHK=OLDG
        NGCHK=NEWG

C CHECK FOR CONVERGENCE
C       IF((DABS(NEWF)+DABS(NGCHK)).GT.(DABS(OLDF)+DABS(OGCHK)))
C     1 GOTO 100
        IF(ITER.GT.IMAX) GOTO 100
        IF((DABS(NEWF)+DABS(NGCHK)).GT.EPSILON) THEN
                OLDF=NEWF
                OLDG=NEWG
                GOTO 25
        ELSE

C CONVERGED, SHOW CONVERGED, HAVE MIXTURE AND THEN EXIT
                ERROR=2
```

```
            IPHASE=3
            GOTO 100
      ENDIF

C*************************************************************
C IF ALL LIQUID OR ALL VAPOR ITERATE TO DETERMINE EXIT TEMP
C*************************************************************

C SET MOLE FRACTIONS FOR USE IN THERMO SUBROUTINES
175   DO 150 I=1,NCOMP
150   X(I)=STREAMF(I)

C SET FIRST GUESS FOR TEMPERATURE

200   TFLASH=TFEED
225   TOUT=TFLASH
      CALL HEAT(0,IPHASE,NCOMP,TOUT,PMIX,X,HFI,HT)
      CALL HEAT(1,IPHASE,NCOMP,TOUT,PMIX,X,DHFIDT,DHT)

C CALCULATE TOTAL ENTHALPIES AND DERIVATIVES
      HT1=HT*STREAMF(NCOMP+1)
      DHT=DHT*STREAMF(NCOMP+1)

C CALCULATE NEW T VALUE
      TFLASH=TOUT-(HT1-(STREAMF(NCOMP+1)*STREAMF(NCOMP+4)))/DHT

      ITER=ITER+1

C CHECK FOR CONVERGENCE
      IF(DABS(TFLASH-TOUT).LE.EPSILON) GOTO 270
C CHECK FOR MAXIMUM NUMBER OF ITERATIONS
      IF(ITER.GT.IMAX) GOTO 100
       GOTO 225

C SHOW CONVERGED CALC, BUT FLAG THE FACT THAT STREAM IS ALL LIQUID OR VAPOR
270    ERROR=3
```

```
C SET RETURN STREAM VALUES
100     DO 300 I=1,NCOMP
        STRL(I)=X(I)
300     STRV(I)=Y(I)

C FLOW, TEMP, PRESSURE, ENTHALPY, PHASE
        STRL(NCOMP+1)=L
        STRV(NCOMP+1)=V

        STRL(NCOMP+2)=TFLASH
        STRV(NCOMP+2)=TFLASH

        STRL(NCOMP+3)=PFLASH
        STRV(NCOMP+3)=PFLASH

        IF(V.EQ.0.0D0) THEN
        HTL=HT
        HTV=0.0D0
        ENDIF

        IF(L.EQ.0.0D0) THEN
        HTL=0.0D0
        HTV=HT
        ENDIF

        STRL(NCOMP+4)=HTL
        STRV(NCOMP+4)=HTV

        STRL(NCOMP+5)=1
        STRV(NCOMP+5)=2

        RETURN
        END
```

```
      SUBROUTINE XCHG(NC,STREAMHI,STREAMCI,STREAMHO,STREAMCO,
     1 UA,EPS,IMAX,ERR)

C HEAT EXCHANGER SUBROUTINE
C
C LAST UPDATE: 06/11/86

C WARNING: THIS ROUTINE WILL NOT CONVERGE FOR VERY LARGE VALUES
C  OF UA (UA>10000 KW/DEG K, I.E. UNREALISTIC VALUES). THE
C  MAXIMUM "SAFE" VALUE IS DETERMINED BY THE STEP SIZE USED
C  TO CALCULATE THE DEVIATIVES WITH RESPECT TO TEMP.

      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 XH(20),XC(20)
      REAL*8 CP(20)
      REAL*8 STREAMHI(25),STREAMCI(25)
      REAL*8 STREAMHO(25),STREAMCO(25)
      REAL*8 STRL(25),STRV(25)

      INTEGER*2 PHASEC,PHASEH
      INTEGER*2 ERR

C SET TEMPERATURE INCREMENT FOR DERIVATIVES
      DELT=0.01D0

C SET RETURN FOR NO ERROR
      ERR=0

C ZERO ITERATION COUNT
      ITER=0

C SET MOLE FRACTIONS FOR CALCULATION AND OUTLET STREAMS
      DO 10 I=1,NC
      XH(I)=STREAMHI(I)
      STREAMHO(I)=XH(I)
      XC(I)=STREAMCI(I)
10    STREAMCO(I)=XC(I)
```

```
C SET OUTLET FLOWS AND PRESSURES
      STREAMHO(NC+1)=STREAMHI(NC+1)
      STREAMHO(NC+3)=STREAMHI(NC+3)

      STREAMCO(NC+1)=STREAMCI(NC+1)
      STREAMCO(NC+3)=STREAMCI(NC+3)

C SET LOCAL VALUES FOR STREAM PRESSURES, PHASES, TEMPERATURES, AND ENTHALPIES
      PRH=STREAMHI(NC+3)
      PHASEH=INT(STREAMHI(NC+5))

      PRC=STREAMCI(NC+3)
      PHASEC=INT(STREAMCI(NC+5))

      TCI=STREAMCI(NC+2)
      THI=STREAMHI(NC+2)

      HCI=STREAMCI(NC+1)*STREAMCI(NC+4)
      HHI=STREAMHI(NC+1)*STREAMHI(NC+4)


C FOR FIRST GUESSES FOR OUTLET TEMPERATURES ASSUME NO PHASE CHANGE
C CALCULATE HEAT CAPACITIES (I.E. DERIVATIVES)
      CALL HEAT(1,PHASEC,NC,TCI,PRC,XC,CP,CPC2)
      CALL HEAT(1,PHASEH,NC,THI,PRH,XH,CP,CPH1)

C ESTIMATE EXIT TEMPERATURES
      ALPHA=(CPH1*STREAMHI(NC+1))/(CPC2*STREAMCI(NC+1))
      BETA=DEXP(UA*(1.0D0/(CPC2*STREAMCI(NC+1))-1.0D0/(CPH1*
     1  STREAMHI(NC+1))))

      TCONEW=(THI*(1-BETA)+TCI*BETA*(1.0D0-ALPHA))/(1.0D0-ALPHA*BETA)

      THONEW=THI+((TCI-THI)*ALPHA*(1.0D0-BETA)/(1.0D0-ALPHA*BETA))

C NOW ITERATE TO CALCULATE EXIT TEMPERATURES ALLOWING FOR POSSIBLE PHASE
```

```
C CHANGE(S)

100     TCO=TCONEW
        THO=THONEW

C FLASH THE STREAMS AT THE ASSUMED EXIT CONDITIONS TO OBTAIN ENTHALPIES
        CALL PTFLASH(NC,STREAMCI,PRC,TCO,STRL,STRV,EPS,IPHC,IMAX,ERR)
        HCO=STRL(NC+1)*STRL(NC+4)+STRV(NC+1)*STRV(NC+4)
        CALL PTFLASH(NC,STREAMHI,PRH,THO,STRL,STRV,EPS,IPHH,IMAX,ERR)
        HHO=STRL(NC+1)*STRL(NC+4)+STRV(NC+1)*STRV(NC+4)

C CALCULATE ENTHALPIES AT T + DELTA T FOR USE IN FINDING DERVIATIVES
        CALL PTFLASH(NC,STREAMCI,PRC,TCO+DELT,STRL,STRV,EPS,IPH,IMAX,
     1 ERR)
        HCODEL=STRL(NC+1)*STRL(NC+4)+STRV(NC+1)*STRV(NC+4)
        CALL PTFLASH(NC,STREAMHI,PRH,THO+DELT,STRL,STRV,EPS,IPH,IMAX,
     1 ERR)
        HHODEL=STRL(NC+1)*STRL(NC+4)+STRV(NC+1)*STRV(NC+4)

C CALCULATE F1,F2, AND DERIVATIVES OF F1,F2 WITH RESPECT TO TCO,THO
        F1=FUNCF1(THI,THO,TCI,TCO,HHI,HHO,HCI,UA)
        F2=FUNCF2(HHI,HHO,HCI,HCO)
        DF1C=FUNCF1(THI,THO,TCI,TCO+DELT,HHI,HHO,HCI,UA)
        DF1H=FUNCF1(THI,THO+DELT,TCI,TCO,HHI,HHODEL,HCI,UA)
        DF2C=FUNCF2(HHI,HHO,HCI,HCODEL)
        DF2H=FUNCF2(HHI,HHODEL,HCI,HCO)
        DF1DTC=(DF1C-F1)/DELT
        DF1DTH=(DF1H-F1)/DELT
        DF2DTC=(DF2C-F2)/DELT
        DF2DTH=(DF2H-F2)/DELT

C UPDATE VALUES FOR TCO, THO
        DENOM=DF1DTC*DF2DTH-DF1DTH*DF2DTC
        TCONEW=TCO+(F2*DF1DTH-F1*DF2DTH)/DENOM
        THONEW=THO+(F1*DF2DTC-F2*DF1DTC)/DENOM

C BOUND TEMPERATURES BY INLET VALUES
```

```
      IF(TCONEW.LT.TCI) THEN
              TCONEW=(TCO+TCI)/2.0D0
      ENDIF
      IF(TCONEW.GT.THI) THEN
              TCONEW=(TCO+THI)/2.0D0
      ENDIF
      IF(THONEW.LT.TCI) THEN
              THONEW=(THO+TCI)/2.0D0
      ENDIF
      IF(THONEW.GT.THI) THEN
              THONEW=(THO+THI)/2.0D0
      ENDIF

C CHECK FOR CONVERGENCE OF OUTLET TEMPERATURES
      IF(DABS((TCONEW-TCO)/TCO).GT.EPS) GOTO 500
      IF(DABS((THONEW-THO)/THO).GT.EPS) GOTO 500

C MUST HAVE CONVERGED, FINISH UP RETURN VALUES
      GOTO 1000

C DID NOT CONVERGE. IF MORE ITERATIONS LEFT UPDATE VARIABLES AGAIN
C ITERATE ON POSSIBLE FLASH CONDITION UNTIL CONVERGED
500   ITER=ITER+1
      IF(ITER.LT.IMAX) GOTO 100

C SHOW NON-CONVERGENCE ERROR
      ERR=1

C SET OUTLET CONDITIONS - T, H, AND PHASE
1000  STREAMCO(NC+2)=TCO
      STREAMCO(NC+4)=HCO/STREAMCO(NC+1)
      STREAMCO(NC+5)=IPHC

      STREAMHO(NC+2)=THO
      STREAMHO(NC+4)=HHO/STREAMHO(NC+1)
      STREAMHO(NC+5)=IPHH
```

```
RETURN
END


REAL*8 FUNCTION FUNCF1(THI,THO,TCI,TCO,HHI,HHO,HCI,UA)
IMPLICIT REAL*8(A-H,O-Z)
DELT1=THI-TCO
DELT2=THO-TCI
TLM=(DELT1-DELT2)/(DLOG(DELT1/DELT2))
FUNCF1=((HHI-HHO)-UA*TLM)/(HHI+HCI)
RETURN
END

REAL*8 FUNCTION FUNCF2(HHI,HHO,HCI,HCO)
IMPLICIT REAL*8(A-H,O-Z)
FUNCF2=(HHI-HHO+HCI-HCO)/(HHI+HCI)
RETURN
END
```

```
      SUBROUTINE REBOIL(NC,STRI,STROL,STROV,PVESS,TRB,QIN,IMAX,EPS,
     1 ERROR)
C REBOILER SUBROUTINE
C
C LAST UPDATE: 04/21/86
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 STRI(25),STROL(25),STROV(25)

      INTEGER*2 ERROR

C SHOW NO ERROR
      ERROR=0

C FLASH THE STREAM AT THE INDICATED TEMPERATURE
      CALL PTFLASH(NC,STRI,PVESS,TRB,STROL,STROV,EPS,IPHASE,IMAX,
     1 ERROR)

C CALCULATE THE HEAT INPUT REQUIRED
      QIN=STROL(NC+1)*STROL(NC+4)+STROV(NC+1)*STROV(NC+4)-
     1 STRI(NC+1)*STRI(NC+4)

      RETURN
      END
```

```
      SUBROUTINE CONDENSE(NCOMP,PVAP,STREAMI,STREAMO,QOUT,EPSILON,
     1 ERROR)
C CONDENSER SUBROUTINE
C
C LAST UPDATE: 04/21/86
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 X(20),Y(20),HL(20)
      REAL*8 STREAMI(25),STREAMO(25)

      INTEGER*2 ERROR

C SHOW NO ERROR
      ERROR=0

C     SET UP NEW LIQUID VALUES
      DO 10 I=1,NCOMP
      Y(I)=STREAMI(I)
10    X(I)=Y(I)

      TVAP=STREAMI(NCOMP+2)

C CALCULATE BUBBLE POINT.  ASSUME THAT THE BUBBLE POINT IS
C THE TEMP IN THE CONDENSER - I.E. ASSUME TOTAL CONDENSER
      CALL BUBDEW(NCOMP,TVAP,X,Y,TBUB,TDEW,PVAP,EPSILON,0)

C CALCULATE NEW LIQUID STREAM ENTHALPY AT THE BUBBLE POINT
      CALL HEAT(0,1,NCOMP,TBUB,PVAP,X,HL,HTL)

C SET ALL OUTLET STREAM VALUES
      DO 20 I=1,NCOMP+1
20    STREAMO(I)=STREAMI(I)
      STREAMO(NCOMP+2)=TBUB
      STREAMO(NCOMP+3)=PVAP
      STREAMO(NCOMP+4)=HTL
      STREAMO(NCOMP+5)=1
```

```
C CALCULATE HEAT REMOVED
      QOUT=STREAMO(NC+1)*STREAMO(NC+4)-STREAMI(NC+1)*STREAMI(NC+4)

      RETURN
      END
```

```
      SUBROUTINE MIXER(NCOMP,STREAM1,STREAM2,STREAMO,PMIX,IPHASE,
     $ EPSILON,ERROR)

C MIXER SUBROUTINE
C
C LAST UPDATE: 04/21/86

      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER*2 ERROR
      REAL*8 STREAM1(25),STREAM2(25),STREAMO(25)
      REAL*8 STRL(25),STRV(25)

      IMAX=20

C CALCULATE EXIT FLOWS FOR MIXTURE
      DO 10 I=1,NCOMP
10    STREAMO(I)=STREAM1(I)*STREAM1(NCOMP+1)+STREAM2(I)*
     1 STREAM2(NCOMP+1)

      STREAMO(NCOMP+1)=STREAM1(NCOMP+1)+STREAM2(NCOMP+1)

C CALCULATE MOLE FRACTIONS FOR MIXTURE
      DO 20 I=1,NCOMP
20    STREAMO(I)=STREAMO(I)/STREAMO(NCOMP+1)

C SET EXIT PRESSURE AND ENTHALPY
      STREAMO(NCOMP+3)=PMIX
      STREAMO(NCOMP+4)=(STREAM1(NCOMP+4)*STREAM1(NCOMP+1)+
     1 STREAM2(NCOMP+4)*STREAM2(NCOMP+1))/STREAMO(NCOMP+1)

C SET GUESS FOR MIXTURE TEMPERATURE FOR FLASH CALC -- USE AVERAGE
      TFEED=((STREAM1(NCOMP+2)*STREAM1(NCOMP+1))+(STREAM2(NCOMP+2)*
     1 STREAM2(NCOMP+1)))/(STREAM1(NCOMP+1)+STREAM2(NCOMP+1))
      STREAMO(NCOMP+2)=TFEED

C DO A FLASH ON THE MIXTURE TO DETERMINE PHASE(S) AND COMPOSITION
      CALL PHFLASH(NCOMP,STREAMO,PMIX,STRL,STRV,EPSILON,
```

```
     1 IPHASE,IMAX,ERROR)

C SET RETURN TEMPERATURE AND PHASE
        STREAMO(NCOMP+2)=STRL(NCOMP+2)
        STREAMO(NCOMP+5)=IPHASE

        RETURN
        END
```

```
      SUBROUTINE SPLIT(NCOMP,STREAMI,STREAM1,STREAM2,FRACT1)

C SPLITTER SUBROUTINE
C
C LAST UPDATE: 04/21/86

      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 STREAMI(25),STREAM1(25),STREAM2(25)

C SET RETURN VALUES FOR ALL ZI,FLOW,TEMP,PRESS,ENTHALPY,PHASE
      DO 10 I=1,NCOMP+5
      STREAM1(I)=STREAMI(I)
10    STREAM2(I)=STREAMI(I)

C CORRECT VALUES OF FLOW FOR SPLIT
      STREAM1(NCOMP+1)=STREAMI(NCOMP+1)*FRACT1
      STREAM2(NCOMP+1)=STREAMI(NCOMP+1)*(1-FRACT1)

      RETURN
      END
```

```
      SUBROUTINE PUMP(NC,STREAMI,STREAMO,POUT,RHOM,POWER,IMAX,EPS,
     1 ERROR)

C LAST UPDATE: 06/10/86

C PUMPING SUBROUTINE. MODIFIES THE PRESSURE OF A STREAM AND RECALCULATES
C THE ENTHALPY. ASSUMES ISOTHERMAL OPERATION.
C ALSO SETS THE EXIT PHASE COMPOSITION AND CALCULATES THE POWER REQUIREMENT.

      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER*2 ERROR
      REAL*8 STREAMI(25),STREAMO(25)
      REAL*8 STRL(25),STRV(25)

C TRANSFER ALL VARIABLES FROM INLET TO OUTLET STREAM
      DO 10 I=1,NC+5
10    STREAMO(I)=STREAMI(I)

C SET NEW VALUE FOR PRESSURE
      STREAMO(NC+3)=POUT

C FLASH THE STREAM TO DETERMINE ENTHALPY AND PHASE
      CALL PTFLASH(NC,STREAMO,POUT,STREAMO(NC+2),STRL,STRV,EPS,
     1  IPHASE,IMAX,ERROR)

      STREAMO(NC+4)=(STRL(NC+1)*STRL(NC+4)+STRV(NC+1)*STRV(NC+4))/
     1  STREAMO(NC+1)

      STREAMO(NC+5)=IPHASE

C CALCULATE POWER REQUIREMENT
      POWER=(POUT-STREAMI(NC+3))*STREAMI(NC+1)/RHOM

      RETURN
      END
```

```
      SUBROUTINE REACTOR(NCOMP,STREAM1,STREAM2,PREACT,QOUT,STREAMO,
     1 FRACT,RK2,VREACT,IPHASE,EPSILON,ERROR)

C REACTOR SUBROUTINE
C
C LAST UPDATE: 04/21/86


      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER*2 ERROR
      INTEGER*2 ALPHA(20)

      REAL*8 STREAM1(25),STREAM2(25),STREAMO(25)
      REAL*8 STRL(25),STRV(25),HFI(20),DHFIDT(20)
      REAL*8 MOLWT(20)

      COMMON/ALPH/ALPHA
      COMMON/MOLEC/MOLWT

C SHOW NO ERROR YET
      ERROR=0

C PERFORM MASS BALANCE
C MIX FIRST, CALCULATE FLOWS BEFORE REACTION

      DO 10 I=1,NCOMP
10    STREAMO(I)=STREAM1(I)*STREAM1(NCOMP+1)+
     1 STREAM2(I)*STREAM2(NCOMP+1)

C DETERMINE LIMITING REACTANT -- H2S OR SO2
      FLOWH2S=STREAMO(3)
      FLOWSO2=STREAMO(4)

      IF(FLOWH2S.GT.2.0D0*FLOWSO2) THEN
C     SO2 IS LIMITING
            FLOWCHG=FRACT*FLOWSO2
      ELSE
```

```
C       H2S IS LIMITING
                FLOWCHG=FRACT*FLOWH2S/2.0D0
        ENDIF

C CALCULATE COMPOSITION AFTER REACTION AND TOTAL OUTFLOW
        STREAMO(NCOMP+1)=0.0D0
        DO 20 I=1,NCOMP
        STREAMO(I)=STREAMO(I)+ALPHA(I)*FLOWCHG
20      STREAMO(NCOMP+1)=STREAMO(NCOMP+1)+STREAMO(I)

C CONVERT FLOWS TO MOLE FRACTIONS
        DO 25 I=1,NCOMP
25      STREAMO(I)=STREAMO(I)/STREAMO(NCOMP+1)


C SET EXIT PRESSURE AND ENTHALPY ACCOUNTING FOR REACTION AND HEAT REMOVAL
        STREAMO(NCOMP+3)=PREACT
        STREAMO(NCOMP+4)=(STREAM1(NCOMP+4)*STREAM1(NCOMP+1)
     1 +STREAM2(NCOMP+4)*STREAM2(NCOMP+1)-QOUT)/
     2 STREAMO(NCOMP+1)


C SET GUESS FOR MIXTURE TEMPERATURE FOR FLASH CALC -- USE AVERAGE
        TFEED=((STREAM1(NCOMP+2)*STREAM1(NCOMP+1))+(STREAM2(NCOMP+2)*
     1 STREAM2(NCOMP+1)))/(STREAM1(NCOMP+1)+STREAM2(NCOMP+1))
        STREAMO(NCOMP+2)=TFEED

        TFLASH=TFEED

        STREAMO(NCOMP+2)=TFEED

C SAVE THE TEMP, ENTHALPY VALUE AND INLET PHASE TYPE
        SAVET=STREAMO(NCOMP+2)
        SAVENTH=STREAMO(NCOMP+4)
        SAVEPH=STREAMO(NCOMP+5)

C ITERATE ON SULFUR SOLUBILITY AND FLASH TO DETERMINE EXIT CONDITIONS
```

```
      IMAX=20
      ITER=0
30    ITER=ITER+1

      IF(ITER.GT.IMAX) THEN
              ERROR=1
              GOTO 40
      ENDIF

      TSOL=TFLASH
      CALL SULFSOL(NCOMP,STREAMO,STREAMO,TSOL,PREACT,EPSILON,ERROR)

C RECOVER CORRECT TEMP, ENTHALPY (SULFSOL RECALCS ENTHALPY FOR STREAM) AND PHASE
      STREAMO(NCOMP+2)=SAVET
      STREAMO(NCOMP+4)=SAVENTH
      STREAMO(NCOMP+5)=SAVEPH

C DO A FLASH ON THE MIXTURE TO DETERMINE PHASE(S) AND COMPOSITION
      CALL PHFLASH(NCOMP,STREAMO,PREACT,STRL,STRV,EPSILON,
     1 IPHASE,30,ERROR)

      TFLASH=STRL(NCOMP+2)

C CHECK FOR TEMPERATURE CONVERGENCE
      IF(DABS(TFLASH-TSOL).GT.EPSILON) GOTO 30

C SET OUTLET VALUES FOR PHASE TYPE AND EXIT TEMPERATURE
C CHECK FOR PRESENCE OF SOLID SULFUR (COMPONENT 17)
40    IF((STREAMO(17).NE.0.0D0).AND.(IPHASE.LE.3)) THEN
       STREAMO(NCOMP+5)=IPHASE+3
      ELSE
       STREAMO(NCOMP+5)=IPHASE
      ENDIF

      STREAMO(NCOMP+2)=TFLASH

      IF(IPHASE.EQ.3) THEN
```

```
            ERROR=2
      ELSE
            ERROR=0
      ENDIF

C CALCULATE REACTOR VOLUME FOR PLUG FLOW OPERATION

C INLET CONCENTRATIONS OF SO2 AND H2S
      DENOM=(STREAM1(NCOMP+1)+STREAM2(NCOMP+1))*MOLWT(1)
      CSO2=1000.D0*FLOWSO2/DENOM
      CH2S=1000.D0*FLOWH2S/DENOM

C FLOW IN GAL PER SEC
      FLOW=STREAMO(NCOMP+1)*MOLWT(1)/3.79

      IF(CH2S.EQ.2.D0*CSO2) THEN
            C1=1.D0/(1.D0-FRACT)-1.D0
            VREACT=FLOW*CONVER/(2.D0*RK2*CSO2)
      ELSE
        IF(CH2S.GT.2.D0*CSO2) THEN
            C1=DLOG((CH2S-2.D0*CSO2*FRACT)/(1.D0-FRACT))
            C2=DLOG(CH2S)
            VREACT=FLOW*(1.D0/RK2)*(1.D0/(CH2S-2.D0*CSO2))*(C1-C2)
        ELSE
            C1=DLOG((CSO2-0.5D0*CH2S*FRACT)/(1.D0-FRACT))
            C2=DLOG(CSO2)
            VREACT=FLOW*(1.D0/RK2)*(1.D0/(CSO2-0.5D0*CH2S))*(C1-C2)
        ENDIF

      ENDIF

      RETURN
      END
```

```
      SUBROUTINE SULFSOL(NCOMP,STREAMI,STREAMO,TSOL,PSOL,EPSILON,
     1 ERROR)

C SULFUR SOLUBILITY ROUTINE
C
C LAST UPDATE: 04/21/86

      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER*2 ERROR

      REAL*8 STREAMI(25),STREAMO(25),X(20),HU(20)

      EXTERNAL SULFUR

      ERROR=0

C TRANSFER ALL INPUT VALUES TO OUTPUT
      DO 10 I=1,NCOMP+5
10    STREAMO(I)=STREAMI(I)

C CALCULATE SULFUR SOLUBILITY
      XS=SULFUR(TSOL)

C SEE IF SOLUBILITY HAS BEEN EXCEEDED. SKIP IF NOT
      IF(XS.GT.STREAMI(5)) GOTO 100

C CALCULATE FLOWS OF DISSOLVED AND SOLID SULFUR

C TOTAL SULFUR FREE FLOW
      FLOWNS=STREAMO(NCOMP+1)*(1.0D0-STREAMO(5)-STREAMO(17))

C TOTAL SULFUR FLOW
      SULF=(STREAMO(5)+STREAMO(17))*STREAMO(NCOMP+1)

C CALCULATE DISSOLVED SULFUR FLOW=XS*FLOWNS/(1.D0-XS)
      FLOWSU=XS*FLOWNS/(1.D0-XS)
```

```
C SET OUTLET FRACTIONS OF DISSOLVED SULFUR AND SOLID SULFUR
      STREAMO(5)=FLOWSU/STREAMO(NCOMP+1)
      STREAMO(17)=(SULF-FLOWSU)/STREAMO(NCOMP+1)

C SHOW STREAM CONTAINS MIX OF LIQUID AND SOLID
      IF(STREAMI(NCOMP+5).LT.4.D0) THEN
              STREAMO(NCOMP+5)=STREAMI(NCOMP+5)+3
      ELSE
              STREAMO(NCOMP+5)=STREAMI(NCOMP+5)
      ENDIF


C SET OUTLET STREAM TEMPERATURE AND PRESSURE
100   STREAMO(NCOMP+2)=TSOL
      STREAMO(NCOMP+3)=PSOL

C CALCULATE OUTLET STREAM ENTHALPY
      DO 150 I=1,NCOMP
150   X(I)=STREAMO(I)

      CALL HEAT(0,1,NCOMP,TSOL,PSOL,X,MU,HT)

      STREAMO(NCOMP+4)=HT

      RETURN
      END
```

```
      SUBROUTINE SETTLER(NC,STRI,STRS,STRO,WTS,PS,EPSILON,ERROR)

C SULFUR SETTLER SUBROUTINE
C
C LAST UPDATE: 04/21/86

      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER*2 ERROR
      REAL*8 STRI(25),STRO(25),STRS(25),XO(20),XS(20),HX(20)
      REAL*8 STRIT(25)
      REAL*8 MOLWT(20)

      COMMON /MOLEC/MOLWT


C SHOW NO ERROR YET
      ERROR=0

C MAKE CALL TO SULFUR CRYSTALLIZER ROUTINE TO CHECK SULFUR SATURATION
      DO 3 I=1,NC+5
3     STRIT(I)=STRI(I)

      CALL CRYSTAL(NC,STRIT,STRI,EPSILON,ERROR)


C INIT TEMPORARY MOLE FRACTION VALUES
      DO 5 I=1,NC
      XO(I)=0.000
5     XS(I)=0.000

C CONVERT FROM WT % TO WT FRACTION
      WTS=WTS/100.000

C CALCULATE THE TOTAL MASS FLOW OF THE INLET STREAM
      WTI=0.000
      DO 10 I=1,NC
10    WTI=WTI+STRI(I)*STRI(NC+1)*MOLWT(I)
```

```
C CALCULATE MASS FLOW OF SOLID SULFUR AND REMOVE IT FROM THE OVERALL FLOW
        WS=STRI(17)*STRI(NC+1)*MOLWT(17)
        WTI=WTI-WS

C CALCULATE FRACTION OF SOLVENT STREAM TO TRANSFER WITH THE SULFUR
C AND FRACTION TO REMOVE
        FT1=(WS*(1.D0-WTS)/(WTS*WTI))
        FT2=1.0D0-FT1

C MAKE SURE THAT SUFFICIENT SOLVENT EXISTS TO YIELD THE DESTIRED WT FRACTION
C OF SULFUR -- IF NOT ENOUGH THEN TRANSFER ALL OF THE SOLVENT STREAM TO
C THE EXIT SULFUR STREAM
        IF(FT1.GT.1.0D0) THEN
                FT1=1.0D0
                FT2=0.0D0
        ENDIF

C SET FLOWS FOR BOTH EXIT STREAMS
        STRO(NC+1)=0.0D0
        STRS(NC+1)=0.0D0

        DO 20 I=1,NC
        STRS(I)=FT1*STRI(I)*STRI(NC+1)
        STRS(NC+1)=STRS(NC+1)+STRS(I)
        STRO(I)=FT2*STRI(I)*STRI(NC+1)
20      STRO(NC+1)=STRO(NC+1)+STRO(I)

C MOVE ALL OF THE SOLID SULFUR TO THE SULFUR STREAM
        STRS(17)=STRI(17)*STRI(NC+1)
        STRS(NC+1)=STRS(NC+1)+STRO(17)
        STRO(NC+1)=STRO(NC+1)-STRO(17)
        STRO(17)=0.0D0

C CALCULATE EXIT MOLE FRACTIONS
        IF(STRS(NC+1).EQ.0.0D0) GOTO 30
        DO 25 I=1,NC
```

```
      STRS(I)=STRS(I)/STRS(NC+1)
25    XS(I)=STRS(I)

30    IF(STRO(NC+1).EQ.0.0D0) GOTO 40
      DO 35 I=1,NC
      STRO(I)=STRO(I)/STRO(NC+1)
35    XO(I)=STRO(I)

C SET EXIT TEMP, PRESS, AND PHASE
40    STRO(NC+2)=STRI(NC+2)
      STRS(NC+2)=STRI(NC+2)
      STRO(NC+3)=PS
      STRS(NC+3)=PS
      STRO(NC+5)=STRI(NC+5)
      IF(STRI(NC+5).LE.3) THEN
              STRS(NC+5)=STRI(NC+5)+3
      ELSE
              STRS(NC+5)=STRI(NC+5)
      ENDIF

C CALCULATE EXIT STREAM ENTHALPY
      CALL HEAT(0,1,NC,STRO(NC+2),PS,XO,HX,STRO(NC+4))
      CALL HEAT(0,1,NC,STRS(NC+2),PS,XS,HX,STRS(NC+4))

      RETURN
      END
```

```
      SUBROUTINE WASHER(NC,STRI,STRW,STRO,STRS,PS,WTS,EPSILON,ERROR)

C SULFUR WASHER
C
C LAST UPDATE: 04/21/86

      IMPLICIT REAL*8(A-H,O-Z)
      LOGICAL ERROR

      REAL*8 STRI(25),STRW(25),STRO(25),STRS(25),STRIM(25)

      ERROR=.FALSE.

C THE WASHER IS TREATED AS A COMBINATION OF A MIXER (INLET +WATER) AND
C A SETTLER (SULFUR/WATER IS REMOVED FROM THE MAIN LIQUID STREAM)

C CALL MIXER ROUTINE TO BLEND INLET SULFUR SLURRY WITH WATER WASH
      CALL MIXER(NC,STRI,STRW,STRIM,PS,IPHASE,EPSILON,ERROR)

C SEND OUTLET OF MIXER TO SETTLER TO REMOVE "WASHED SULFUR"
      CALL SETTLER(NC,STRIM,STRO,STRS,WTS,PS,EPSILON,ERROR)

      RETURN
      END
```

```
      SUBROUTINE MELTDEC(NC,STRI,TMELT,PMELT,STRW,STRS,QIN,EPS,ERROR)

C MELTER/DECANTER SUBROUTINE
C
C LAST UPDATE: 04/21/86

      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER*2 ERROR
      REAL*8 STRI(25),STRW(25),STRS(25),HX(20)
      REAL*8 X(20)
      REAL*8 STL(25),STV(25)

C INITIALIZE OUTPUT STREAM MOLE FRACTIONS TO ZERO
      DO 10 I=1,NC
      STRW(I)=0.0D0
10    STRS(I)=0.0D0

C SET OUTLET FLOWS AND MOLE FRACTIONS
C ASSUME OUTLET SULFUR STREAM IS PURE SULFUR LIQUID
      STRS(NC+1)=STRI(17)*STRI(NC+1)
      STRS(17)=1.0D0

C SET EXIT WATER STREAM
C CALCULATE COMPONENT FLOWS AND TOTAL FLOW
      STRW(NC+1)=0.D0
      DO 20 I=1,NC
      STRW(I)=STRI(I)*STRI(NC+1)
20    STRW(NC+1)=STRW(NC+1)+STRW(I)

      STRW(NC+1)=STRW(NC+1)-STRI(17)*STRI(NC+1)
      STRW(17)=0.D0

C CALCULATE EXIT MOLE FRACTIONS IN WATER STREAM
      IF(STRW(NC+1).EQ.0.D0) GOTO 35
      DO 30 I=1,NC
30    STRW(I)=STRW(I)/STRW(NC+1)
```

```
C SET OUTLET TEMP,PRESS, AND PHASE
35      STRW(NC+2)=TMELT
        STRW(NC+3)=PMELT
        STRW(NC+5)=1

        STRS(NC+2)=TMELT
        STRS(NC+3)=PMELT
        STRS(NC+5)=1

C PERFORM ISOTHERMAL FLASH ON WATER STREAM
        IMAX=20
        CALL PTFLASH(NC,STRW,PMELT,TMELT,STL,STV,EPS,IPHASE,IMAX,ERROR)

        STRW(NC+4)=(STL(NC+4)*STL(NC+1)+STV(NC+4)*STV(NC+1))/
     1  STRW(NC+1)

        IF(IPHASE.EQ.2) THEN
         STRW(NC+5)=2
        ENDIF

        IF(IPHASE.EQ.3) THEN
         STRW(NC+5)=3
        ENDIF

C CALCULATE ENTHALPY OF EXIT SULFUR STREAM
        DO 40 I=1,NC
40      X(I)=STRS(I)
        CALL HEAT(0,1,NC,TMELT,PMELT,X,HX,STRS(NC+4))

C CALCULATE INPUT HEAT REQUIRED
        QIN=STRW(NC+4)*STRW(NC+1)+STRS(NC+4)*STRS(NC+1)-
     1 STRI(NC+4)*STRI(NC+1)

        RETURN
        END
```

```
      SUBROUTINE FURNACE(NC,SO2REQ,STSULF,ST2,STAIR,STOUT,TOUT,POUT,
     1 QOUT,EPSILON, ERROR)

C FURNACE SUBROUTINE
C
C LAST UPDATE: 04/21/86

      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER*2 ERROR
      INTEGER*2 KEY
      REAL*8 STSULF(25),ST2(25),STAIR(25),STOUT(25),HX(20),X(20)
      REAL*8 CMAT(20,5)
C MATRIX OF COEFFICIENTS FOR COMBUSTION CALCULATIONS
      DATA (CMAT(1,I),I=1,5)/0.D0,0.D0,0.D0,0.D0,0.D0/
      DATA (CMAT(2,I),I=1,5)/0.D0,0.D0,0.D0,0.D0,1.D0/
      DATA (CMAT(3,I),I=1,5)/1.5D0,1.D0,1.D0,0.D0,0.D0/
      DATA (CMAT(4,I),I=1,5)/0.D0,0.D0,0.D0,0.D0,1.D0/
      DATA (CMAT(5,I),I=1,5)/0.D0,0.D0,0.D0,0.D0,0.D0/
      DATA (CMAT(6,I),I=1,5)/0.D0,0.D0,0.D0,0.D0,1.D0/
      DATA (CMAT(7,I),I=1,5)/0.D0,0.D0,0.D0,0.D0,1.D0/
      DATA (CMAT(8,I),I=1,5)/2.D0,0.D0,2.D0,1.D0,0.D0/
      DATA (CMAT(9,I),I=1,5)/0.5D0,0.D0,1.D0,0.D0,0.D0/
      DATA (CMAT(10,I),I=1,5)/0.5D0,0.D0,0.D0,1.D0,0.D0/
      DATA (CMAT(11,I),I=1,5)/3.5D0,0.D0,3.D0,2.D0,0.D0/
      DATA (CMAT(12,I),I=1,5)/5.D0,0.D0,4.D0,3.D0,0.D0/
      DATA (CMAT(13,I),I=1,5)/6.5D0,0.D0,5.D0,4.D0,0.D0/
      DATA (CMAT(14,I),I=1,5)/8.D0,0.D0,6.D0,5.D0,0.D0/
      DATA (CMAT(15,I),I=1,5)/3.D0,1.D0,2.D0,1.D0,0.D0/
      DATA (CMAT(16,I),I=1,5)/4.5D0,1.D0,3.D0,2.D0,0.D0/
      DATA (CMAT(17,I),I=1,5)/1.D0,1.D0,0.D0,0.D0,0.D0/
      DATA (CMAT(18,I),I=1,5)/0.D0,0.D0,0.D0,0.D0,1.D0/
      DATA (CMAT(19,I),I=1,5)/0.D0,0.D0,0.D0,0.D0,0.D0/
      DATA (CMAT(20,I),I=1,5)/0.D0,0.D0,0.D0,0.D0,0.D0/


C NOTE: ON INPUT THE VALUES FOR COMPOSITION, T, P, AND PHASE FOR
C THE AIR STREAM ARE SET. THE FLOW IS CALCULATED BY THIS ROUTINE.
```

```
C IT IS ASSUMED THAT THERE IS NOTHING IN THE AIR WHICH CAN BE
C BURNED

C WE ASSUME COMPLETE COMBUSTTION OF ALL COMPONENTS WHICH BURN.
C A STOICHIOMETRIC AMOUNT OF OXYGEN IS USED.

C THE INLET STREAMS ARE:
C       STSULF = MAIN SULFUR CONTAINING STREAM FOR SO2 PRODUCTION
C       ST2    = "JUNK" STREAM CONTAINING ANYTHING WHICH WE WOULD
C                  LIKE TO BURN (INCLUDING SULFUR CPNDS)

C ON INPUT THE TEMPERATURE, PRESSURE AND PHASE OF THE SULFUR STREAM
C SHOULD BE SET. THE FLOW OF STSULF IS ADJUSTED TO YIELD THE CORRECT
C AMOUNT OF SO2 IN THE OUTLET (SO2REQ) TAKING THE SO2 FORMED FROM THE
C "JUNK" STREAM INTO ACCOUNT

        ERROR=0

C CALCULATE THE SO2 AVAILABLE FROM THE "JUNK" STREAM
        SO2FORM=ST2(4)*ST2(NC+1)
        DO 5 I=1,NC
5       SO2FORM=SO2FORM+ST2(I)*ST2(NC+1)*CMAT(I,2)

C ADJUST FLOW OF MAIN SULFUR STREAM TO ACCOUNT FOR SO2 PRODUCED
C BY BURNING THE "JUNK" STREAM
        DELSO2=SO2REQ-SO2FORM
        SULFSUM=STSULF(4)
        DO 7 I=1,NC
7       SULFSUM=SULFSUM+STSULF(I)*CMAT(I,2)
        STSULF(NC+1)=DELSO2/SULFSUM

C ZERO THE OUTPUT STREAM
        DO 10 I=1,NC+5
10      STOUT(I)=0.000

        O2REQ=0.000
C DETERMINE OXYGEN REQUIREMENT
```

```
      DO 20 I=1,NC
20    O2REQ=O2REQ+(STSULF(I)*STSULF(NC+1)+ST2(I)*ST2(NC+1))*CMAT(I,1)

C CALCULATE INLET AIR REQUIREMENT
      STAIR(NC+1)=O2REQ/STAIR(19)

C TRANSFER ALL INERTS TO THE OUTLET STREAM
      DO 30 I=1,NC
30    STOUT(I)=(STSULF(I)*STSULF(NC+1)+ST2(I)*ST2(NC+1)+
     1 STAIR(I)*STAIR(NC+1))*CMAT(I,5)

C DETERMINE AMOUNT OF SO2, CO2, AND H2O FORMED
      DO 40 I=1,NC
      STOUT(4)=STOUT(4)+(STSULF(I)*STSULF(NC+1)+ST2(I)*ST2(NC+1))
     1 *CMAT(I,2)
      STOUT(2)=STOUT(2)+(STSULF(I)*STSULF(NC+1)+ST2(I)*ST2(NC+1))
     1 *CMAT(I,3)
40    STOUT(6)=STOUT(6)+(STSULF(I)*STSULF(NC+1)+ST2(I)*ST2(NC+1))
     1 *CMAT(I,4)

C CALC TOTAL OUTLET FLOW AND CONVERT COMPONENT FLOWS TO MOLE FRACTIONS
      DO 50 I=1,NC
50    STOUT(NC+1)=STOUT(NC+1)+STOUT(I)

      DO 60 I=1,NC
60    STOUT(I)=STOUT(I)/STOUT(NC+1)

C CALCULATE AIR ENTHALPY
      DO 70 I=1,NC
70    X(I)=STAIR(I)
      CALL HEAT(0,2,NC,STAIR(NC+2),STAIR(NC+3),X,HX,STAIR(NC+4))

C CALCULATE SULFUR STREAM ENTHALPY
      DO 75 I=1,NC
75    X(I)=STSULF(I)
      CALL HEAT(0,1,NC,STSULF(NC+2),STSULF(NC+3),X,HX,STSULF(NC+4))
```

```
C SET EXIT GAS T,P,PHASE
      STOUT(NC+2)=TOUT
      STOUT(NC+3)=POUT
      STOUT(NC+5)=2.0D0

C CALC EXIT GAS ENTHALPY
      DO 80 I=1,NC
80    X(I)=STOUT(I)
      CALL HEAT(0,2,NC,STOUT(NC+2),STOUT(NC+3),X,HX,STOUT(NC+4))

C CALCULATE QOUT
      QOUT=STAIR(NC+1)*STAIR(NC+4)+STSULF(NC+1)*STSULF(NC+4)+
     1 ST2(NC+1)*ST2(NC+4)-STOUT(NC+1)*STOUT(NC+4)

      RETURN
      END
```

```
      SUBROUTINE COMPRESS(NC,STREAMI,STREAMO,POUT,POWER,IMAX,EPS,
     1  ERROR)

C LAST UPDATE: 06/10/86

C COMPRESSOR SUBROUTINE. USES PRESSURE CHANGE TO CALCULATE THE OUTLET TEMP.
C PHASE, AND ENTHALPY. ASSUMES ADIABATIC OPERATION AND IDEAL GASSES.
C ALSO RETURNS THE POWER REQUIRED.

      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER*2 ERROR
      REAL*8 MOLWT(20)
      REAL*8 STREAMI(25),STREAMO(25)
      REAL*8 STRL(25),STRV(25)

      COMMON/MOLEC/MOLWT

C TRANSFER ALL VARIABLES FROM INLET TO OUTLET STREAM
      DO 10 I=1,NC+5
10    STREAMO(I)=STREAMI(I)

C SET NEW VALUE FOR PRESSURE
      STREAMO(NC+3)=POUT

C CALCULATE THE EXIT TEMPERATURE
      STREAMO(NC+2)=STREAMI(NC+2)*((POUT/STREAMI(NC+3))**.028)


C FLASH THE STREAM TO DETERMINE ENTHALPY AND PHASE
      CALL PTFLASH(NC,STREAMO,POUT,STREAMO(NC+2),STRL,STRV,EPS,
     1  IPHASE,IMAX,ERROR)

      STREAMO(NC+4)=(STRL(NC+1)*STRL(NC+4)+STRV(NC+1)*STRL(NC+4))/
     1  STREAMO(NC+1)

      STREAMO(NC+5)=IPHASE
```

```
C CALCULATE THE POWER REQUIREMENT
      PRATIO=POUT/STREAMI(NC+3)
      POCONST=29.099D0*STREAMI(NC+2)*STREAMI(NC+1)

      IF(PRATIO.LT.5.D0) THEN
C SINGLE STAGE COMPRESSION
            POWER=POCONST*(PRATIO**0.28D0-1.D0)

      ELSE
C HIGH COMPRESSION, USE 2 STAGE COMPRESSOR
            POWER=POCONST*2.D0*(PRATIO**0.14D0-1.D0)

      ENDIF

      IF(STREAMO(NC+5).EQ.3) THEN
            ERROR=2
      ELSE
            ERROR=0
      ENDIF

      RETURN
      END
```

```
      SUBROUTINE SULCENT(NC,P,STRI1,STRI2,STRO1,STRO2,WTS1,WTS2,
     1 EPSILON,ERROR)

C SULFUR CENTRIFUGE ROUTINE
C
C LAST UPDATE: 04/21/86
C
C THE SULFUR CENTRIFUGE SIMULATES THE ACTION OF A PUSHER TYPE
C  CENTRIFUGE.
C THE ROUTINE CONSISTS OF FOUR OPERATIONS:
C  1. SETTLE INLET SULFUR SLURRY TO FORM INDICATED WT% SLURRY
C  2. WASH SLURRY (USUALLY WITH WATER)
C  3. SETTLE SLURRY TO NEW WT% SULFUR IN EXIT
C  4. MIX LIQUIDS OBTAINED IN STEPS 1 AND 3 TO FORM EXIT LIQUID
C

      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER*2 ERROR
      REAL*8 STRI1(25),STRI2(25)
      REAL*8 STRO1(25),STRO2(25)

C STORAGE FOR INTERMEDIATE STREAMS
      REAL*8 STRINT1(25),STRINT2(25),STRINT3(25),STRINT4(25)

C DO FIRST SPLIT (CENTRIFUGATION) OF SULFUR FROM INITIAL ENTRANCE STREAM
      CALL SETTLER(NC,STRI1,STRINT3,STRINT1,WTS1,P,EPSILON,ERROR)

C NOW MIX IN WATER WASH
      CALL MIXER(NC,STRINT3,STRI2,STRINT4,P,0,EPSILON,ERROR)

C PERFORM SECOND SPLIT
      CALL SETTLER(NC,STRINT4,STRO1,STRINT2,WTS2,P,EPSILON,ERROR)

C DO FINAL MIX TO FORM EXIT LIQUID STREAM
      CALL MIXER(NC,STRINT1,STRINT2,STRO2,P,0,EPSILON,ERROR)

      RETURN
```

END

```
      SUBROUTINE PARCOND(NC,STRF,PCOND,TCOND,STRL,STRV,QCHG,EPS,
     1 IMAX,ERROR)

C PARTIAL CONDENSER OR ISOTHERMAL FLASH
C
C LAST UPDATE: 04/21/86

      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER*2 ERROR
      REAL*8 STRF(25),STRL(25),STRV(25)

C TREAT PARTIAL CONDENSER AS AN ISOTHERMAL FLASH CALCULATION
      CALL PTFLASH(NC,STRF,PCOND,TCOND,STRL,STRV,EPS,IPHASE,
     1 IMAX,ERROR)


C CALCULATE ENTHALPY REMOVAL
      QCHG=STRL(NC+1)*STRL(NC+4)+STRV(NC+1)*STRV(NC+4)-
     1 STRF(NC+1)*STRF(NC+4)

      RETURN
      END
```

```
      SUBROUTINE HEATER(NC,STRI,STRO,QIN,EPSILON,ERROR)

C HEATER
C
C LAST UPDATE: 04/21/86

      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 STRI(25),STRO(25),QIN,EPSILON,STRL(25),STRV(25)
      REAL*8 HX(20),DHXDT(20),X(20)
      INTEGER*2 ERROR

C SET MAXIMUM ITERATIONS FOR TEMPERATURE CONVERGENCE
      IMAX=20

      ITER=0

C ADD THE HEAT GIVEN BY QIN TO THE STREAM INLET
C FLASH THE STREAM TO SEE IF A PHASE CHANGE OCCURS

C TRANSFER INLET STREAM TO OUTLET STREAM AND SAVE MOLE FRACTIONS
      DO 10 I=1,NC+5
10    STRO(I)=STRI(I)

      DO 15 I=1,NC
15    X(I)=STRO(I)

C CALCULATE UPDATED ENTHALPY
      STRO(NC+4)=(STRI(NC+4)*STRI(NC+1)+QIN)/STRI(NC+1)


C FLASH THE STREAM TO CHECK FOR A PHASE CHANGE, DETERMINE OUTLET TEMP
50    CALL PHFLASH(NC,STRO,STRO(NC+3),STRL,STRV,EPSILON,IPHASE,IMAX
     1 ,ERROR)

C SET RETURN PHASE
      STRO(NC+5)=IPHASE
```

```
C SET RETURN TEMPERATURE
      STRO(NC+2)=STRL(NC+2)

      RETURN
      END
```

```
      SUBROUTINE STORAGE(NC,STREAMI,STREAMO,FLOWIO,ICOMP,TOUT)

C STORAGE OR COMPONENT FLOW SET ROUTINE
C
C LAST UPDATE: 04/21/86

      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 STREAMI(25),STREAMO(25),X(20),HX(20)


C SET RETURN VALUES FOR ALL ZI,FLOW,TEMP,PRESS,ENTHALPY,PHASE
      DO 10 I=1,NC+5
10    STREAMO(I)=STREAMI(I)

C MODIFY FLOW BASED ON DESIRED FLOW OF COMPONENT I
C LEAVE COMPOSITION UNCHANGED

      STREAMO(NC+1)=FLOWIO/STREAMI(ICOMP)

C MODIFY TEMPERATURE TO THAT SPECIFIED
      STREAMO(NC+2)=TOUT

C RECALC ENTHALPY
      DO 20 I=1,NC
20    X(I)=STREAMO(I)

      IPHASE=INT(STREAMO(NC+5))
      PRESS=STREAMO(NC+3)

      CALL HEAT(0,IPHASE,NC,TOUT,PRESS,X,HX,STREAMO(NC+4))

      RETURN
      END
```

```
      SUBROUTINE CRYSTAL(NC,STRI,STRO,EPSILON,ERROR)

C SULFUR CRYSTALLIZER
C
C LAST UPDATE: 04/21/86

C CALCULATES AND RETURNS THE AMOUNT OF SULFUR WHICH IS TRANSFERED TO
C THE SOLID PHASE BASED ON THE INLET CONDITIONS OF STRI
C USES THE SULFSOL ROUTINE

      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER*2 ERROR

      REAL*8 STRI(25),STRO(25)


C SET TEMPERATURE AND PRESSURE FOR SULFUR SATURATOR
      TSOL=STRI(NC+2)
      PSOL=STRI(NC+3)

C CALL THE SULFUR SOLUBILITY ROUTINE
      CALL SULFSOL(NC,STRI,STRO,TSOL,PSOL,EPSILON,ERROR)

      RETURN
      END
```

```
      SUBROUTINE MAKEUP(NC,STREAMI,STREAMO,FLOWIO,ICOMP,IMAX,EPS,ERR)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 STREAMI(25),STREAMO(25),X(20),HX(20)
      REAL*8 STRL(25),STRV(25)

      INTEGER*2 ERR

C LAST UPDATE: 06/10/86

C SET FLOW OF COMPONENT ICOMP LEAVING ALL OTHER FLOWS UNCHANGED

C CALCULATE ALL FLOWS
      DO 10 I=1,NC
10    STREAMO(I)=STREAMI(I)*STREAMI(NC+1)

C SET FLOW OF ICOMP, CALCULATE NEW MOLE FRACTIONS
      STREAMO(ICOMP)=FLOWIO
      STREAMO(NC+1)=0.0D0

      DO 20 I=1,NC
20    STREAMO(NC+1)=STREAMO(NC+1)+STREAMO(I)

      DO 30 I=1,NC
30    STREAMO(I)=STREAMO(I)/STREAMO(NC+1)

C SET TEMPERATURE, AND PRESSURE
      STREAMO(NC+2)=STREAMI(NC+2)
      STREAMO(NC+3)=STREAMI(NC+3)

C SET ENTHALPY AND PHASE BY FLASH CALCULATION
      CALL PTFLASH(NC,STREAMO,STREAMO(NC+3),STREAMO(NC+2),STRL,STRV,
     1  EPS,IPHASE,IMAX,ERR)

C SET ENTHALPY
      STREAMO(NC+4)=(STRL(NC+1)*STRL(NC+4)+STRV(NC+1)*STRV(NC+4))/
     1  STREAMO(NC+1)
```

```
C SET PHASE
      STREAMO(NC+5)=IPHASE

      RETURN
      END
```

```
      SUBROUTINE TSET(NC,STRI,STRO,T,QCHG,IMAX,EPS,ERR)

C TEMPERATURE SETTER ROUTINE
C
C LAST UPDATE: 04/21/86

C CHANGE THE TEMPERATURE OF THE STREAM TO T
C RECALC ENTHALPY AND RETURN HEAT REMOVED OR ADDED
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 STRI(25),STRO(25)
      REAL*8 STRL(25),STRV(25)
      INTEGER*2 ERR

C TRANSFER MOLE FRACTIONS TO OUTLET AND TEMPORARY VECTOR
      DO 10 I=1,NC
10    STRO(I)=STRI(I)

C SET EXIT FLOW, TEMPERATURE, PRESSURE, PHASE
      STRO(NC+1)=STRI(NC+1)
      STRO(NC+2)=T
      STRO(NC+3)=STRI(NC+3)
      STRO(NC+5)=STRI(NC+5)

C FLASH THE EXIT STREAM

      CALL PTFLASH(NC,STRO,STRO(NC+3),T,STRL,STRV,EPS,IPH,IMAX,ERR)

C SET OUTLET STREAM ENTHALPY AND PHASE

      STRO(NC+4)=(STRL(NC+1)*STRL(NC+4)+STRV(NC+1)*STRV(NC+4))/
     1 STRO(NC+1)
      STRO(NC+5)=IPH


C CALCULATE ENTHALPY CHANGE
      QCHG=(STRO(NC+4)-STRI(NC+4))*STRI(NC+1)
```

```
RETURN
END
```

```
      SUBROUTINE HCSPLIT(NC,STRI,STRO1,STRO2)

C HYDROCARBON REMOVAL ROUTINE
C
C LAST UPDATE: 04/21/86

C ROUTINE TO SPLIT STREAMS LEAVING ALL OF THE HYDROCARBONS
C IN EXIT STREAM 1.
C THIS ROUTINE IS NOT THERMODYNAMICALLY VALID!
C IT SHOULD ONLY BE USED IN CASES WHERE A NEARLY COMPLETE
C  SPLIT BETWEEN HYDROCARBONS AND OTHER COMPONENTS CAN BE
C  OBTAINED. AN EXAMPLE OF THIS WOULD BE A PHASE SPLIT BETWEEN
C  WATER AND HYDROCARBONS. UNDER THESE CONDITIONS THE ROUTINE PERFORMS
C  A USEFUL ALTHOUGH NOT STRICTLY "KOSHER" FUNCTION.
C
C IF THE SIMULATION IS EVER UPDATED TO SIMULTANEOUSLY HANDLE TWO
C  SEPARATE LIQUID PHASES, THIS ROUTINE CAN BE CHANGED TO AN
C  EXTRACTION SUBROUTINE

      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 STRI(25),STRO1(25),STRO2(25),X1(20),X2(20),HX(20)

C INITIALIZE OUTLET STREAMS
      DO 10 I=1,25
      STRO1(I)=0.0D0
10    STRO2(I)=0.0D0

C TRANSFER ALL HYDROCARBONS TO EXIT STREAM 1
      STRO1(8)=STRI(8)*STRI(NC+1)
      STRO1(11)=STRI(11)*STRI(NC+1)
      STRO1(12)=STRI(12)*STRI(NC+1)
      STRO1(13)=STRI(13)*STRI(NC+1)
      STRO1(14)=STRI(14)*STRI(NC+1)

C CALCULATE EXIT FLOW IN STREAM 1
      DO 15 I=1,NC
15    STRO1(NC+1)=STRO1(NC+1)+STRO1(I)
```

```
C TRANSFER OTHER COMPONENTS TO EXIT STREAM 2
      DO 20 I=1,NC
20    STRO2(I)=STRI(I)*STRI(NC+1)

C CORRECT STREAM 2 FOR REMOVED HYDROCARBONS
      STRO2(8)=0.0D0
      STRO2(11)=0.0D0
      STRO2(12)=0.0D0
      STRO2(13)=0.0D0
      STRO2(14)=0.0D0

C CALCULATE EXIT FLOW IN STREAM 2
      DO 25 I=1,NC
25    STRO2(NC+1)=STRO2(NC+1)+STRO2(I)

C CALCULATE NEW MOLE FRACTIONS
      IF(STRO1(NC+1).EQ.0.D0) GOTO 35
      DO 30 I=1,NC
30    STRO1(I)=STRO1(I)/STRO1(NC+1)

35    IF(STRO2(NC+1).EQ.0.D0) GOTO 45
      DO 40 I=1,NC
40    STRO2(I)=STRO2(I)/STRO2(NC+1)

C SET EXIT TEMP, PRESSURE, AND PHASE
45    STRO1(NC+2)=STRI(NC+2)
      STRO1(NC+3)=STRI(NC+3)
      STRO1(NC+5)=STRI(NC+5)

      STRO2(NC+2)=STRI(NC+2)
      STRO2(NC+3)=STRI(NC+3)
      STRO2(NC+5)=STRI(NC+5)

C CALCULATE EXIT ENTHALPIES
```

```
      DO 50 I=1,NC
      X1(I)=STRO1(I)
50    X2(I)=STRO2(I)

      CALL HEAT(0,1,NC,STRO1(NC+2),STRO1(NC+3),X1,HX,STRO1(NC+4))
      CALL HEAT(0,1,NC,STRO2(NC+2),STRO2(NC+3),X2,HX,STRO2(NC+4))

      RETURN
      END
```

```
      SUBROUTINE PTFLASH(NCOMP,STREAMF,PFLASH,TFLASH,STRL,STRV,
     1EPSILON,IPHASE,IMAX,ERROR)

C ISOTHERMAL FLASH
C
C LAST UPDATE: 06/16/86
C
C REFERENCE: KING, 1971, PGS. 75-80


      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER*2 ERROR
      INTEGER*2 NCOMP,IMAX,ITER
      REAL*8 STREAMF(25),PFEED,TFEED,F,ZF(20),HF,PFLASH,
     1 TFLASH,V,L,X(20),Y(20),EPSILON
      REAL*8 STRL(25),STRV(25)
      REAL*8 TBUB,TDEW,VF
      REAL*8 K(20)
      REAL*8 OLDF,NEW
      REAL*8 FDREV
      REAL*8 HT,HFI(20)

C SET ERROR RETURN TO AN ERROR CONDITION, SET RETURN ENTHALPIES TO 0
      ERROR=1
      ITER=0
      HTL=0.0D0
      HTV=0.0D0

C SET VALUES FOR MOLE FRACTIONS - FIRST GUESS
      DO 5 I=1,NCOMP
      ZF(I)=STREAMF(I)
      X(I)=ZF(I)
5     Y(I)=ZF(I)

C SET LOCAL STREAM VARIABLES
      HF=STREAMF(NCOMP+4)
      PFEED=STREAMF(NCOMP+3)
```

```
      TFEED=STREAMF(NCOMP+2)
      F=STREAMF(NCOMP+1)

C CALCULATE F(V/F) AT V/F=0 AND V/F=1 AND USE THESE VALUES TO DETERMINE
C IF MIXTURE CAN BE FLASHED
      CALL KVAL(0,NCOMP,TFLASH,PFLASH,X,Y,K)

      VF0=0.0D0
      VF1=1.0D0
      FVF0=0.0D0
      FVF1=0.0D0

      DO 7 I=1,NCOMP
      IF(K(I).EQ.0.0D0) GOTO 7
      FVF0=FVF0+(ZF(I)*(K(I)-1.0D0))/(VF0*(K(I)-1.0D0)+1.0D0)
      FVF1=FVF1+(ZF(I)*(K(I)-1.0D0))/(VF1*(K(I)-1.0D0)+1.0D0)
7     CONTINUE

C IF F(V/F) IS POSITIVE AT V/F=1 THEN HAVE SUPERHEATED VAPOR
      IF(FVF1.GT.0.0D0) THEN
       DO 10 I=1,NCOMP
10     X(I)=0.0D0
       V=F
       L=0.0D0
       IPHASE=2
       ERROR=3
       GOTO 100
      ENDIF

C IF F(V/F) IS NEGATIVE AT V/F=0 THEN HAVE SUBCOOLED LIQUID
      IF(FVF0.LT.0.0D0) THEN
       DO 15 I=1,NCOMP
15     Y(I)=0.0D0
       L=F
       V=0.0D0
       IPHASE=1
       ERROR=3
```

```
         GOTO 100
        ENDIF

C*********************************************************
C HAVE MIXTURE, PERFORM FLASH CALCULATION
C*********************************************************

C MAKE FIRST GUESS FOR TFLASH AND V/F
        CALL BUBDEW(NCOMP,TFEED,X,Y,TBUB,TDEW,PFLASH,EPSILON,2)
        VF=(TFLASH-TBUB)/(TDEW-TBUB)
        V=VF*F

C CALCULATE F
        OLDF=0.0
        DO 20 I=1,NCOMP
20      OLDF=OLDF+(ZF(I)*(K(I)-1))/(VF*(K(I)-1)+1.0D0)

C CALCULATE DF/DV
25      FDERV=0.0D0
        DO 30 I=1,NCOMP
30      FDERV=FDERV-(ZF(I)*((K(I)-1)**2))/((VF*(K(I)-1)+1.0D0)**2)

C CALCULATE NEW VF, V
        VFOLD=VF
        VF=VF-OLDF/FDERV

C BOUND VF BETWEEN 0 AND 1 BY AVERAGING OLD VALUE WITH 0 OR 1
        IF(VF.GT.1.D0) THEN
                VF=(VFOLD+1.D0)/2.D0
        ENDIF
        IF(VF.LT.0.0D0) THEN
                VF=VFOLD/2.D0
        ENDIF


        V=VF*F
```

```
C CALCULATE NEW X,Y
        L=F-V
        WRITE(6,99000) F,V,L
99000   FORMAT(X,'F=',D20.10,' V=',D20.10,' L=',D20.10)

        DO 40 I=1,NCOMP
        X(I)=ZF(I)/(((K(I)-1)*VF)+1)
40      Y(I)=X(I)*K(I)

C CALCULATE NEW K VALUES
        CALL KVAL(0,NCOMP,TFLASH,PFLASH,X,Y,K)

C CALCULATE NEW F FUNCTION
        NEWF=0.0
        DO 50 I=1,NCOMP
50      NEWF=NEWF+(ZF(I)*(K(I)-1))/(VF*(K(I)-1)+1.0D0)

        ITER=ITER+1

C CHECK FOR CONVERGENCE
        IF(ITER.GT.IMAX) GOTO 100
        IF(DABS(NEWF).GT.EPSILON) THEN
                OLDF=NEWF
                GOTO 25
        ELSE

C CONVERGED, SHOW NO ERROR (EXCEPT MIXED PHASE), HAVE MIXTURE AND THEN EXIT
                ERROR=2
                IPHASE=3
                GOTO 100
        ENDIF

C SET RETURN STREAM VALUES
C CALCULATE STREAM ENTHALPIES
100     IF(L.EQ.0.0D0) GOTO 110
        CALL HEAT(0,1,NCOMP,TFLASH,PFLASH,X,HFI,HTL)
110     IF(V.EQ.0.0D0) GOTO 120
```

```
      CALL HEAT(0,2,NCOMP,TFLASH,PFLASH,Y,HFI,HTV)

C SET EXIT STREAM COMPOSITIONS
120   DO 300 I=1,NCOMP
      STRL(I)=X(I)
300   STRV(I)=Y(I)

C FLOW, TEMP, PRESSURE, ENTHALPY, PHASE
      STRL(NCOMP+1)=L
      STRV(NCOMP+1)=V

      STRL(NCOMP+2)=TFLASH
      STRV(NCOMP+2)=TFLASH

      STRL(NCOMP+3)=PFLASH
      STRV(NCOMP+3)=PFLASH

      STRL(NCOMP+4)=HTL
      STRV(NCOMP+4)=HTV

      STRL(NCOMP+5)=1
      STRV(NCOMP+5)=2

      RETURN
      END
```

```
      SUBROUTINE BUBDEW(NCOMP,TFEED,XO,YO,TBUB,TDEW,PVESS,EPSILON,KEY)

C LAST UPDATE: 06/10/86

C THIS ROUTINE CALCULATES THE BUBBLE AND DEW POINTS GIVEN THE FEED
C PROPERTIES AND THE VESSEL TEMPERATURE
C IF KEY=0 THEN CALCULATE BUBBLE POINT
C    KEY=1 THEN CALCULATE DEW POINT
C    KEY=2 THEN CALCULATE BOTH

      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 XO(20),X(20),YO(20),Y(20),K(20),DKDT(20),K1(20)
      INTEGER*2 KEY

      IMAX=20

C RENORMALIZE MOLE FRACTIONS FOR ANY SOLID OR DISSOLVED SULFUR
C PRESENT IN THE INLET
      DO 1 I=1,NCOMP
      X(I)=XO(I)
1     Y(I)=YO(I)

            SUM0=1.0-X(17)-X(5)
            SUM1=1.0-Y(17)-Y(5)

C IF STREAM IS ALL SULFUR, SET BUBBLE AND DEW POINTS TO NORMAL SULFUR
C  BOILING POINT
      IF(SUM0.EQ.0) THEN
            TBUB=717.7D0
            TDEW=TBUB
            RETURN
      ENDIF

            DO 2 I=1,NCOMP
            X(I)=X(I)/SUM0
2           Y(I)=Y(I)/SUM1
            X(17)=0.0D0
```

```
      Y(17)=0.0D0
      X(5)=0.0D0
      Y(5)=0.0D0


      TTEMP=TFEED+100.0D0
      CALL KVAL(0,NCOMP,TFEED,PVESS,X,Y,K)
      CALL KVAL(0,NCOMP,TTEMP,PVESS,X,Y,K1)

      SUMB0=0.0D0
      SUMB1=0.0D0
      SUMD0=0.0D0
      SUMD1=0.0D0
      DO 100 I=1,NCOMP
      SUMB0=SUMB0+X(I)*K(I)
      SUMB1=SUMB1+X(I)*K1(I)
      IF(K(I).EQ.0.0D0) GOTO 100
      IF(K1(I).EQ.0.0D0) GOTO 100
      SUMD0=SUMD0+Y(I)/K(I)
      SUMD1=SUMD1+Y(I)/K1(I)
100   CONTINUE

      TBUBNEW=(1.0D0/TFEED)+(1.0D0/TTEMP-1.0D0/TFEED)*(DLOG(SUMB0)/
     1 (DLOG(SUMB0)-DLOG(SUMB1)))

      TBUBNEW=1.0D0/TBUBNEW

      TDEWNEW=(1.0D0/TFEED)+(1.0D0/TTEMP-1.0D0/TFEED)*(DLOG(SUMD0)/
     1 (DLOG(SUMD0)-DLOG(SUMD1)))

      TDEWNEW=1.0D0/TDEWNEW

      IF(KEY.EQ.1) GOTO 15

      ITER=0

5     TBUB=TBUBNEW
```

```
      CALL KVAL(0,NCOMP,TBUB,PVESS,X,Y,K)
      CALL KVAL(1,NCOMP,TBUB,PVESS,X,Y,DKDT)

      SUM0=0.000
      SUM1=0.000
      DO 10 I=1,NCOMP
      SUM0=SUM0+K(I)*X(I)
10    SUM1=SUM1+X(I)*DKDT(I)


      TBUBNEW=TBUB-DLOG(SUM0)/(SUM1/SUM0)

      IF(TBUBNEW.LT.0.000) THEN
             TBUBNEW=TBUB/2.000
      ENDIF

      ITER=ITER+1

      IF(ITER.GT.IMAX) GOTO 13

      IF((DABS(TBUB-TBUBNEW)).GT.EPSILON) GOTO 5


13    WRITE(6,10000) TBUB
10000 FORMAT(X,'TBUB= ',F6.2)

      IF(KEY.EQ.0) GOTO 25
15    ITER=0
17    TDEW=TDEWNEW

      CALL KVAL(0,NCOMP,TDEW,PVESS,X,Y,K)
      CALL KVAL(1,NCOMP,TDEW,PVESS,X,Y,DKDT)

      SUM0=0.000
      SUM1=0.000
      DO 20 I=1,NCOMP
```

```
      IF(K(I).EQ.0.000) GOTO 20
      SUM0=SUM0+Y(I)/K(I)
      SUM1=SUM1+(Y(I)/K(I)**2)*DKDT(I)
20    CONTINUE

      TDEWNEW=TDEW+DLOG(SUM0)/(SUM1/SUM0)

      IF(TDEWNEW.LT.0.000) THEN
             TDEWNEW=TDEW/2.000
      ENDIF

      ITER=ITER+1
      IF(ITER.GT.IMAX) GOTO 22

      IF((DABS(TDEW-TDEWNEW)).GT.EPSILON) GOTO 17

22    WRITE(6,10100) TDEW
10100 FORMAT(X,'TDEW= ',F6.2)

25    RETURN
      END
```

```
      REAL*8 FUNCTION SULFUR(T)
      IMPLICIT REAL*8(A-H,O-Z)

C  SULFUR SATURATION FUNCTION
C
C LAST UPDATE: 04/21/86

C CALCULATE AND RETURN THE MOLE FRACTION SULFUR AT SATURATION
C SET THE VALUES OF A AND B TO FIT THE EQN. OF THE FORM
C LN(WT% SULFUR)=A*TM/T+B
C FOR THE SOLVENT OF INTEREST
C NOTE: TM = MELTING POINT OF SULFUR (112.8 C)
C THE CONSTANTS A1,B1 APPLY FOR T<TM WHILE A2,B2 APPLY FOR T>TM

C DATA OF SCIAMANNA, 1986

      REAL*8 MOLWT(20)
      REAL*8 MOLR

      COMMON/MOLEC/MOLWT

      DATA A1,B1,A2,B2/-10.899,12.39,-5.64,7.12/
      DATA TM/385.95/

C CALCULATE WT% SULFUR, CHECK FOR TEMPERATURE RANGE
      IF(T.LT.TM) THEN
            WTS=DEXP(A1*TM/T+B1)
      ELSE
            WTS=DEXP(A2*TM/T+B2)
      ENDIF

C FOR TEMPERATURES LESS THEN 273.16 K, ASSUME ZERO SOLUBILITY
      IF(T.LT.273.16D0) THEN
            WTS=0.0D0
      ENDIF
```

```
C CONVERT CALCULATED VALUE TO MOLE FRACTION FOR THIS MIXTURE
      WTS=WTS/100.0

C GET RATIO OF MOLECULAR WT OF SULFUR TO THAT OF SOLVENT
      MOLR=32.0D0/MOLWT(1)

      SULFUR=WTS/(WTS+MOLR*(1-WTS))

      RETURN
      END
```

```
C*************************************************************************
C  Thermodynamic Data Subroutines- Notes
C*************************************************************************
C
C LAST UPDATE: 04/21/86
C
C 1.Calling parameters for Heat and Kval
C       HEAT(TYPE,PHASE,NCOMP,T,P,XEXPR,HEXPR,HT)
C       KVAL(TYPE,NCOMP,T,P,X,Y,KEXPR)

C where

C TYPE=0 for calculation of enthalpies (HEXPR= Vector of HL, HV, K)
C TYPE=1 for calculation of derivatives (HEXPR=Vector of DHLDT, DKDT, etc.)

C PHASE=1 for liquid and 2 for vapor
C (If sulfur is present the phase value may be 4 or 5. These routines
C treat the fluid as a liquid if phase=4 and a vapor if phase=5 for the
C purpose of calculating the properies of the other components)

C NCOMP=Number of Components

C T=Temperature in degrees K

C P=Pressure in KPa

C XEXPR= Vector of liquid or vapor mole fractions

C HT=Molar average enthalpy for the stream

C Components in the Subroutines are numbered as followed:
C 1=SOLV
C 2=H2O
C 3=H2S
C 4=SO2
C 5=SULFUR DISSOLVED IN SOLVENT
C 6=CO2
```

```
C 7=N2
C 8=C1
C 9=H2
C 10=CO
C 11=C2
C 12=C3
C 13=C4
C 14=C5+
C 15=CH3SH
C 16=C2H5SH
C 17=PURE SOLID AND LIQUID SULFUR (PHASE SET BY TEMPERATURE)
C 18=PURE LIQUID AND GASEOUS WATER
C 19=O2
C 20=NOT USED

C REFERENCES:
C A. ENTHALPIES
C       1. VAPOR HEAT CAPACITIES - REID ET AL. 1977
C       2. HEATS OF VAPORIZATION - REID ET AL., 1977
C       3. HEATS OF SOLUTION IN TETRAGLYME
C               H2S,SO2,CO2,C3,C4 - SCIAMANNA, 1986
C       4. ENTHALPIES OF FORMATION - CRC, 1984
C       5. ENTHALPY OF SOLID AND LIQUID SULFUR - CRC, 1984
C       6. ENTHALPY OF TETRAGLYME - GRANT CHEMICAL, 1982
C
C B. K VALUES
C       1. SOLUBILITIES OF GASES IN TETRAGLYME
C               H2S,SO2,CO2,C3,C4 - SCIAMANNA, 1986
C               OTHERS - ESTIMATED FROM SELEXOL VALUES - SWENY, 1980
C       2. SOLUBILITIES OF GASES IN WATER - WILHELM ET. AL., 1977
C       3. ANTOINE EQUATION CONSTANTS - REID ET. AL., 1977
C       4. TETRAGLYME VAPOR PRESSURE -  GRANT CHEMICAL, 1982
C
C C. ENTHALPY DEPARTURE FUNCTION AND FUGACITY COEFFICIENT.
C       1. EQUATIONS AND SOLUTION TO REDLICH-KWONG EQS. OF STATE
C               EDMISTER, 1968
C               SMITH AND VAN NESS, 1975
```

```
C       2. CRITICAL CONSTANTS - SMITH AND VAN NESS, 1975
```

```
C BEGIN THERMODYNAMICS SUBROUTINES

      SUBROUTINE HEAT(TYPE,IPHASE,NCOMP,T,P,X,H,HT)
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER*2 TYPE,PHASE,NCOMP
      INTEGER*2 INDEXI(20)
      REAL*8 X(20),H(20),HSOLN(20),HVAP(20)
      REAL*8 A(20),B(20),C(20),D(20),E(20)
      REAL*8 HDEP,PHI(20),SOLD(23)

      COMMON/THRM/HDEP,PHI,SOLD,Z

C AT PRESENT HV=AT+BT**2+CT**3+DT**4+E, HL=HV-HSOLN
C FOR COMPONENT 17, SULFUR, SIMILAR EXPRESSIONS ARE USED FOR THE
C CALCULATION OF THE LIQUID OR SOLID ENTHALPY - PHASE DETERMINED BY
C TEMPERATURE.
C

C CONSTANTS FOR EQNS.

      DATA A(1),B(1),C(1),D(1),E(1),HSOLN(1),HVAP(1)/397.14D0,
     1 0.0D0,0.0D0,0.0D0,-40123.2D0,78244.5,78244.5/

      DATA A(2),B(2),C(2),D(2),E(2),HSOLN(2),HVAP(2)/32.223D0,
     1 0.193D-2,1.055D-5,-3.594D-9,-2.418D5,40665.9,40665.9/

      DATA A(3),B(3),C(3),D(3),E(3),HSOLN(3),HVAP(3)/31.921D0,
     1 0.143D-2,2.431D-5,-11.758D-9,-0.202D5,18765.5,18660.6/

      DATA A(4),B(4),C(4),D(4),E(4),HSOLN(4),HVAP(4)/23.837D0,
     1 6.695D-2,-4.958D-5,13.272D-9,-2.969D5,35650.0,24915.7/

      DATA A(5),B(5),C(5),D(5),E(5),HSOLN(5),HVAP(5)/14.98D0,
     1 1.305D-2,0.0D0,0.0D0,-5622.9D0,0.0D0,0.0D0/

      DATA A(6),B(6),C(6),D(6),E(6),HSOLN(6),HVAP(6)/19.783D0,
```

```
1 7.339D-2,-5.598D-5,17.143D-9,-3.935D5,11337.8,17154.4/

   DATA A(7),B(7),C(7),D(7),E(7),HSOLN(7),HVAP(7)/31.130D0,
1 -1.356D-2,2.678D-5,-11.674D-9,0.0D0,5577.3,5577.3/

   DATA A(8),B(8),C(8),D(8),E(8),HSOLN(8),HVAP(8)/19.239D0,
1 5.209D-2,1.197D-5,-11.310D-9,-.748D5,8179.7,8179.7/

   DATA A(9),B(9),C(9),D(9),E(9),HSOLN(9),HVAP(9)/27.126D0,
1 .927D-2,-1.30D-5,7.640D-9,0.0D0,903.7,903.7/

   DATA A(10),B(10),C(10),D(10),E(10),HSOLN(10),HVAP(10)/30.850D0,
1 -1.284D-2,2.787D-5,-12.707D-9,-1.106D5,6041.7,6041.7/

   DATA A(11),B(11),C(11),D(11),E(11),HSOLN(11),HVAP(11)/5.406D0,
1 17.800D-2,-6.933D-5,8.707D-9,-.847D5,14706.8,14706.8/

   DATA A(12),B(12),C(12),D(12),E(12),HSOLN(12),HVAP(12)/-4.222D0,
1 30.607D-2,-15.854D-5,32.126D-9,-1.038D5,10300.0,18773.6/

   DATA A(13),B(13),C(13),D(13),E(13),HSOLN(13),HVAP(13)/9.481D0,
1 33.110D-2,-11.076D-5,-2.820D-9,-1.261D5,14970.0,22392.8/

   DATA A(14),B(14),C(14),D(14),E(14),HSOLN(14),HVAP(14)/-3.623D0,
1 48.704D-2,-25.787D-5,53.014D-9,-1.464D5,25773.4,25773.4/

   DATA A(15),B(15),C(15),D(15),E(15),HSOLN(15),HVAP(15)/13.260D0,
1 14.557D-2,-8.540D-5,20.737D-9,-0.230D5,24560.1,24560.1/

   DATA A(16),B(16),C(16),D(16),E(16),HSOLN(16),HVAP(16)/14.912D0,
1 23.494D-2,-13.553D-5,31.599D-9,-0.461D5,26777.6,26777.6/

   DATA A(17),B(17),C(17),D(17),E(17),HSOLN(17),HVAP(17)/0.0D0,
1 0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0/

   DATA A(18),B(18),C(18),D(18),E(18),HSOLN(18),HVAP(18)/32.223D0,
1 0.193D-2,1.055D-5,-3.594D-9,-2.418D5,40655.9,40655.9/
```

```
      DATA A(19),B(19),C(19),D(19),E(19),HSOLN(19),HVAP(19)/28.087D0,
     1 -0.00037D-2,1.745D-5,-10.644D-9,0.0D0,6819.9,6819.9/

      DATA A(20),B(20),C(20),D(20),E(20),HSOLN(20),HVAP(20)/0.0D0,
     1 0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0/


C DATA FOR PURE SULFUR CALCULATION

      DATA TMELTS,ASL,BSL,CSL,ASS,BSS,CSS/392.0D0,-5622.9D0,14.98D0,
     1 1.305D-2,-6972.9D0,22.59D0,1.04D-2/

      PHASE=IPHASE

      IF(TYPE.EQ.0) GOTO 1000

C CALCULATE DERIVATIVES
      DO 10 I=1,NCOMP
10    H(I)=A(I)+2*B(I)*T+3*C(I)*T**2+4*D(I)*T**3

C CALCULATE ENTHALPY DERIVATIVE FOR PURE SULFUR
      IF(T.GE.TMELTS) THEN
              H(17)=BSL+2*CSL*T
      ELSE
              H(17)=BSS+2*CSS*T
      ENDIF

      GOTO 1600

C CALCULATE ENTHALPIES
C FIRST CALCULATE THE ENTHALPY DEPARTURE FUNCTION
1000  CALL RKZ(P,T,X,NCOMP)

      DO 20 I=1,NCOMP
20    H(I)=E(I)+A(I)*T+B(I)*T**2+C(I)*T**3+D(I)*T**4
```

```
C CORRECT PHASE TYPE FOR  PRESENCE OF SULFUR
        IF(PHASE.GT.3) THEN
        PHASE=PHASE-3
        ENDIF

C CHECK FOR LIQUID PHASE
        IF(PHASE.EQ.1) THEN
C IF THERE IS A SMALL AMOUNT OF SOLVENT THEN USE HEAT OF SOLN
                IF(X(1).GT.0.01D0) THEN
                DO 30 I=1,NCOMP
30              H(I)=H(I)-HSOLN(I)
                ELSE
C ELSE USE THE HEAT OF VAPORIZATION
                DO 33 I=1,NCOMP
33              H(I)=H(I)-HVAP(I)
                ENDIF

        ELSE
C FOR VAPOR PHASE ADD DEPARTURE FUNCTION CORRECTION FOR PRESSURE
        DO 35 I=1,NCOMP
35      H(I)=H(I)+HDEP

        ENDIF

C CALCULATE SULFUR ENTHALPY
1500    IF(T.GE.TMELTS) THEN
                H(17)=ASL+BSL*T+CSL*T**2
        ELSE
                H(17)=ASS+BSS*T+CSS*T**2
        ENDIF

C CALCULATE AVERAGE MOLAR STREAM ENTHALPY OR AVERAGE DERIVATIVE
1600    HT=0.0D0
        DO 40 I=1,NCOMP
40      HT=HT+H(I)*X(I)

2000    RETURN
```

```
      END

      SUBROUTINE KVAL(TYPE,NCOMP,T,P,X,Y,K)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 K(20),X(20),Y(20),A(20),B(20),C(20),HO(20),HSOLN(20)
      REAL*8 HOW(20),HSOLNW(20)
      REAL*8 HDEP,PHI(20),SOLD(23),PHITEMP(20),DKDT(20)
      REAL*8 MOLWT(20)
      INTEGER*2 TYPE

      COMMON/THRM/HDEP,PHI,SOLD,Z
      COMMON/MOLEC/MOLWT

      DATA A(1),B(1),C(1),HO(1),HSOLN(1),HOW(1),HSOLNW(1)/0.0D0,0.0D0,
     1 0.0D0,1.0D0,0.0D0,1.0D0,0.0D0/

      DATA A(2),B(2),C(2),HO(2),HSOLN(2),HOW(2),HSOLNW(2)/18.3036,
     1 3816.44,-46.13,7.54,40665.9,1.D0,0.0D0/

      DATA A(3),B(3),C(3),HO(3),HSOLN(3),HOW(3),HSOLNW(3)/16.1040,
     1 1768.69,-26.06,420.67,18765.5,52442.0D0,16347.48D0/

      DATA A(4),B(4),C(4),HO(4),HSOLN(4),HOW(4),HSOLNW(4)/16.7680,
     1 2302.35,-35.97,16.96,35650.0,4002.D0,23081.99D0/

      DATA A(5),B(5),C(5),HO(5),HSOLN(5),HOW(5),HSOLNW(5)/0.0D0,
     1 0.0D0,0.0D0,1.0D0,0.0D0,1.D0,0.0D0/

      DATA A(6),B(6),C(6),HO(6),HSOLN(6),HOW(6),HSOLNW(6)/22.5898D0,
     1 3103.39,-0.16,2995.4,11337.8,156632.D0,17745.07D0/

      DATA A(7),B(7),C(7),HO(7),HSOLN(7),HOW(7),HSOLNW(7)/14.9542D0,
     1 588.72D0,-6.60D0,41250D0,5577.3,8084071.D0,8787.23D0/

      DATA A(8),B(8),C(8),HO(8),HSOLN(8),HOW(8),HSOLNW(8)/15.2243,
     1 897.84,-7.16,82500.0,8179.7,3779490.D0,11375.88D0/
```

```
      DATA A(9),B(9),C(9),HO(9),HSOLN(9),HOW(9),HSOLNW(9)/13.6333D0,
     1 164.9D0,3.19D0,41250D0,903.7,6867911.D0,3026.54D0/

      DATA A(10),B(10),C(10),HO(10),HSOLN(10),HOW(10),HSOLNW(10)/
     1 14.36860D0,530.22D0,-13.15D0,193359.0,6041.7,5521138.D0,8872.28D0/

      DATA A(11),B(11),C(11),HO(11),HSOLN(11),HOW(11),HSOLNW(11)/
     1 15.6637,1511.42,-17.16,12944.6,14706.8,2747805.D0,16261.1D0/

      DATA A(12),B(12),C(12),HO(12),HSOLN(12),HOW(12),HSOLNW(12)/
     1 15.7260,1872.46,-25.16,2142.0,10300.0,3353518.D0,19795.4D0/

      DATA A(13),B(13),C(13),HO(13),HSOLN(13),HOW(13),HSOLNW(13)/
     1 15.6782,2154.90,-34.42, 723.0,14970.0,4086309.D0,21618.2D0/

      DATA A(14),B(14),C(14),HO(14),HSOLN(14),HOW(14),HSOLNW(14)/
     1 15.8333,2477.07,-39.94,998.95,25773.4,8857999.D0,18879.9D0/

      DATA A(15),B(15),C(15),HO(15),HSOLN(15),HOW(15),HSOLNW(15)/
     1 16.1909,2338.38,-34.44,243.86,24560.1,10000000.D0,24560.1D0/

      DATA A(16),B(16),C(16),HO(16),HSOLN(16),HOW(16),HSOLNW(16)/
     1 16.0077,2497.23,-41.77,121.93,26777.6,1000000.D0,26777.6D0/

      DATA A(17),B(17),C(17),HO(17),HSOLN(17),HOW(17),HSOLN(17)/
     1 0.000,0.000,0.000,1.000,0.000,1.D0,0.000/

      DATA A(18),B(18),C(18),HO(18),HSOLN(18),HOW(18),HSOLNW(18)/
     1 18.3036,3816.44,-46.13,7.54,40665.9,1.000,0.000/

      DATA A(19),B(19),C(19),HO(19),HSOLN(19),HOW(19),HSOLNW(19)/
     1 15.4075,734.55,-6.45,412500.0,6819.9,4132809.D0,9686.9D0/

      DATA A(20),B(20),C(20),HO(20),HSOLN(20),HOW(20),HSOLNW(20)/
     1 0.000,0.000,0.000,1.000,0.000,1.D0,0.D0/
```

```
C*********************************************************************
C CALCULATE K VALUES

C BOUND THE TEMPERATURE TO AVOID CALCULATION OF GARBAGE
C HENRYS LAW VALUES
        IF(T.LT.10.D0) THEN
                TT=10.D0
        ELSE
                TT=T
        ENDIF

C CALCULATE SOLVENT AND WATER WEIGHT FRACTIONS IN THE MIXTURE
        WTS=0.0D0
        DO 5 I=1,NCOMP
5       WTS=WTS+MOLWT(I)*X(I)

        WTSOL=MOLWT(1)*X(1)/WTS
        WTH2O=MOLWT(2)*X(2)/WTS

C*********************************************************************
C IF THE COMBINED WEIGHT FRACTION OF WATER AND SOLVENT EXCEEDS 0.5 THEN
C THE K VALUES ARE CALCULATED AS A WEIGHTED AVERAGE OF THE HENRYS LAW
C VALUES FOR THE SELECTED COMPONENT IN H2O AND SOLVENT
C OTHERWISE CALCULATE K BY RAOULTS LAW

      IF((WTSOL+WTH2O).GT.0.5D0) THEN

C CALCULATE SOLVENT/WATER WT FRACTIONS EXCLUSIVE OF OTHER COMPONENTS
        WSOL=X(1)*MOLWT(1)
        WH2O=X(2)*MOLWT(2)
        WTOT=WSOL+WH2O
        WSOL=WSOL/WTOT
        WH2O=WH2O/WTOT


C*************************************************
C CALCULATE ALL COMPONENTS BY HENRYS LAW
```

```
      DO 100 I=1,NCOMP
      HENRS=HO(I)*DEXP(-(HSOLN(I)/8.314)*(1.0D0/TT-0.00336))
      HENRW=HOW(I)*DEXP(-(HSOLNW(I)/8.314)*(1.0D0/TT-0.00336))
      HENRY=DEXP(WSOL*DLOG(HENRS)+WH2O*DLOG(HENRW))
100   K(I)=HENRY/P

C*****************************************************************
C RECALCULATE WATER K BY RAOULTS LAW IF WEIGHT FRACTION WATER > 0.1
    IF(WTH2O.GT.0.1D0) THEN

      PS=A(2)-B(2)/(TT+C(2))

C PERFORM RANGE CHECK FOR LOW TEMPS -- I.E.
C CHECK FOR T CLOSE TO C(I) AND T LESS THAN C(I)
      IF((PS.LT.-10.D0).OR.(TT+C(2).LE.0.0D0)) THEN
              PS=1.0D-10
      ELSE
              PS=DEXP(PS)
      ENDIF

      K(2)=0.1333*PS/P
      K(18)=K(2)

    ENDIF


    ELSE

C*************************************************
C CALCULATE ALL COMPONENTS BY RAOULTS LAW
200   DO 210 I=1,NCOMP
      PS=A(I)-B(I)/(TT+C(I))

C PERFORM RANGE CHECK FOR LOW TEMPS -- I.E.
C CHECK FOR T CLOSE TO C(I) AND T LESS THAN C(I)
      IF((PS.LT.-10.0D0).OR.(TT+C(I).LT.0.D0)) THEN
              PS=1.D-10
```

```
      ELSE

      PS=DEXP(PS)

      ENDIF

210   K(I)=0.1333*PS/P

     ENDIF

C*************************************************************
C ALWAYS CALCULATE SOLVENT K (THE EQN BELOW IS FOR TETRAGLYME)
C BY RAOULTS LAW
      K(1)=(1.0D0/P)*DEXP(15.48*DLOG(TT)-91.04)

C*********************************************************
C ASSUME SOLID AND DISSOLVED SULFUR HAVE NO VAPOR PRESSURE
      K(5)=0.0D0
      K(17)=0.0D0

C*****************************************
C CALCULATE FUGACITY COEFICIENT CORRECTION
      CALL RKZ(P,TT,Y,NCOMP)

C*******************************
C FOR DERIVATIVES CALC DKDT
      IF(TYPE.EQ.1) GOTO 1000

C*****************************************
C CORRECT K VALUES FOR FUGACITY COEFFICIENT
      DO 320 I=1,NCOMP
320   K(I)=K(I)/PHI(I)

      RETURN

C*************************************************************************
C CALCULATE DERIVATIVES
```

```
C NOTE THAT FOR THE DERIVATIVES PHI IS ASSUMED TO BE INDEPENDENT OF T

C AGAIN MAKE CHECK ON SOLVENT AND WATER MOLE FRACTION

1000  IF((WTSOL+WTH2O).GT.0.5D0) THEN

C CALCULATE SOLVENT/WATER WT FRACTIONS EXCLUSIVE OF OTHER COMPONENTS
        WSOL=X(1)*MOLWT(1)
        WH2O=X(2)*MOLWT(2)
        WTOT=WSOL+WH2O
        WSOL=WSOL/WTOT
        WH2O=WH2O/WTOT

C*************************************************
C CALCULATE DERIVS OF HENRYS LAW VALUES
        DO 1100 I=1,NCOMP
        ALPHAS=HO(I)*DEXP(HSOLN(I)/2477.57)
        DHDTS=(ALPHAS*HSOLN(I)/(8.314*TT**2))*DEXP(-HSOLN(I)/(8.314*TT))
        ALPHAW=HOW(I)*DEXP(HSOLNW(I)/2477.57)
        DHDTW=(ALPHAW*HSOLNW(I)/(8.314*TT**2))*DEXP(-HSOLNW(I)/
     1 (8.314*TT))
        HENRS=HO(I)*DEXP(-(HSOLN(I)/8.314)*(1.0D0/TT-0.00336))
        HENRW=HOW(I)*DEXP(-(HSOLNW(I)/8.314)*(1.0D0/TT-0.00336))
        HENRY=DEXP(WSOL*DLOG(HENRS)+WH2O*DLOG(HENRW))
        HDT=WSOL*DHDTS/HENRS+WH2O*DHDTW/HENRW
1100    DKDT(I)=HENRY*HDT/P


C*****************************************************************
C RECALCULATE FOR WATER USING RAOULTS LAW IF WEIGHT FRACTION H2O>0.1
      IF(WTH2O.GT.0.1D0) THEN
        PMM=7.50*P
        ALPHA=B(2)/(PMM*(TT+C(2))**2)
        PS=A(2)-B(2)/(TT+C(2))
        IF(PS.LT.-10.D0) THEN
                PS=1.0D-10
        ELSE
```

```
                DKDT(2)=ALPHA*DEXP(PS)
        ENDIF

C PERFORM VALIDITY CHECK ON VAPOR PRESSURE CALCULATION
C IF NOT VALID (I.E. TEMP TOO LOW) THEN SET K TO ARBITRARY LOW VALUE
        IF((C(2)+TT).LT.0.0D0) THEN
                DKDT(2)=1.0D-10
        ENDIF

        DKDT(18)=DKDT(2)

      ENDIF

      ELSE

C*************************************************
C CALCULATE DERIVS OF RAOULTS LAW VALUES

C CONVERT P TO MM HG
1200    P=7.50*P

        DO 1210 I=1,NCOMP
        ALPHA=B(I)/(P*(TT+C(I))**2)
        PS=A(I)-B(I)/(TT+C(I))
        IF(PS.LT.-10.D0) THEN
                DKDT(I)=1.0D-10
        ELSE
                DKDT(I)=ALPHA*DEXP(PS)
        ENDIF

C PERFORM VALIDITY CHECK ON VAPOR PRESSURE CALCULATION
C IF NOT VALID (I.E. TEMP TOO LOW) THEN SET K TO ARBITRARY LOW VALUE
        IF((C(I)+TT).LT.0.0D0) THEN
                DKDT(I)=1.0D-10
        ENDIF

1210    CONTINUE
```

```
C CONVERT PRESSURE BACK TO KPA
        P=P/7.50

      ENDIF

C*************************************************************
C CALCULATE DERIVATIVE FOR SOLVENT (TETRAGLYME)
1300    DKDT(1)=(1.0D0/P)*(15.48/TT)*DEXP(15.48*DLOG(TT)-91.04)

C*********************************************
C CORRECT DERIVATIVES FOR FUGACITY COEFFICIENT

C MOVE COEFFICIENTS AT T INTO TEMPORARY STORAGE
        DO 1400 I=1,NCOMP
1400    PHITEMP(I)=PHI(I)

C TO GET DPHI/DT INCREMENT T

        CALL RKZ(P,TT+1.0D0,Y,NCOMP)

C NOW CALCULATE DERIVATIVES TO RETURN
        DO 1450 I=1,NCOMP
        DPHIDT=PHI(I)-PHITEMP(I)/1.0D0
1450    K(I)=(PHI(I)*DKDT(I)-K(I)*DPHIDT)/(PHI(I)*PHI(I))

        RETURN
        END
```

```
      SUBROUTINE RKZ(P,T,Y,NCOMP)
C
C LAST UPDATE: 04/21/86

C CALCULATES FUGACITY COEFFICIENTS USING THE REDLICH-KWONG EQUATION
C OF STATE. SEE NOTES FOR DETAILS
C REFERENCES: SMITH AND VAN NESS, 1975, PGS 272-276
C             EDMISTER, 1968

      IMPLICIT REAL*8(A-H,O-Z)

      REAL*8 Y(20),PHI(20),HDEP,SOLD(23)
      REAL*8 BI(20),AIJ(20,20),SUM(20)
      REAL*8 TC(20),VC(20),ZC(20),PC(20)

      COMMON/THRM/HDEP,PHI,SOLD,Z
      DATA TC(1),VC(1),ZC(1),PC(1)/705.8,0.0,0.0,0.0/
      DATA TC(2),VC(2),ZC(2),PC(2)/647.1,56.0,.230,217.6/
      DATA TC(3),VC(3),ZC(3),PC(3)/373.2,98.5,.284,88.2/
      DATA TC(4),VC(4),ZC(4),PC(4)/430.8,122.0,.268,77.8/
      DATA TC(5),VC(5),ZC(5),PC(5)/1314.0,0.0,0.0,0.0/
      DATA TC(6),VC(6),ZC(6),PC(6)/304.2,94.0,.274,72.8/
      DATA TC(7),VC(7),ZC(7),PC(7)/126.2,89.5,.290,33.5/
      DATA TC(8),VC(8),ZC(8),PC(8)/190.6,99.0,.288,45.4/
      DATA TC(9),VC(9),ZC(9),PC(9)/33.2,65.0,0.305,12.8/
      DATA TC(10),VC(10),ZC(10),PC(10)/132.9,93.1,.295,34.5/
      DATA TC(11),VC(11),ZC(11),PC(11)/305.4,148.0,.285,48.2/
      DATA TC(12),VC(12),ZC(12),PC(12)/369.8,203.0,.281,41.9/
      DATA TC(13),VC(13),ZC(13),PC(13)/425.2,255.0,.274,37.5/
      DATA TC(14),VC(14),ZC(14),PC(14)/469.6,304.0,.262,33.0/
      DATA TC(15),VC(15),ZC(15),PC(15)/470.0,145.0,.268,71.4/
      DATA TC(16),VC(16),ZC(16),PC(16)/499.0,207.0,.274,54.2/
      DATA TC(17),VC(17),ZC(17),PC(17)/1314.0,0.0,0.0,0.0/
      DATA TC(18),VC(18),ZC(18),PC(18)/647.1,0.0,0.0,0.0/
      DATA TC(19),VC(19),ZC(19),PC(19)/154.6,73.4,.288,49.8/
      DATA TC(20),VC(20),ZC(20),PC(20)/0.0,0.0,0.0,0.0/
C NOTE THAT THE CRITICAL TEMPS HAVE FOR PURE H2O, PURE SULFUR AND
```

```
C PURE SOLVENT HAVE BEEN INCLUDED TO PERMIT CALCULATION OF THE
C MIXTURE CRITICAL TEMP. HOWEVER, THESE COMPONENTS ARE NOT INCLUDED
C IN THE CALC OF THE FUGACITY COEFFS.


      DATA ATHIRD/0.3333333333333333/
      DATA TWO27,TWO3/.074074074074,.6666666666667/
      DATA A27TH/.037037037037000/
      DATA R/82.05D0/
      DATA RJO/8.314D0/
      DATA EPSILON/1.0D-4/
      DATA PI/3.14159265400/

C CONVERT P TO ATM FROM KPA
      PL=P
      PL=PL/101.325

C CHECK TO SEE IF WE HAVE ALREADY DONE THE CALCULATION FOR THESE CONDITIONS
      DO 2 I=1,NCOMP
      IF(SOLD(I).NE.Y(I)) GOTO 4
2     CONTINUE
      IF(SOLD(21).NE.T) GOTO 4
      IF(SOLD(22).NE.P) GOTO 4
C INPUT PARAMETERS MATCH. LEAVE VALUES UNCHANGED
      GOTO 400

C MAKE ROUGH ESTIMATE OF CRITICAL TEMP OF THE MIXTURE
C INITIALIZE THE PHI VALUES AND HDEP
4     HDEP=0.0D0
      TCMIX=0.0D0
      DO 3 I=1,NCOMP
      PHI(I)=1.0D0
3     TCMIX=TCMIX+Y(I)*TC(I)

C IF THE MIXTURE CRITICAL TEMP IS LESS THAN 100K OR THE CALCULATION
C TEMPERATURE IS LESS THAN 150K THEN
C THIS ROUTINE SUCKS POND WATER. UNDER THESE CONDITIONS ALL PHI(I)
```

```
C ARE LEFT SET TO 1.0D0 AND HDEP=0.0D0

      IF((TCMIX.LT.100.0D0).OR.(T.LT.150)) GOTO 150

C THE FUGACITY COEFFIENT CALC IS NOT NECESSARY IF THE PRESSURE IS LOW
      IF(PL.LT.2.D0) GOTO 150


      DO 5 I=1,NCOMP
      BI(I)=0.0D0
      SUM(I)=0.0D0
      DO 5 J=1,NCOMP
5     AIJ(I,J)=0.0D0

      A=0.0D0
      B=0.0D0

C CALCULATE ALL AIJ, BI

      DO 10 I=1,NCOMP
      IF(PC(I).EQ.0.0D0) GOTO 10
      BI(I)=0.0867D0*R*TC(I)/PC(I)
      B=B+Y(I)*BI(I)

      DO 10 J=1,NCOMP
      IF(PC(J).EQ.0.0D0) GOTO 10
      VCIJ=((VC(I)**ATHIRD+VC(J)**ATHIRD)/2.0)**3.0
      TCIJ=DSQRT(TC(I)*TC(J))
      ZCIJ=(ZC(I)+ZC(J))/2.0
      PCIJ=ZCIJ*R*TCIJ/VCIJ
      AIJ(I,J)=(.4278D0*R*R*TCIJ**2.5)/PCIJ
      ATEMP=Y(J)*AIJ(I,J)
      A=A+Y(I)*ATEMP
      SUM(I)=SUM(I)+ATEMP

10    CONTINUE
```

```
C MAKE SURE THAT SOMETHING USEFUL WAS IN THE GAS PHASE
C IF NOT THEN SET ALL THE PHI TO 1.0 AND EXIT

      IF((A.EQ.0.0D0).AND.(B.EQ.0.0D0)) THEN
            DO 25 I=1,NCOMP
25          PHI(I)=1.0D0
            GOTO 150
      ENDIF

      A2B=A/(B*R*T**1.5)
      BP=(B*PL)/(R*T)
      SQ=BP*(A2B-BP-1.D0)
      SR=-A2B*(BP**2)
      SM=SQ-ATHIRD
      SN=(SQ*ATHIRD+SR-TWO27)
      SN2=SN*SN/4.D0
      SM3=SM*SM*SM/27.D0
      DIS=SN2+SM3

      IF(DIS) 5000,6000,7000
5000  SGN=-1.0D0*DSIGN(1.0D0,SN)
      THETA=DACOS(SGN*DSQRT(-SN2/SM3))
      X1=2.D0*DSQRT(-SM/3.D0)*DCOS(THETA/3.D0)
      X2=2.D0*DSQRT(-SM/3.D0)*DCOS(THETA/3.D0+2.D0*PI/3.D0)
      X3=2.D0*DSQRT(-SM/3.D0)*DCOS(THETA/3.D0+4.D0*PI/3.D0)
      Z=DMAX1(X1,X2,X3)+ATHIRD

      GOTO 8000

6000  Z=1.D0
      WRITE(6,99000)
99000 FORMAT(X,'RKZ IS HOSED!!!!!!!!!')
      GOTO 8000

7000  BTEMP=-SN/2.D0+DSQRT(DIS)
      BM=DSIGN(DABS(BTEMP)**ATHIRD,BTEMP)
      BTEMP=-SN/2.D0-DSQRT(DIS)
```

```
      BN=DSIGN(DABS(BTEMP)**ATHIRD,BTEMP)
      Z=BM+BN+ATHIRD

C CALCULATE PHI VALUES

8000  IF(Z.LT.0.5D0) GOTO 150

      H=(B*PL)/(Z*R*T)
      T1=(Z-1.D0)/B
      T2=DLOG(Z-Z*H)
      T3=A/(B*R*T**1.5)
      T4=2.0D0/A
      T5=DLOG(1.0D0+H)

      DO 100 I=1,NCOMP

C CALCULATE FUGACITY COEFFICIENTS. SET UNKNOWN COMPONENTS TO PHI=1
      IF(PC(I).EQ.0.0D0) THEN
      PHI(I)=1.0D0
      ELSE
      PHI(I)=DEXP(BI(I)*T1-T2+T3*(BI(I)/B-T4*SUM(I))*T4)
      ENDIF

100   CONTINUE

C CALCULATE THE ENTHALPY DEPARTURE FUNCTION (KJ/KMOLE)
      HDEP=RJO*T*(((3.0D0*A)/(2.0D0*B*R*T**1.5))*DLOG(1+H)+1-Z)

C SAVE INPUT PARAMETERS FOR SUBSEQUENT ACCESSES
150   DO 200 I=1,NCOMP
200   SOLD(I)=Y(I)
      SOLD(21)=T
      SOLD(22)=P

400   RETURN
      END
```

```
      SUBROUTINE BAND(J)

C THE EVER POPULAR BAND AND MATINV SUBROUTINES
C REF: NEWMAN, 1968.

      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(18,18),B(18,18),C(18,25),D(18,37),E(18,19,23)
      DIMENSION G(18),X(18,18),Y(18,18)
      REAL*8 GAR(500)
      COMMON /BND/ A,B,C,D,G,X,Y,N,NJ
      COMMON/GARBAGE/GAR
      COMMON/EEE/E
101   FORMAT(' DETERM=0 AT J=',I4)
      NP1=N+1
      IF(J-2) 1,6,8
1     DO 2 I=1,N
      D(I,2*N+1)=G(I)
      DO 2 L=1,N
      LPN=L+N
2     D(I,LPN)=X(I,L)
      CALL MATINV(N,2*N+1,DETERM,B,D)
      IF (DETERM) 4,3,4
3     WRITE(6,101) J
4     DO 5 K=1,N
      E(K,NP1,1)=D(K,2*N+1)
      DO 5 L=1,N
      E(K,L,1)=-D(K,L)
      LPN=L+N
5     X(K,L)=-D(K,LPN)
      RETURN
6     DO 7 I=1,N
      DO 7 K=1,N
      DO 7 L=1,N
7     D(I,K)=D(I,K)+A(I,L)*X(L,K)
8     IF (J-NJ) 11,9,9
9     DO 10 I=1,N
      DO 10 L=1,N
```

```
      G(I)=G(I)-Y(I,L)*E(L,NP1,J-2)
      DO 10 M=1,N
10    A(I,L)=A(I,L)+Y(I,M)*E(M,L,J-2)
11    DO 12 I=1,N
      D(I,NP1)=-G(I)
      DO 12 L=1,N
      D(I,NP1)=D(I,NP1)+A(I,L)*E(L,NP1,J-1)
      DO 12 K=1,N
12    B(I,K)=B(I,K)+A(I,L)*E(L,K,J-1)
      CALL MATINV (N,NP1,DETERM,B,D)
      IF (DETERM) 14,13,14
13    WRITE(6,101) J
14    DO 15 K=1,N
      DO 15 M=1,NP1
15    E(K,M,J)=-D(K,M)
      IF (J-NJ) 20,16,16
16    DO 17 K=1,N
17    C(K,J)=E(K,NP1,J)
      DO 18 JJ=2,NJ
      M=NJ-JJ+1
      DO 18 K=1,N
      C(K,M)=E(K,NP1,M)
      DO 18 L=1,N
18    C(K,M)=C(K,M)+E(K,L,M)*C(L,M+1)
      DO 19 L=1,N
      DO 19 K=1,N
19    C(K,1)=C(K,1)+X(K,L)*C(L,3)
20    RETURN
      END
```

```
      SUBROUTINE MATINV(N,M,DETERM,B,D)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION ID(18),B(18,18),D(18,37)
      DETERM=1.D0
      DO 1 I=1,N
1     ID(I)=0.0D0
      DO 18 NN=1,N
      BMAX=0.0D0
      DO 6 I=1,N
      IF (ID(I)) 2,2,6
2     DO 5 J=1,N
      IF (ID(J)) 3,3,5
3     IF (DABS(B(I,J))-BMAX) 5,5,4
4     BMAX=DABS(B(I,J))
      IROW=I
      JCOL=J
5     CONTINUE
6     CONTINUE
      IF (BMAX) 7,7,8
7     DETERM=0.0D0
      RETURN
8     ID(JCOL)=1
      IF (JCOL-IROW) 9,12,9
9     DO 10 J=1,N
      SAVE=B(IROW,J)
      B(IROW,J)=B(JCOL,J)
10    B(JCOL,J)=SAVE
      DO 11 K=1,M
      SAVE=D(IROW,K)
      D(IROW,K)=D(JCOL,K)
11    D(JCOL,K)=SAVE
12    F=1.D0/B(JCOL,JCOL)
      DO 13 J=1,N
13    B(JCOL,J)=B(JCOL,J)*F
      DO 14 K=1,M
14    D(JCOL,K)=D(JCOL,K)*F
```

```
      DO 18 I=1,N
      IF (I-JCOL) 15,18,15
15    F=B(I,JCOL)
      DO 16 J=1,N
16    B(I,J)=B(I,J)-F*B(JCOL,J)
      DO 17 K=1,M
17    D(I,K)=D(I,K)-F*D(JCOL,K)
18    CONTINUE
      RETURN
      END
```

A-IV. Simulation Data Entry Programs

The programs listed below can be used to facilitate data entry to the simulation. These routines have been written to enable the user to create, modify, and examine the data files, STREAMIO.DAT and STREAMS.DAT, which provide the unit operation, unit interconnection, and initial stream information to the simulation program. The code is entirely menu-driven and makes extensive use of the report generation capabilities of the Knowledge Manager Data Base Management System Ver 1.06.

```
! BERKIN, MAIN DATA ENTRY PROGRAM
! INVOKE BY TYPING:
!  KMAN BERKIN
!
! KMAN DATABASE MANAGEMENT PROGRAM FOR USE WITH THE UCB SULFUR
! SIMULATION. THIS PROGRAM CAN BE EMPLOYED TO SETUP THE
! DATA FILES STREAMIO (UNIT OPERATIONS) AND STREAMS (STREAM DATA)
! FOR USE IN THE SIMULATION.
!
! LAST UPDATE: 04/10/86, DWN

E.LSTR=80
CLEAR
OUTPUT "ONE MOMENT PLEASE..."
! LOAD CONTEXT FILE FOR UNIT OPERATION INPUT FORMS
LOAD "OFORM.ICF"
CLEAR

! SET VARIOUS ENVIRONMENT VARIABLES
        E.SERR=TRUE
        E.SUPD=TRUE
        E.LNUM=7
        E.LSTR=80
        E.LMOD=FALSE
        E.ICOM=FALSE



! FORM FOR USE WITH STREAMS DATA ENTRY

FORM STRFORM AT 1,25 PUT "ENTER DATA FOR STREAM= "
        AT 1,48 PUT REC
        AT 2,2 PUT "COMPONENT              MOLE FRACTION"
        AT 4,2 PUT "SOLV"
        AT 4,28 PUT SOLV
        AT 4,28 GET SOLV
        AT 5,2 PUT "H2O"
```

```
AT 5,28 PUT H2O
AT 5,28 GET H2O
AT 6,2 PUT "H2S"
AT 6,28 PUT H2S
AT 6,28 GET H2S
AT 7,2 PUT "SO2"
AT 7,28 PUT SO2
AT 7,28 GET SO2
AT 8,2 PUT "DISSOLVED SULFUR"
AT 8,28 PUT DSULF
AT 8,28 GET DSULF
AT 9,2 PUT "CO2"
AT 9,28 PUT CO2
AT 9,28 GET CO2
AT 10,2 PUT "N2"
AT 10,28 PUT N2
AT 10,28 GET N2
AT 11,2 PUT "C1"
AT 11,28 PUT C1
AT 11,28 GET C1
AT 12,2 PUT "H2"
AT 12,28 PUT H2
AT 12,28 GET H2
AT 13,2 PUT "CO"
AT 13,28 PUT CO
AT 13,28 GET CO
AT 14,2 PUT "C2"
AT 14,28 PUT C2
AT 14,28 GET C2
AT 15,2 PUT "C3"
AT 15,28 PUT C3
AT 15,28 GET C3
AT 16,2 PUT "C4"
AT 16,28 PUT C4
AT 16,28 GET C4
AT 17,2 PUT "C5+"
AT 17,28 PUT C5
```

```
        AT 17,28 GET C5
        AT 18,2 PUT "CH3SH"
        AT 18,28 PUT CH3SH
        AT 18,28 GET CH3SH
        AT 19,2 PUT "C2H5SH"
        AT 19,28 PUT C2H5SH
        AT 19,28 GET C2H5SH
        AT 20,2 PUT "SOLID AND LIQUID SULFUR"
        AT 20,28 PUT SULF
        AT 20,28 GET SULF
        AT 21,2 PUT "PURE WATER"
        AT 21,28 PUT WATER
        AT 21,28 GET WATER
        AT 22,2 PUT "O2"
        AT 22,28 PUT O2
        AT 22,28 GET O2
        AT 23,2 PUT "FLOW="
        AT 23,7 PUT FLOW
        AT 23,7 GET FLOW
        AT 23,20 PUT "TEMP="
        AT 23,26 PUT TEMP
        AT 23,26 GET TEMP
        AT 23,34 PUT "PRESSURE="
        AT 23,43 PUT PRESS
        AT 23,43 GET PRESS
        AT 23,56 PUT "PHASE="
        AT 23,62 PUT PHASE
        AT 23,62 GET PHASE
ENDFORM

FORM OPMENU AT 1,60 PUT "UNIT"
        AT 1,65 PUT UNIT
        AT 2,25 PUT "OPERATIONS AVAILABLE"
        AT 4,1 PUT " 1=ABSORBER/STRIPPER/DISTILLATION"
        AT 4,50 PUT " 2=FLOW SPLITTER"
        AT 5,1 PUT " 3=ADIABATIC FLASH"
        AT 5,50 PUT " 4=HEAT EXCHANGER"
```

```
        AT 6,1 PUT " 5=REBOILER"
        AT 6,50 PUT " 6=CONDENSER"
        AT 7,1 PUT " 7=MIXER"
        AT 7,50 PUT " 8=SPLITTER"
        AT 8,1 PUT " 9=PUMP"
        AT 8,50 PUT "10=REACTOR"
        AT 9,1 PUT "11=SULFUR SATURATOR"
        AT 9,50 PUT "12=SULFUR SETTLER"
        AT 10,1 PUT "13=SULFUR WASHER"
        AT 10,50 PUT "14=MELTER/DECANTER"
        AT 11,1 PUT "15=FURNACE"
        AT 11,50 PUT "16=COMPRESSOR"
        AT 12,1 PUT "17=SULFUR CENTRIFUGE"
        AT 12,50 PUT "18=PARTIAL CONDENSER"
        AT 13,1 PUT "19=HEATER"
        AT 13,50 PUT "20=STORAGE UNIT"
        AT 14,1 PUT "21=CRYSTALLIZER"
        AT 14,50 PUT "22=SULFUR SPLITTER"
        AT 15,1 PUT "23=MAKE-UP"
        AT 15,50 PUT "24=TEMPERATURE SET"
        AT 16,1 PUT "25=HYDROCARBON SPLITTER"
        AT 23,1 PUT "OPERATION CODE SELECTION="
        AT 23,40 PUT OPCODE
        AT 23,40 GET OPCODE
ENDFORM

TTEMP=""

        CHOICE=0

        WHILE CHOICE<>11 DO

CLEAR
FINISH ALL

#ERRNO=0
```

```
AT  1,1 OUTPUT "DATABASE CREATION PROGRAM FOR BERKELEY SULFUR PROCESS"
AT  3,1 OUTPUT "YOU CAN:"
AT  4,5 OUTPUT " 1) CREATE A NEW OPCODE MATRIX"
AT  5,5 OUTPUT " 2) ENTER DATA INTO THE OPCODE MATRIX"
AT  6,5 OUTPUT " 3) INSERT A NEW UNIT INTO AN EXISTING OPCODE MATRIX"
AT  7,5 OUTPUT " 4) DELETE A UNIT FROM AN EXISTING OPCODE MATRIX"
AT  8,5 OUTPUT " 5) PRINT THE OPERATIONS SUMMARY"
AT 10,5 OUTPUT " 6) CREATE A NEW STREAMS MATRIX"
AT 11,5 OUTPUT " 7) ENTER DATA INTO THE STREAMS MATRIX"
AT 12,5 OUTPUT " 8) ADD A STREAM TO THE STREAMS MATRIX"
AT 14,5 OUTPUT " 9) NORMAL EXIT -- CREATE STREAMIO AND STREAMS FILES"
AT 16,5 OUTPUT "10) READ OLD STREAMIO AND STREAMS FILES INTO TABLES"
AT 18,5 OUTPUT "11) QUIT WITHOUT CREATING .DAT FILES"
AT 20,5 INPUT CHOICE USING "uu" WITH "YOUR SELECTION? "

CLEAR
TEST CHOICE

CASE 1:
STNO=0
! SETUP NEW OPCODE  MATRIX
        CLEAR
        USE STREAMIO
        ATTACH 1 TO STREAMIO
! EMPTY THE DATABASE OF THE OLD VALUES
        MARK ALL
        COMPRESS STREAMIO

        CLEAR
        AT 8,1 INPUT STNO USING "dd" WITH "NUMBER OF OPERATIONS = "
        CLEAR

! CREATE TABLE WITH STNO RECORDS
        UCNT=1
        WHILE STNO>0 DO
         ATTACH 1 TO STREAMIO
         UNIT=UCNT
```

```
        UCNT=UCNT+1
        STNO=STNO-1
       ENDWHILE

! ADD EXTRA BLANK RECORD WITH UNIT=99 AS END OF FILE DELIMITER
       OBTAIN LAST
       ATTACH 1 TO STREAMIO
       UNIT=99

       OBTAIN 0 FROM STREAMIO

BREAK


CASE 2:
! ENTER DATA INTO EXISTING STREAMIO TABLE
       CLEAR
       USE STREAMIO

       MORE="Y"
       REC=0
       OPC=0

       WHILE MORE="Y" DO
       INPUT REC USING "dd.dd" WITH "ENTER DESIRED UNIT NUMBER "
       OBTAIN FOR UNIT=REC
       IF(#FOUND) THEN
              CLEAR
              PUTFORM OPMENU
              TALLY OPMENU
              GETFORM OPMENU
              OPC=OPCODE
              CLEAR
              AT 1,60 OUTPUT "UNIT"
              AT 1,65 OUTPUT UNIT
              PERFORM UNITS
              CLEAR
```

```
        ELSE
                OUTPUT "ERROR! UNIT NOT FOUND"
                OUTPUT "STRIKE ANY KEY TO CONTINUE"
                WAIT
                CLEAR
        ENDIF

        INPUT MORE USING "u" WITH "MORE UNITS (Y/N)? "
        ENDWHILE

OBTAIN LAST
UNIT=99
CLEAR
BREAK

CASE 3:
! INSERT A NEW UNIT INTO EXISTING STREAMIO TABLE
UNO=0
REC=0
OPC=0
        CLEAR
        USE STREAMIO
        INPUT UNO USING "dd.dd" WITH "NEW UNIT NUMBER="
        OBTAIN LAST
        ATTACH 1 TO STREAMIO
! RESET END UNIT FLAG TO NEW END OF FILE
        UNIT=99
! BACK UP ONE RECORD TO NEW LAST UNIT
        OBTAIN PRIOR
        UNIT=UNO
        CLEAR
! PROMPT USER FOR DESIRED OPERATION
        PUTFORM OPMENU
        TALLY OPMENU
        GETFORM OPMENU
! SET UP THE UNIT
        CLEAR
```

```
        UNIT=UNO
        OPC=OPCODE
        AT 1,60 OUTPUT "UNIT"
        AT 1,65 OUTPUT UNIT
        PERFORM UNITS
        CLEAR
BREAK

CASE 4:
! DELETE A UNIT FROM STREAMIO
UNO=0
        CLEAR
        USE STREAMIO
        INPUT UNO USING "dd.dd" WITH "ENTER NUMBER OF UNIT TO REMOVE "
        OBTAIN FOR UNIT=UNO
        IF(#ERRNO<>36) THEN
                REC=CURREC(STREAMIO)
                MARK RANGE REC,REC
                COMPRESS STREAMIO
                CLEAR
                BREAK
        ELSE
                OUTPUT "THAT UNIT DOES NOT EXIST!"
                OUTPUT "HIT RETURN TO CONTINUE"
                WAIT
                BREAK
        ENDIF
BREAK

CASE 5:
! SEND THE ENTIRE OPERATIONS MATRIX TO THE PRINTER
        CLEAR
        OUTPUT "HIT RETURN WHEN THE PRINTER IS READY"
        WAIT
        PERFORM OPRINT
        BREAK
```

```
CASE 6:
STNO=0
! SETUP NEW STREAM  MATRIX
        CLEAR
        USE STREAMS
        ATTACH 1 TO STREAMS
! EMPTY THE DATABASE OF THE OLD VALUES
        MARK ALL
        COMPRESS STREAMS

        CLEAR
        AT 8,1 INPUT STNO USING "dd" WITH "NUMBER OF STREAMS = "
        CLEAR

! CREATE STREAMS TABLE WITH STNO RECORDS
        WHILE STNO>0
         ATTACH 1 TO STREAMS
         STNO=STNO-1
        ENDWHILE

! APPEND 1 BLANK RECORD TO THE DATABASE AND SET END OF DATA DELIMITER
        OBTAIN LAST
        ATTACH 1 TO STREAMS
        OBTAIN LAST
        FLOW=-1.0

        OBTAIN 0 FROM STREAMS

BREAK

CASE 7:
! ENTER DATA INTO EXISTING STREAMS TABLE
        CLEAR
        USE STREAMS

        MORE="Y"
        REC=0
```

```
        WHILE MORE="Y" DO
        INPUT REC USING "ddd" WITH "ENTER DESIRED STREAM NUMBER "
        OBTAIN REC
        IF((#ERRNO<>36) AND (FLOW<>-1)) THEN
                BROWSE RANGE REC,REC WITH STRFORM
                CLEAR
        ELSE
                OUTPUT "THAT STREAM DOES NOT EXIST!"
                OUTPUT "STRIKE ANY KEY TO CONTINUE"
                WAIT
        ENDIF
        CLEAR
        INPUT MORE USING "u" WITH "MORE STREAMS (Y/N)? "
        ENDWHILE


CLEAR
BREAK

CASE 8:
! ADD A STREAM TO THE END OF THE STREAMS TABLE
        CLEAR
        USE STREAMS

! APPEND 1 BLANK RECORD TO THE DATABASE
        E.LMOD=TRUE
        OBTAIN LAST
        ATTACH 1 TO STREAMS
        OBTAIN PRIOR
        FLOW=0.0
        REC=CURREC(STREAMS)
        PUTFORM STRFORM
        TALLY STRFORM
        GETFORM STRFORM
        E.LMOD=FALSE
CLEAR
```

```
BREAK

CASE 9:
! ALL DONE
! WRITE NEW STREAMIO AND STREAMS .DAT FILES
! CLOSE UP SHOP
        CHOICE=11
OK=""
CLEAR
OUTPUT "NEW STREAMIO AND STREAMS FILES ARE ABOUT TO BE CREATED"
OUTPUT "THIS WILL ERASE THE OLD FILES."
INPUT OK USING "u" WITH "IS THIS WHAT YOU WANT? "

IF OK<>"Y" THEN
        BREAK
ENDIF

! SET UP UNIT INDEX FOR USE WITH STREAMIO
USE STREAMIO
INDEX "STRIO" FOR STREAMIO BY AZ UNIT

FINISH ALL

USE STREAMIO WITH STRIO
USE STREAMS

! REQUEST NUMBER OF ITERATIONS USER WISHES TO PERFORM
ITNO=0
        CLEAR
        AT 10,1 OUTPUT "PLEASE ENTER THE NUMBER OF ITERATIONS WHICH YOU WOULD"
        AT 11,1 OUTPUT "LIKE THE PROGRAM TO PERFORM OR 0 TO ITERATE TO "
        AT 12,1 OUTPUT "CONVERGENCE (MAX=9)"
        AT 13,1 INPUT ITNO USING "dd" WITH "ITERATIONS="

                OBTAIN LAST FROM STREAMIO
                OPCODE=-ITNO
```

```
! WARN USER OF ZERO OPCODES IMBEDED IT STREAMIO
        RECCNT=CURREC(STREAMIO)-1
        OBTAIN 0 FROM STREAMIO
        REC=1
        WHILE RECCNT>0 DO
                OBTAIN REC FROM STREAMIO
                IF OPCODE=0 THEN
                 OUTPUT "WARNING! ZERO OPCODE ENCOUNTERED FOR UNIT ",UNIT," IN RECORD ",REC
                ENDIF
        REC=REC+1
        RECCNT=RECCNT-1
        ENDWHILE

! EXPORT DATA TO .DAT FILES
OUTPUT "CREATING STREAMIO.DAT"
E.CF=2
        CONVERT OPCODE,UNIT,PARAM1,PARAM2,PARAM3,PARAM4,PARAM5,PARAM6,PARAM7,\
        PARAM8,PARAM9,PARAM10,PARAM11,PARAM12,PARAM13,PARAM14,PARAM15,PARAM16,\
        PARAM17,PARAM18,PARAM19,PARAM20,PARAM21,PARAM22,PARAM23,PARAM24,\
        PARAM25,PARAM26,PARAM27,PARAM28 FROM STREAMIO TO "STREAMIO.DAT" ALL


OUTPUT "CREATING STREAMS.DAT"

        CONVERT SOLV,H2O,H2S,SO2,DSULF,CO2,N2,C1,H2,CO,C2,C3,C4,C5,\
        CH3SH,C2H5SH,SULF,WATER,O2,UNKN,FLOW,TEMP,PRESS,ENTH,PHASE FROM\
        STREAMS TO "STREAMS.DAT" ALL


! SET OPCODE DELIMITER ON STREAMIO BACK TO 0
OBTAIN LAST FROM STREAMIO
OPCODE=0

OUTPUT "ALL DONE! TAKE A BREAK!"

FINISH ALL
```

```
BREAK

CASE 10:
! READ EXITING DATA FILES FOR STREAMIO AND STREAMS BACK INTO THE
! TABLES
OK=" "
FNAME=" "
CLEAR
OUTPUT "WARNING! THIS OPERATION WILL OVERWRITE EXISTING DATA"
OUTPUT "IN THE STREAMS AND STREAMIO DATA TABLE"
INPUT OK USING "u" WITH "IS THIS WHAT YOU WANT? "

OUTPUT "READING FROM STREAMIO.DAT"

IF OK<>"Y" THEN
        BREAK
ENDIF

USE STREAMIO
MARK ALL
COMPRESS STREAMIO

ATTACH FROM "STREAMIO.DAT" TO STREAMIO

FINISH ALL

OUTPUT "READING FROM STREAMS.DAT"

USE STREAMS
MARK ALL
COMPRESS STREAMS

ATTACH FROM "STREAMS.DAT" TO STREAMS WITH SOLV,H2O,H2S,SO2,DSULF,\
        CO2,N2,C1,H2,CO,C2,C3,C4,C5,CH3SH,C2H5SH,SULF,WATER,O2,\
        UNKN,FLOW,TEMP,PRESS,ENTH,PHASE

FINISH ALL
```

```
BREAK

! END OF CASE LOOP
ENDTEST

! END OF MAIN LOOP
ENDWHILE
BYE
RETURN
```

```
! OFORM.IPF
! SUBROUTINE TO DEFINE THE FORMS FOR DATA ENTRY OF THE DESIRED
! UNIT OPERATION IN STREAMIO.ITB
! RUN THIS PROGRAM TO CREATE THE CONTEXT FILE (OFORM.ICF) FOR USE WITH
! THE DATA ENTRY PROGRAM BERKIN.IFP
! LAST UPDATE: 04/16/86


! ABSORBER/STRIPPER/DISTILLATION
FORM UFORM1 AT 2,20 PUT "ABSORBER/STRIPPER DISTILLATION UNIT"
        AT 4,1 PUT "NUMBER OF STAGES="
        AT 4,50 PUT PARAM1 using "dd"
        AT 4,50 GET PARAM1 using "dd"
        AT 5,1 PUT "PRESSURE (KPA)="
        AT 5,50 PUT PARAM2 using "ddddd.dd"
        AT 5,50 GET PARAM2 using "ddddd.dd"
        AT 6,1 PUT "VOLUME OF LIQUID ON A TRAY (M3)="
        AT 6,50 PUT PARAM3 using "dd.dd"
        AT 6,50 GET PARAM3 using "dd.dd"
        AT 7,1 PUT "LIQUID MOLAR DENSITY (KMOL/M3)="
        AT 7,50 PUT PARAM4 using "dd.dd"
        AT 7,50 GET PARAM4 using "dd.dd"
        AT 8,1 PUT "SECOND ORDER RATE CONSTANT="
        AT 8,50 PUT PARAM5 using "dddd.dd"
        AT 8,50 GET PARAM5 using "dddd.dd"
        AT 9,1 PUT "FEED1 STREAM NUMBER="
        AT 9,21 PUT PARAM6 using "dd"
        AT 9,21 GET PARAM6 using "dd"
        AT 9,40 PUT "FEED1 STAGE NUMBER="
        AT 9,59 PUT PARAM7 using "dd"
        AT 9,59 GET PARAM7 using "dd"
        AT 10,1 PUT "FEED2 STREAM NUMBER="
        AT 10,21 PUT PARAM8 using "dd"
        AT 10,21 GET PARAM8 using "dd"
        AT 10,40 PUT "FEED2 STAGE NUMBER="
        AT 10,59 PUT PARAM9 using "dd"
        AT 10,59 GET PARAM9 using "dd"
```

```
AT 11,1 PUT "FEED3 STREAM NUMBER="
AT 11,21 PUT PARAM10 using "dd"
AT 11,21 GET PARAM10 using "dd"
AT 11,40 PUT "FEED3 STAGE NUMBER="
AT 11,59 PUT PARAM11 using "dd"
AT 11,59 GET PARAM11 using "dd"
AT 12,1 PUT "FEED4 STREAM NUMBER="
AT 12,21 PUT PARAM12 using "dd"
AT 12,21 GET PARAM12 using "dd"
AT 12,40 PUT "FEED4 STAGE NUMBER="
AT 12,59 PUT PARAM13 using "dd"
AT 12,59 GET PARAM13 using "dd"
AT 13,1 PUT "FEED5 STREAM NUMBER="
AT 13,21 PUT PARAM14 using "dd"
AT 13,21 GET PARAM14 using "dd"
AT 13,40 PUT "FEED5 STAGE NUMBER="
AT 13,59 PUT PARAM15 using "dd"
AT 13,59 GET PARAM15 using "dd"
AT 15,1 PUT "EXIT VAPOR STREAM NUMBER="
AT 15,29 PUT PARAM16 using "dd"
AT 15,29 GET PARAM16 using "dd"
AT 15,43 PUT "EXIT LIQUID STREAM NUMBER="
AT 15,70 PUT PARAM17 using "dd"
AT 15,70 GET PARAM17 using "dd"
AT 16,1 PUT "REBOILER HEAT INPUT (KJ/S)="
AT 16,29 PUT PARAM18 using "ddddd.dd"
AT 16,29 GET PARAM18 using "ddddd.dd"
AT 16,43 PUT "CONDENSER TEMPERATURE (K)="
AT 16,70 PUT PARAM23 using "dddd.dd"
AT 16,70 GET PARAM23 using "dddd.dd"
AT 17,1 PUT "FRACTION OF CONDENSER LIQUID TO SIDE DRAW="
AT 17,50 PUT PARAM24 using "d.ddd"
AT 17,50 GET PARAM24 using "d.ddd"
AT 18,1 PUT "CONDENSER EXIT LIQUID STREAM NUMBER="
AT 18,50 PUT PARAM25 using "dd"
AT 18,50 GET PARAM25 using "dd"
AT 19,1 PUT "FIRST HEATER/INTERCOOLER STAGE="
```

```
        AT 19,33 PUT PARAM19 using "dd"
        AT 19,33 GET PARAM19 using "dd"
        AT 19,40 PUT "HEAT LOAD (KJ/S)="
        AT 19,60 PUT PARAM20 using "ddddd.dd"
        AT 19,60 GET PARAM20 using "ddddd.dd"
        AT 20,1 PUT "SECOND HEATER/INTERCOOLER STAGE="
        AT 20,33 PUT PARAM21 using "dd"
        AT 20,33 GET PARAM21 using "dd"
        AT 20,40 PUT "HEAT LOAD (KJ/S)="
        AT 20,60 PUT PARAM22 using "ddddd.dd"
        AT 20,60 GET PARAM22 using "ddddd.dd"
        AT 21,1 PUT "PROFILE FLAG="
        AT 21,15 PUT PARAM26 using "d"
        AT 21,15 GET PARAM26 using "d"
        AT 22,1 PUT "(=0 GENERATE NEW INITIAL GUESSES)"
        AT 23,1 PUT "(=1 USE PROFILE FROM PREVIOUS CALCULATION)"
ENDFORM


! FLOW SPLITTER
FORM UFORM2 AT 2,20 PUT "FLOW SPLITTER"
        AT 4,1 PUT "INLET STREAM NUMBER="
        AT 4,40 PUT PARAM1 using "dd"
        AT 4,40 GET PARAM1 using "dd"
        AT 5,1 PUT "OUTLET STREAM 1 NUMBER="
        AT 5,40 PUT PARAM2 using "dd"
        AT 5,40 GET PARAM2 using "dd"
        AT 6,1 PUT "OUTLET STREAM 2 NUMBER="
        AT 6,40 PUT PARAM3 using "dd"
        AT 6,40 GET PARAM3 using "dd"
        AT 7,1 PUT "FLOW TO STREAM 1="
        AT 7,40 PUT PARAM4 using "dddd.dddd"
        AT 7,40 GET PARAM4 using "dddd.dddd"
ENDFORM


! ADIABATIC FLASH
```

```
FORM UFORM3 AT 2,20 PUT "ADIABATIC FLASH"
        AT 4,1 PUT "INLET STREAM NUMBER="
        AT 4,40 PUT PARAM1 using "dd"
        AT 4,40 GET PARAM1 using "dd"
        AT 5,1 PUT "FLASH PRESSURE="
        AT 5,40 PUT PARAM2 using "ddddd.dd"
        AT 5,40 GET PARAM2 using "ddddd.dd"
        AT 6,1 PUT "OUTLET LIQUID STREAM NUMBER="
        AT 6,40 PUT PARAM3 using "dd"
        AT 6,40 GET PARAM3 using "dd"
        AT 7,1 PUT "OUTLET VAPOR STREAM NUMBER="
        AT 7,40 PUT PARAM4 using "dd"
        AT 7,40 GET PARAM4 using "dd"
ENDFORM


! HEAT EXCHANGER
FORM UFORM4 AT 2,20 PUT "HEAT EXCHANGER"
        AT 4,1 PUT "INLET STREAM 1 (HOT) NUMBER="
        AT 4,40 PUT PARAM1 using "dd"
        AT 4,40 GET PARAM1 using "dd"
        AT 5,1 PUT "INLET STREAM 2 (COLD) NUMBER="
        AT 5,40 PUT PARAM2 using "dd"
        AT 5,40 GET PARAM2 using "dd"
        AT 6,1 PUT "OUTLET STREAM 1 NUMBER="
        AT 6,40 PUT PARAM3 using "dd"
        AT 6,40 GET PARAM3 using "dd"
        AT 7,1 PUT "OUTLET STREAM 2 NUMBER="
        AT 7,40 PUT PARAM4 using "dd"
        AT 7,40 GET PARAM4 using "dd"
        AT 8,1 PUT "UA (KJ/S-K) ="
        AT 8,40 PUT PARAM5 using "ddddd.dd"
        AT 8,40 GET PARAM5 using "ddddd.dd"
ENDFORM


! REBOILER
```

```
FORM UFORM5 AT 2,20 PUT "REBOILER"
        AT 4,1 PUT "INLET STREAM NUMBER="
        AT 4,40 PUT PARAM1 using "dd"
        AT 4,40 GET PARAM1 using "dd"
        AT 5,1 PUT "OUTLET LIQUID STREAM NUMBER="
        AT 5,40 PUT PARAM2 using "dd"
        AT 5,40 GET PARAM2 using "dd"
        AT 6,1 PUT "OUTLET VAPOR STREAM NUMBER="
        AT 6,40 PUT PARAM3 using "dd"
        AT 6,40 GET PARAM3 using "dd"
        AT 7,1 PUT "PRESSURE="
        AT 7,40 PUT PARAM4 using "ddddd.dd"
        AT 7,40 GET PARAM4 using "ddddd.dd"
        AT 8,1 PUT "TEMPERATURE="
        AT 8,40 PUT PARAM5 using "dddd.dd"
        AT 8,40 GET PARAM5 using "dddd.dd"
ENDFORM


! CONDENSER
FORM UFORM6 AT 2,20 PUT "CONDENSER"
        AT 4,1 PUT "INLET STREAM NUMBER"
        AT 4,40 PUT PARAM1 using "dd"
        AT 4,40 GET PARAM1 using "dd"
        AT 5,1 PUT "OUTLET STREAM NUMBER="
        AT 5,40 PUT PARAM2 using "dd"
        AT 5,40 GET PARAM2 using "dd"
        AT 6,1 PUT "PRESSURE="
        AT 6,40 PUT PARAM3 using "ddddd.dd"
        AT 6,40 GET PARAM3 using "ddddd.dd"
ENDFORM


! MIXER
FORM UFORM7 AT 2,20 PUT "MIXER"
        AT 4,1 PUT "INLET STREAM 1 NUMBER="
        AT 4,40 PUT PARAM1 using "dd"
```

```
        AT 4,40 GET PARAM1 using "dd"
        AT 5,1 PUT "INLET STREAM 2 NUMBER="
        AT 5,40 PUT PARAM2 using "dd"
        AT 5,40 GET PARAM2 using "dd"
        AT 6,1 PUT "OUTLET STREAM NUMBER="
        AT 6,40 PUT PARAM3 using "dd"
        AT 6,40 GET PARAM3 using "dd"
        AT 7,1 PUT "PRESSURE="
        AT 7,40 PUT PARAM4 using "ddddd.dd"
        AT 7,40 GET PARAM4 using "ddddd.dd"
ENDFORM


! SPLITTER
FORM UFORM8 AT 2,20 PUT "SPLITTER"
        AT 4,1 PUT "INLET STREAM NUMBER="
        AT 4,40 PUT PARAM1 using "dd"
        AT 4,40 GET PARAM1 using "dd"
        AT 5,1 PUT "OUTLET STREAM 1 NUMBER="
        AT 5,40 PUT PARAM2 using "dd"
        AT 5,40 GET PARAM2 using "dd"
        AT 6,1 PUT "OUTLET STREAM 2 NUMBER="
        AT 6,40 PUT PARAM3 using "dd"
        AT 6,40 GET PARAM3 using "dd"
        AT 7,1 PUT "FRACTION OF INLET TO OUTLET STREAM 1="
        AT 7,50 PUT PARAM4 using "d.ddd"
        AT 7,50 GET PARAM4 using "d.ddd"
ENDFORM


! PUMP
FORM UFORM9 AT 2,20 PUT "PUMP"
        AT 4,1 PUT "INLET STREAM NUMBER="
        AT 4,40 PUT PARAM1 using "dd"
        AT 4,40 GET PARAM1 using "dd"
        AT 5,1 PUT "OUTLET STREAM NUMBER="
        AT 5,40 PUT PARAM2 using "dd"
```

```
        AT 5,40 GET PARAM2 using "dd"
        AT 6,1 PUT "PRESSURE IN OUTLET="
        AT 6,40 PUT PARAM3 using "ddddd.dd"
        AT 6,40 GET PARAM3 using "ddddd.dd"
        AT 7,1 PUT "LIQUID MOLAR DENSITY (KMOLE/M3)="
        AT 7,40 PUT PARAM4 using "dd.dd"
        AT 7,40 GET PARAM4 using "dd.dd"
ENDFORM


! REACTOR
FORM UFORM10 AT 2,20 PUT "REACTOR"
        AT 4,1 PUT "INLET STREAM 1 NUMBER="
        AT 4,40 PUT PARAM1 using "dd"
        AT 4,40 GET PARAM1 using "dd"
        AT 5,1 PUT "INLET STREAM 2 NUMBER="
        AT 5,40 PUT PARAM2 using "dd"
        AT 5,40 GET PARAM2 using "dd"
        AT 6,1 PUT "OUTLET STREAM NUMBER="
        AT 6,40 PUT PARAM3 using "dd"
        AT 6,40 GET PARAM3 using "dd"
        AT 7,1 PUT "PRESSURE="
        AT 7,40 PUT PARAM4 using "ddddd.dd"
        AT 7,40 GET PARAM4 using "ddddd.dd"
        AT 8,1 PUT "HEAT REMOVED (KJ/S) ="
        AT 8,40 PUT PARAM5 using "ddddd.dd"
        AT 8,40 GET PARAM5 using "ddddd.dd"
        AT 9,1 PUT "FRACTION CONVERSION="
        AT 9,40 PUT PARAM6 using "d.ddddd"
        AT 9,40 GET PARAM6 using "d.ddddd"
        AT 10,1 PUT "SECOND ORDER RATE CONSTANT="
        AT 10,40 PUT PARAM7 using "ddddd.dd"
        AT 10,40 GET PARAM7 using "ddddd.dd"
ENDFORM


! SULFUR SATURATOR
```

```
FORM UFORM11 AT 2,20 PUT "SULFUR SATURATOR"
        AT 4,1 PUT "INLET STREAM NUMBER="
        AT 4,40 PUT PARAM1 using "dd"
        AT 4,40 GET PARAM1 using "dd"
        AT 5,1 PUT "OUTLET STREAM NUMBER="
        AT 5,40 PUT PARAM2 using "dd"
        AT 5,40 GET PARAM2 using "dd"
        AT 7,1 PUT "TEMPERATURE="
        AT 7,40 PUT PARAM3 using "ddddd.dd"
        AT 7,40 GET PARAM3 using "ddddd.dd"
        AT 8,1 PUT "PRESSURE="
        AT 8,40 PUT PARAM4 using "ddddd.dd"
        AT 8,40 GET PARAM4 using "ddddd.dd"
ENDFORM


! SULFUR SETTLER
FORM UFORM12 AT 2,20 PUT "SULFUR SETTLER"
        AT 4,1 PUT "INLET STREAM NUMBER="
        AT 4,50 PUT PARAM1 using "dd"
        AT 4,50 GET PARAM1 using "dd"
        AT 5,1 PUT "OUTLET SULFUR STREAM NUMBER="
        AT 5,50 PUT PARAM2 using "dd"
        AT 5,50 GET PARAM2 using "dd"
        AT 6,1 PUT "OUTLET SOLVENT STREAM NUMBER="
        AT 6,50 PUT PARAM3 using "dd"
        AT 6,50 GET PARAM3 using "dd"
        AT 7,1 PUT "WT% SULFUR IN SULFUR STREAM="
        AT 7,50 PUT PARAM4 using "ddd.dd"
        AT 7,50 GET PARAM4 using "ddd.dd"
        AT 8,1 PUT "PRESSURE="
        AT 8,50 PUT PARAM5 using "ddddd.dd"
        AT 8,50 GET PARAM5 using "ddddd.dd"
ENDFORM


! SULFUR WASHER
```

```
FORM UFORM13 AT 2,20 PUT "SULFUR WASHER"
        AT 4,1 PUT "INLET SULFUR STREAM NUMBER="
        AT 4,50 PUT PARAM1 using "dd"
        AT 4,50 GET PARAM1 using "dd"
        AT 5,1 PUT "INLET WASH STREAM NUMBER="
        AT 5,50 PUT PARAM2 using "dd"
        AT 5,50 GET PARAM2 using "dd"
        AT 6,1 PUT "OUTLET WASH STREAM NUMBER="
        AT 6,50 PUT PARAM3 using "dd"
        AT 6,50 GET PARAM3 using "dd"
        AT 7,1 PUT "OUTLET SULFUR STREAM NUMBER="
        AT 7,50 PUT PARAM4 using "dd"
        AT 7,50 GET PARAM4 using "dd"
        AT 8,1 PUT "WT% SULFUR IN SULFUR STREAM="
        AT 8,50 PUT PARAM5 using "ddd.dd"
        AT 8,50 GET PARAM5 using "ddd.dd"
        AT 9,1 PUT "PRESSURE="
        AT 9,50 PUT PARAM6 using "ddddd.dd"
        AT 9,50 GET PARAM6 using "ddddd.dd"
ENDFORM


! MELTER/DECANTER
FORM UFORM14 AT 2,20 PUT "MELTER/DECANTER"
        AT 4,1 PUT "INLET STREAM NUMBER="
        AT 4,50 PUT PARAM1 using "dd"
        AT 4,50 GET PARAM1 using "dd"
        AT 5,1 PUT "OUTLET NON-SULFUR STREAM NUMBER="
        AT 5,50 PUT PARAM2 using "dd"
        AT 5,50 GET PARAM2 using "dd"
        AT 6,1 PUT "OUTLET SULFUR STREAM NUMBER="
        AT 6,50 PUT PARAM3 using "dd"
        AT 6,50 GET PARAM3 using "dd"
        AT 7,1 PUT "TEMPERATURE="
        AT 7,50 PUT PARAM4 using "ddddd.dd"
        AT 7,50 GET PARAM4 using "ddddd.dd"
        AT 8,1 PUT "PRESSURE="
```

```
        AT 8,50 PUT PARAM5 using "ddddd.dd"
        AT 8,50 GET PARAM5 using "ddddd.dd"
ENDFORM


! FURNACE
FORM UFORM15 AT 2,20 PUT "FURNACE"
        AT 4,1 PUT "INLET SULFUR STREAM NUMBER="
        AT 4,50 PUT PARAM1 using "dd"
        AT 4,50 GET PARAM1 using "dd"
        AT 5,1 PUT "INLET RESIDUAL STREAM NUMBER="
        AT 5,50 PUT PARAM2 using "dd"
        AT 5,50 GET PARAM2 using "dd"
        AT 6,1 PUT "INLET AIR STREAM NUMBER="
        AT 6,50 PUT PARAM3 using "dd"
        AT 6,50 GET PARAM3 using "dd"
        AT 7,1 PUT "FLOW OF SO2 OUT OF FURNACE="
        AT 7,50 PUT PARAM4 using "dddd.ddddd"
        AT 7,50 GET PARAM4 using "dddd.ddddd"
        AT 8,1 PUT "OUTLET STREAM NUMBER="
        AT 8,50 PUT PARAM5 using "dd"
        AT 8,50 GET PARAM5 using "dd"
        AT 9,1 PUT "OUTLET TEMPERATURE="
        AT 9,50 PUT PARAM6 using "ddddd.dd"
        AT 9,50 GET PARAM6 using "ddddd.dd"
        AT 10,1 PUT "OUTLET PRESSURE="
        AT 10,50 PUT PARAM7 using "ddddd.dd"
        AT 10,50 GET PARAM7 using "ddddd.dd"
ENDFORM


! COMPRESSOR
FORM UFORM16 AT 2,20 PUT "COMPRESSOR"
        AT 4,1 PUT "INLET STREAM NUMBER="
        AT 4,50 PUT PARAM1 using "dd"
        AT 4,50 GET PARAM1 using "dd"
        AT 5,1 PUT "OUTLET STREAM NUMBER="
```

```
        AT 5,50 PUT PARAM2 using "dd"
        AT 5,50 GET PARAM2 using "dd"
        AT 6,1 PUT "OUTLET PRESSURE="
        AT 6,50 PUT PARAM3 using "ddddd.dd"
        AT 6,50 GET PARAM3 using "ddddd.dd"
ENDFORM


! SULFUR CENTRIFUGE
FORM UFORM17 AT 2,20 PUT "SULFUR CENTRIFUGE"
        AT 4,1 PUT "INLET STREAM NUMBER="
        AT 4,50 PUT PARAM1 using "dd"
        AT 4,50 GET PARAM1 using "dd"
        AT 5,1 PUT "INLET WASH STREAM NUMBER="
        AT 5,50 PUT PARAM2 using "dd"
        AT 5,50 GET PARAM2 using "dd"
        AT 6,1 PUT "OUTLET SULFUR SLURRY STREAM NUMBER="
        AT 6,50 PUT PARAM3 using "dd"
        AT 6,50 GET PARAM3 using "dd"
        AT 7,1 PUT "OUTLET LIQUID STREAM NUMBER="
        AT 7,50 PUT PARAM4 using "dd"
        AT 7,50 GET PARAM4 using "dd"
        AT 8,1 PUT "PRESSURE="
        AT 8,50 PUT PARAM5 using "ddddd.dd"
        AT 8,50 GET PARAM5 using "ddddd.dd"
        AT 9,1 PUT "WT% SULFUR TO SLURRY IN FIRST SEPARATION="
        AT 9,50 PUT PARAM6 using "ddd.dd"
        AT 9,50 GET PARAM6 using "ddd.dd"
        AT 10,1 PUT "WT% SULFUR TO SLURRY IN EXIT="
        AT 10,50 PUT PARAM7 using "ddd.dd"
        AT 10,50 GET PARAM7 using "ddd.dd"
ENDFORM


! PARTIAL CONDENSER
FORM UFORM18 AT 2,20 PUT "PARTIAL CONDENSER"
        AT 4,1 PUT "INLET STREAM NUMBER="
```

```
        AT 4,50 PUT PARAM1 using "dd"
        AT 4,50 GET PARAM1 using "dd"
        AT 5,1 PUT "PRESSURE IN CONDENSER="
        AT 5,50 PUT PARAM2 using "ddddd.dd"
        AT 5,50 GET PARAM2 using "ddddd.dd"
        AT 6,1 PUT "TEMPERATURE="
        AT 6,50 PUT PARAM3 using "ddddd.dd"
        AT 6,50 GET PARAM3 using "ddddd.dd"
        AT 7,1 PUT "OUTLET LIQUID STREAM NUMBER="
        AT 7,50 PUT PARAM4 using "dd"
        AT 7,50 GET PARAM4 using "dd"
        AT 8,1 PUT "OUTLET VAPOR STREAM NUMBER="
        AT 8,50 PUT PARAM5 using "dd"
        AT 8,50 GET PARAM5 using "dd"
ENDFORM


! HEATER
FORM UFORM19 AT 2,20 PUT "HEATER"
        AT 4,1 PUT "INLET STREAM NUMBER="
        AT 4,50 PUT PARAM1 using "dd"
        AT 4,50 GET PARAM1 using "dd"
        AT 5,1 PUT "OUTLET STREAM NUMBER="
        AT 5,50 PUT PARAM2 using "dd"
        AT 5,50 GET PARAM2 using "dd"
        AT 6,1 PUT "HEAT ADDED (KJ/S) ="
        AT 6,50 PUT PARAM3 using "ddddd.dd"
        AT 6,50 GET PARAM3 using "ddddd.dd"
ENDFORM


! STORAGE
FORM UFORM20 AT 2,20 PUT "STORAGE UNIT"
        AT 4,1 PUT "INLET STREAM NUMBER="
        AT 4,50 PUT PARAM1 using "dd"
        AT 4,50 GET PARAM1 using "dd"
        AT 5,1 PUT "OUTLET STREAM NUMBER="
```

```
        AT 5,50 PUT PARAM2 using "dd"
        AT 5,50 GET PARAM2 using "dd"
        AT 6,1 PUT "OUTLET FLOW OF COMPONENT "
        AT 6,25 PUT PARAM4 using "dd"
        AT 6,25 GET PARAM4 using "dd"
        AT 6,40 PUT " (KMOLE/S) ="
        AT 6,52 PUT PARAM3 using "ddddd.dddd"
        AT 6,52 GET PARAM3 using "ddddd.dddd"
        AT 7,1 PUT "OUTLET TEMPERATURE="
        AT 7,52 PUT PARAM5 using "ddddd.dd"
        AT 7,52 GET PARAM5 using "ddddd.dd"
        AT 9,1 PUT "THIS UNIT SETS THE FLOW OF THE DESIRED COMPONENT"
        AT 10,1 PUT "AND SCALES THE OUTPUT FLOWS OF THE OTHER COMPONENTS"
        AT 11,1 PUT "AT THE SAME COMPOSITION AS THE INLET"

ENDFORM


! CRYSTALLIZER
FORM UFORM21 AT 2,20 PUT "CRYSTALLIZER"
        AT 4,1 PUT "INLET STREAM NUMBER="
        AT 4,50 PUT PARAM1 using "dd"
        AT 4,50 GET PARAM1 using "dd"
        AT 5,1 PUT "OUTLET STREAM NUMBER="
        AT 5,50 PUT PARAM2 using "dd"
        AT 5,50 GET PARAM2 using "dd"
ENDFORM


! SULFUR SPLITTER
FORM UFORM22 AT 2,20 PUT "SULFUR SPLITTER"
        AT 4,1 PUT "INLET STREAM NUMBER="
        AT 4,50 PUT PARAM1 using "dd"
        AT 4,50 GET PARAM1 using "dd"
        AT 5,1 PUT "OUTLET SULFUR STREAM NUMBER="
        AT 5,50 PUT PARAM2 using "dd"
        AT 5,50 GET PARAM2 using "dd"
```

```
        AT 6,1 PUT "OUTLET LIQUID STREAM NUMBER="
        AT 6,50 PUT PARAM3 using "dd"
        AT 6,50 GET PARAM3 using "dd"
        AT 7,1 PUT "FRACTION OF LIQUID TO SULFUR STREAM="
        AT 7,50 PUT PARAM4 using "d.dddd"
        AT 7,50 GET PARAM5 using "d.dddd"
ENDFORM


! MAKEUP
FORM UFORM23 AT 2,20 PUT "MAKEUP UNIT"
        AT 4,1 PUT "INLET STREAM NUMBER="
        AT 4,50 PUT PARAM1 using "dd"
        AT 4,50 GET PARAM1 using "dd"
        AT 5,1 PUT "OUTLET STREAM NUMBER="
        AT 5,50 PUT PARAM2 using "dd"
        AT 5,50 GET PARAM2 using "dd"
        AT 6,1 PUT "OUTLET FLOW OF COMPONENT "
        AT 6,25 PUT PARAM4 using "dd"
        AT 6,25 GET PARAM4 using "dd"
        AT 6,40 PUT " (KMOLE/S) ="
        AT 6,55 PUT PARAM3 using "ddddd.ddddd"
        AT 6,55 GET PARAM3 using "ddddd.ddddd"
        AT 9,1 PUT "THIS UNIT SETS THE OUTLET FLOW OF THE DESIRED COMPONENT"
        AT 10,1 PUT "WHILE PASSING THE FLOWS OF THE OTHER COMPONENTS THROUGH"
        AT 11,1 PUT "FROM THE INLET TO THE OUTLET"
ENDFORM


! TSET
FORM UFORM24 AT 2,20 PUT "TSET ROUTINE"
        AT 4,1 PUT "INLET STREAM NUMBER="
        AT 4,50 PUT PARAM1 using "dd"
        AT 4,50 GET PARAM1 using "dd"
        AT 5,1 PUT "OUTLET STREAM NUMBER="
        AT 5,50 PUT PARAM2 using "dd"
        AT 5,50 GET PARAM2 using "dd"
```

```
        AT 6,1 PUT "SET TEMPERATURE OF OUTLET="
        AT 6,50 PUT PARAM3 using "ddddd.dd"
        AT 6,50 GET PARAM3 using "ddddd.dd"
ENDFORM


! HYDROCARBON SPLITTER
FORM UFORM25 AT 2,20 PUT "HYDROCARBON SPLITTER"
        AT 4,1 PUT "INLET STREAM NUMBER="
        AT 4,50 PUT PARAM1 using "dd"
        AT 4,50 GET PARAM1 using "dd"
        AT 5,1 PUT "OUTLET HYDROCARBON STREAM NUMBER="
        AT 5,50 PUT PARAM2 using "dd"
        AT 5,50 GET PARAM2 using "dd"
        AT 6,1 PUT "OUTLET HYDROCARBON FREE STREAM NUMBER="
        AT 6,50 PUT PARAM3 using "dd"
        AT 6,50 GET PARAM3 using "dd"
ENDFORM


SAVE TO "OFORM"
```

```
! UNITS.IPF
!
! UNIT PRINTOUT SUBROUTINE FOR USE WITH BERKIN
! USED FOR NORMAL DATA ENTRY TO STREAMIO.ITB
! SENDS THE FORM TO THE SCREEN

! LAST UPDATE: 04/13/86

TEST OPC

CASE 1:
        PUTFORM UFORM1
        TALLY UFORM1
        GETFORM UFORM1
        BREAK

CASE 2:
        PUTFORM UFORM2
        TALLY UFORM2
        GETFORM UFORM2
        BREAK
CASE 3:
        PUTFORM UFORM3
        TALLY UFORM3
        GETFORM UFORM3
        BREAK
CASE 4:
        PUTFORM UFORM4
        TALLY UFORM4
        GETFORM UFORM4
        BREAK
CASE 5:
        PUTFORM UFORM5
        TALLY UFORM5
        GETFORM UFORM5
        BREAK
CASE 6:
```

```
        PUTFORM UFORM6
        TALLY UFORM6
        GETFORM UFORM6
        BREAK
CASE 7:
        PUTFORM UFORM7
        TALLY UFORM7
        GETFORM UFORM7
        BREAK
CASE 8:
        PUTFORM UFORM8
        TALLY UFORM8
        GETFORM UFORM8
        BREAK
CASE 9:
        PUTFORM UFORM9
        TALLY UFORM9
        GETFORM UFORM9
        BREAK
CASE 10:
        PUTFORM UFORM10
        TALLY UFORM10
        GETFORM UFORM10
        BREAK
CASE 11:
        PUTFORM UFORM11
        TALLY UFORM11
        GETFORM UFORM11
        BREAK
CASE 12:
        PUTFORM UFORM12
        TALLY UFORM12
        GETFORM UFORM12
        BREAK
CASE 13:
        PUTFORM UFORM13
        TALLY UFORM13
```

```
        GETFORM UFORM13
        BREAK
CASE 14:
        PUTFORM UFORM14
        TALLY UFORM14
        GETFORM UFORM14
        BREAK
CASE 15:
        PUTFORM UFORM15
        TALLY UFORM15
        GETFORM UFORM15
        BREAK
CASE 16:
        PUTFORM UFORM16
        TALLY UFORM16
        GETFORM UFORM16
        BREAK
CASE 17:
        PUTFORM UFORM17
        TALLY UFORM17
        GETFORM UFORM17
        BREAK
CASE 18:
        PUTFORM UFORM18
        TALLY UFORM18
        GETFORM UFORM18
        BREAK
CASE 19:
        PUTFORM UFORM19
        TALLY UFORM19
        GETFORM UFORM19
        BREAK
CASE 20:
        PUTFORM UFORM20
        TALLY UFORM20
        GETFORM UFORM20
        BREAK
```

```
CASE 21:
        PUTFORM UFORM21
        TALLY UFORM21
        GETFORM UFORM21
        BREAK
CASE 22:
        PUTFORM UFORM22
        TALLY UFORM22
        GETFORM UFORM22
        BREAK
CASE 23:
        PUTFORM UFORM23
        TALLY UFORM23
        GETFORM UFORM23
        BREAK
CASE 24:
        PUTFORM UFORM24
        TALLY UFORM24
        GETFORM UFORM24
        BREAK
CASE 25:
        PUTFORM UFORM25
        TALLY UFORM25
        GETFORM UFORM25
        BREAK

OTHERWISE:
        OUTPUT "ERROR! ILLEGAL UNIT IGNORED"
        OUTPUT "STRIKE ANY KEY TO CONTINUE."
        WAIT
ENDTEST
```

```
! OPRINT.IPF

! ROUTINE TO SEND THE ENTIRE OPCODE MATRIX TO THE PRINTER
! THE OUTPUT IS IN THE SAME FORM AS THAT WHICH IS DISPLAYED
! ON THE SCREEN FOR DATA ENTRY.
! ALL ENTRIES ARE DISPLAYED FOUR TO A PAGE EXCEPT FOR
! THE ABSORBER/STRIPPER/DISTILLATION WHICH OCCUPIES AN ENTIRE PAGE
!
! LAST UPDATE: 03/28/86, DWN
!

E.PMAR=4
E.PWID=80

USE STREAMIO
OBTAIN FIRST
UCOUNT=0

WHILE UNIT<>99

        IF OPCODE=1 THEN
                EJECT
        ENDIF

        PRINT " "
        PRINT "UNIT=",UNIT

        TEST OPCODE

        CASE 1:
                PRINT UFORM1
                UCOUNT=-1
                EJECT
        BREAK

        CASE 2:
                PRINT UFORM2
```

```
BREAK

CASE 3:
        PRINT UFORM3
BREAK

CASE 4:
        PRINT UFORM4
BREAK

CASE 5:
        PRINT UFORM5
BREAK

CASE 6:
        PRINT UFORM6
BREAK

CASE 7:
        PRINT UFORM7
BREAK

CASE 8:
        PRINT UFORM8
BREAK

CASE 9:
        PRINT UFORM9
BREAK

CASE 10:
        PRINT UFORM10
BREAK

CASE 11:
        PRINT UFORM11
BREAK
```

```
CASE 12:
        PRINT UFORM12
BREAK

CASE 13:
        PRINT UFORM13
BREAK

CASE 14:
        PRINT UFORM14
BREAK

CASE 15:
        PRINT UFORM15
BREAK

CASE 16:
        PRINT UFORM16
BREAK

CASE 17:
        PRINT UFORM17
BREAK

CASE 18:
        PRINT UFORM18
BREAK

CASE 19:
        PRINT UFORM19
BREAK

CASE 20:
        PRINT UFORM20
BREAK
```

```
        CASE 21:
                PRINT UFORM21
        BREAK

        CASE 22:
                PRINT UFORM22
        BREAK

        CASE 23:
                PRINT UFORM23
        BREAK

        CASE 24:
                PRINT UFORM24
        BREAK

        CASE 25:
                PRINT UFORM25
        BREAK

        ENDTEST

PRINT " "
PRINT " "
PRINT " "

UCOUNT=UCOUNT+1

IF UCOUNT=4 THEN
        EJECT
        UCOUNT=0
ENDIF

OBTAIN NEXT

ENDWHILE
```

EJECT

RETURN

KMAN DATA BASE TABLE STRUCTURES FOR STREAMIO.ITB AND STREAMS.ITB

```
Table name    : STREAMIO
File name     : STREAMIO.ITB
Read  Access : A...............
Write Access : A...............
Creation Date      :  04/10/85
Modification Date :  04/02/86
Number of Records :          32

     ..........
Field : #MARK       LOGIC
Read  Access : A...............
Write Access : A...............
Picture :  (default)

     ..........
Field : OPCODE      NUM
Read  Access : A...............
Write Access : A...............
Picture : "dd"

     ..........
Field : UNIT        NUM
Read  Access : A...............
Write Access : A...............
Picture : "dd.dd"

     ..........
Field : PARAM1      NUM
Read  Access : A...............
Write Access : A...............
Picture : "ddddd.ddddd"

     ..........
Field : PARAM2      NUM
Read  Access : A...............
```

```
Write Access : A................
Picture : "ddddd.ddddd"

     ..........
Field : PARAM3      NUM
Read  Access : A................
Write Access : A................
Picture : "ddddd.ddddd"

     ..........
Field : PARAM4      NUM
Read  Access : A................
Write Access : A................
Picture : "ddddd.ddddd"

     ..........
Field : PARAM5      NUM
Read  Access : A................
Write Access : A................
Picture : "ddddd.ddddd"

     ..........
Field : PARAM6      NUM
Read  Access : A................
Write Access : A................
Picture : "ddddd.ddddd"

     ..........
Field : PARAM7      NUM
Read  Access : A................
Write Access : A................
Picture : "ddddd.ddddd"

     ..........
Field : PARAM8      NUM
Read  Access : A................
Write Access : A................
```

```
Picture : "ddddd.ddddd"

     ----------
Field : PARAM9      NUM
Read  Access : A...............
Write Access : A...............
Picture : "ddddd.ddddd"

     ----------
Field : PARAM10     NUM
Read  Access : A...............
Write Access : A...............
Picture : "ddddd.ddddd"

     ----------
Field : PARAM11     NUM
Read  Access : A...............
Write Access : A...............
Picture : "ddddd.ddddd"

     ----------
Field : PARAM12     NUM
Read  Access : A...............
Write Access : A...............
Picture : "ddddd.ddddd"

     ----------
Field : PARAM13     NUM
Read  Access : A...............
Write Access : A...............
Picture : "ddddd.ddddd"

     ----------
Field : PARAM14     NUM
Read  Access : A...............
Write Access : A...............
Picture : "ddddd.ddddd"
```

```
     ..........
Field : PARAM15     NUM
Read  Access : A...............
Write Access : A...............
Picture : "ddddd.ddddd"

     ..........
Field : PARAM16     NUM
Read  Access : A...............
Write Access : A...............
Picture : "ddddd.ddddd"

     ..........
Field : PARAM17     NUM
Read  Access : A...............
Write Access : A...............
Picture : "ddddd.ddddd"

     ..........
Field : PARAM18     NUM
Read  Access : A...............
Write Access : A...............
Picture : "ddddd.ddddd"

     ..........
Field : PARAM19     NUM
Read  Access : A...............
Write Access : A...............
Picture : "ddddd.ddddd"

     ..........
Field : PARAM20     NUM
Read  Access : A...............
Write Access : A...............
Picture : "ddddd.ddddd"
```

```
     ..........
Field : PARAM21     NUM
Read  Access : A...............
Write Access : A...............
Picture : "ddddd.ddddd"

     ..........
Field : PARAM22     NUM
Read  Access : A...............
Write Access : A...............
Picture : "ddddd.ddddd"

     ..........
Field : PARAM23     NUM
Read  Access : A...............
Write Access : A...............
Picture : "ddddd.ddddd"

     ..........
Field : PARAM24     NUM
Read  Access : A...............
Write Access : A...............
Picture : "ddddd.ddddd"

     ..........
Field : PARAM25     NUM
Read  Access : A...............
Write Access : A...............
Picture : "ddddd.ddddd"

     ..........
Field : PARAM26     NUM
Read  Access : A...............
Write Access : A...............
Picture : "ddddd.ddddd"

     ..........
```

```
Field : PARAM27     NUM
Read  Access : A...............
Write Access : A...............
Picture : "ddddd.ddddd"

     ..........
Field : PARAM28     NUM
Read  Access : A...............
Write Access : A...............
Picture : "ddddd.ddddd"

     ..........
```

```
Table name   : STREAMS
File name    : STREAMS.ITB
Read  Access : A...............
Write Access : A...............
Creation Date     :  04/10/85
Modification Date :  04/02/86
Number of Records :          54

     ..........
Field : #MARK       LOGIC
Read  Access : A...............
Write Access : A...............
Picture :  (default)

     ..........
Field : H2          NUM
Read  Access : A...............
Write Access : A...............
Picture : "d.ddddd"

     ..........
Field : N2          NUM
Read  Access : A...............
Write Access : A...............
Picture : "d.ddddd"

     ..........
Field : CO          NUM
Read  Access : A...............
Write Access : A...............
Picture : "d.ddddd"

     ..........
Field : CO2         NUM
Read  Access : A...............
Write Access : A...............
Picture : "d.ddddd"
```

```
     ..........
Field : C1          NUM
Read  Access : A...............
Write Access : A...............
Picture : "d.ddddd"

     ..........
Field : C2          NUM
Read  Access : A...............
Write Access : A...............
Picture : "d.ddddd"

     ..........
Field : C3          NUM
Read  Access : A...............
Write Access : A...............
Picture : "d.ddddd"

     ..........
Field : C4          NUM
Read  Access : A...............
Write Access : A...............
Picture : "d.ddddd"

     ..........
Field : C5          NUM
Read  Access : A...............
Write Access : A...............
Picture : "d.ddddd"

     ..........
Field : H2S         NUM
Read  Access : A...............
Write Access : A...............
Picture : "d.ddddd"
```

```
     ----------
Field : SO2          NUM
Read  Access : A................
Write Access : A................
Picture : "d.ddddd"

     ----------
Field : H2O          NUM
Read  Access : A................
Write Access : A................
Picture : "d.ddddd"

     ----------
Field : SOLV         NUM
Read  Access : A................
Write Access : A................
Picture : "d.ddddd"

     ----------
Field : DSULF        NUM
Read  Access : A................
Write Access : A................
Picture : "d.ddddd"

     ----------
Field : CH3SH        NUM
Read  Access : A................
Write Access : A................
Picture : "d.ddddd"

     ----------
Field : C2H5SH       NUM
Read  Access : A................
Write Access : A................
Picture : "d.ddddd"

     ----------
```

```
Field : SULF        NUM
Read  Access : A................
Write Access : A................
Picture : "d.ddddd"

     ..........
Field : WATER       NUM
Read  Access : A................
Write Access : A................
Picture : "d.ddddd"

     ..........
Field : O2          NUM
Read  Access : A................
Write Access : A................
Picture : "d.ddddd"

     ..........
Field : UNKN        NUM
Read  Access : A................
Write Access : A................
Picture : "d.ddddd"

     ..........
Field : FLOW        NUM
Read  Access : A................
Write Access : A................
Picture : "ddddd.ddddd"

     ..........
Field : TEMP        NUM
Read  Access : A................
Write Access : A................
Picture : "dddd.dd"

     ..........
Field : PRESS       NUM
```

```
Read  Access : A...............
Write Access : A...............
Picture : "ddddd.dd"

     ..........
Field : ENTH        NUM
Read  Access : A...............
Write Access : A...............
Picture : "dddddd.dd"

     ..........
Field : PHASE       NUM
Read  Access : A...............
Write Access : A...............
Picture : "d.d"

     ..........
```

This report was done with support from the Department of Energy. Any conclusions or opinions expressed in this report represent solely those of the author(s) and not necessarily those of The Regents of the University of California, the Lawrence Berkeley Laboratory or the Department of Energy.

Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Department of Energy to the exclusion of others that may be suitable.

LAWRENCE BERKELEY LABORATORY
TECHNICAL INFORMATION DEPARTMENT
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720