

AD-A192 885

CUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AND DATE SELECTED		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE 2 1988		4. MONITORING ORGANIZATION REPORT NUMBER(S)	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) C6D		5. MONITORING ORGANIZATION REPORT NUMBER(S)	
6a. NAME OF PERFORMING ORGANIZATION University of Texas at Austin Department of Mechanical	6b. OFFICE SYMBOL (if applicable)	7a. NAME OF MONITORING ORGANIZATION	
6c. ADDRESS (City, State, and ZIP Code) Engineering Austin, TX 78712-1063		7b. ADDRESS (City, State, and ZIP Code)	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION U.S. Army Medical Research & Development Command	8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DAMD17-84-C-4076	
8c. ADDRESS (City, State, and ZIP Code) Fort Detrick Frederick, Maryland 21701-5012		10. SOURCE OF FUNDING NUMBERS	
		PROGRAM ELEMENT NO. 62777A	PROJECT NO. 3E1 62777A878
		TASK NO. AF	WORK UNIT ACCESSION NO. 154
11. TITLE (Include Security Classification) (U) Computer Modeling and Optimization of OBOGS with Contaminants			
12. PERSONAL AUTHOR(S) Joseph J. Beaman			
13a. TYPE OF REPORT Annual/Final	13b. TIME COVERED FROM 7/15/84 TO 12/31/87	14. DATE OF REPORT (Year, Month, Day) 1988 February 15	15. PAGE COUNT 302
16. SUPPLEMENTARY NOTATION * Annual Report covers period 15 July 1986 - 31 December 1987 Final Report covers period 15 July 1984 - 31 December 1987			
17. COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	SUB-GROUP	
23	05		
20	05		
		OBOGS	
19. ABSTRACT: (Continue on reverse if necessary and identify by block number)			
The ultimate goal of this research is to insure proper design of molecular sieve oxygen generation systems for the US Army's in-flight, medivac, and field hospital use. Specifically the research involves further development of an OBOGS model to include the effects of contaminants in the feed air. This OBOGS model can be used to optimize and design OBOGS systems with respect to system parameters such as cycle time and bed and valve dimensions.			
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS		21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Mary Frances Bostian		22b. TELEPHONE (Include Area Code) 301/663-7325	22c. OFFICE SYMBOL SGRD-RMI-S

AD _____

**COMPUTER MODELING AND OPTIMIZATION
OF OBOGS WITH CONTAMINANTS**

ANNUAL/FINAL REPORT

JOSEPH J. BEAMAN

15 FEBRUARY 1988

Supported by

**U.S. ARMY MEDICAL RESEARCH AND DEVELOPMENT COMMAND
Fort Detrick, Frederick, Maryland 21701-5012**

Contract No. DAMD17-84-C-4076

**University of Texas at Austin
Department of Mechanical Engineering
Austin, TX 78712-1063**

Approved for public release; distribution unlimited

**The findings in this report are not to be construed as an official Department
of the Army position unless so designated by other authorized documents.**

Accession For	
NTIS CRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and for special
A-1	

Table of Contents

Executive Summary	1
Introduction.....	3
General Description of Pressure Swing Adsorption.....	3
Adsorbents.....	5
Motivation for Study of PSA Systems	8
Modeling Approaches	9
Bed Equations.....	10
Isotherms	11
Mass Transfer.....	18
Solution of Model Equations	18
Experimental Approach	21
Single Bed Apparatus	21
Dual Bed Apparatus	28
Future Work/Recommendations	32
Nomenclature.....	47
References	49
List of Contributors.....	51
Appendices	52
Appendix A: OBOGS Model Code, Temperature Correlated, for Oxygen/Nitrogen Feed	53
Appendix B: OBOGS Model Code, Three Component Feed.....	81
Appendix C: ISOBANK Code and Manual.....	114
Appendix D: On Board Oxygen Generation System (OBOGS) Simulator Manual	181
Distribution List.....	302

List of Figures

Figure 1: Simple two bed PSA system, with bypass	36
Figure 2: Zeolite A and X structures, left and right, respectively	37
Figure 3: Temperature dependence of oxygen K_A isotherm parameter, power law fit	38
Figure 4: Temperature dependence of nitrogen K_B isotherm parameter, power law fit	39
Figure 5: Linear fit of two parameter model to oxygen data	40
Figure 6: Linear fit of two parameter model to nitrogen data	41

Figure 7: Comparison of two parameter (GDM) and power law (WAL) K_A correlation for oxygen.....	42
Figure 8: Comparison of two parameter (GDM) and power law (WAL) K_B correlation for nitrogen.....	43
Figure 9: Single bed schematic	44
Figure 10: Single bed apparatus photographs	45
Figure 11: Dual bed apparatus schematic (Solenoid valves are numbered 1 to 4 from the top)	46
Figure 12: Dual bed apparatus photographs	47
Figure 13: Schematic of feed system for contaminant experiments	48

11-13

Executive Summary

A three component feed OBOGS model has been developed. This model runs on a DEC® Vax® computer and uses ReGIS® graphics output to convey information about the simulation to the user. The model uses a linear oxygen isotherm and Langmuir type isotherms for nitrogen and the contaminant; the isotherms are uncoupled. The model assumes plug flow, isothermal conditions, and uses a common mass transfer coefficient (linear driving force) for all components. Radial gradients and intraparticle gradients are ignored. The model is expected to work best for contaminants that are relatively small and non-polar, so that mass transfer resistance is small. A contaminant with a large mass transfer resistance will require at least a separate mass transfer coefficient. The assumption of a linear driving force with a constant coefficient may be entirely invalid for very large resistances. Also, the model will work best for relatively small beds. As the bed cross-section increases, the center of the bed becomes more adiabatic and heat transfer at the wall becomes important in determining the temperature profile (and, through the isotherm, the concentration profile). This effect is relatively unimportant in smaller beds because the amount of heat generated is small compared to the heat transfer area. The assumptions of negligible radial gradients and isothermal conditions prohibit the present model from considering the effects of heat transfer at the wall.

The temperature dependence of the oxygen and nitrogen isotherms have been determined between -55° and 120° C, and this information has been included in the binary feed OBOGS model. For interpolation, a three parameter power law relationship has proven superior for matching data than a more fundamental approach with two physically meaningful parameters. However, because these curves diverge outside the temperature range, the two parameter approach is recommended for extrapolation. The three parameter fit is used in the simulator.

A single bed PSA device has been constructed for determination of isotherm and mass transfer coefficients. A new, more general approach to determination of model parameters (including a better equilibrium description) has been developed. This approach matches experimental time domain breakthrough curves to those predicted by a combined equilibrium-flow-mass transfer set of equations. Matches are found by varying the parameters using a constrained optimization code. The system and method have been tested using a zeolite 4A derivative, but the results are not conclusive. Parameters found are of the proper magnitudes, but sensitivities to the objective function minimized must be addressed. This requires more data.

Data have been collected from literature and placed into an interactive program (ISOBANK) to generate curves for further investigation of the optimization problems. This program allows a user to generate isotherms and phase diagrams from a literature based data base.

A dual bed PSA system has also been constructed. This system is equipped with one axially tapped bed (to determine contaminant penetration into the bed) and has a variable timer to control cycle time. Redesign of supporting components has been necessary due to problems associated with previously completed similar experiments and also problems associated with contaminants of interest to the Army. The contaminants most likely to match model assumptions as well as Army interests may prove to be explosive mixtures, especially as the oxygen concentration is increased in the bed (ethane and methane are the contaminants). Therefore, special equipment is required, including explosion relief devices. In a typical 40 psig experiment, system pressures could become 1000 psi in an explosion.

Introduction

The pressure swing adsorption (PSA) process is an alternative to cryogenic distillation for the separation of gaseous mixtures. PSA is an important alternative because it is a less capital intensive process and can therefore be economically applied to the separation of smaller feed streams. Application of PSA to industrial processes is a quite recent development compared to more classic unit operations. Practical adsorbents were not available until the late 1950's (Milton, 1959), and a regenerative cycle was not available until the early 1960's (Skarstrom, 1960). However, even though PSA is a relatively new process, it has been used for many separations, including air, aromatics, natural gas, and air drying.

General Description of Pressure Swing Adsorption

Adsorptive processes separate gaseous mixtures based on their relative affinities for an adsorbent. More strongly adsorbed species are retained as they pass through a packed bed of adsorbent, thereby rendering the product enriched in the less strongly adsorbed (light) species. It is important to note the difference between adsorption and absorption.

Adsorption is a process that involves the transfer of molecules to the surface of an adsorbent; absorption involves the transfer of molecules through the surface that separates the adsorbent from the bulk stream.

As a gas mixture is passed through a bed of adsorbent, molecules of one (or more) of the components of the mixture become adsorbed on the bed. Eventually, the bed becomes saturated and must be regenerated. There are several options available for regeneration, and they lend their names to the different adsorption processes. In pressure swing adsorption, feed gas enters at a high pressure. Thus, the bed becomes saturated at the high pressure. Regeneration is accomplished by reducing the bed pressure as isothermally as possible. The reduction in pressure allows the adsorbed molecules to desorb, rendering the bed available for another adsorption step.

Other regeneration schemes are also available. Thermal swing adsorption (TSA) raises the bed temperature (which reduces the adsorptive capacities of adsorbents) at nearly constant pressure to clear the bed for another adsorption cycle. Inert gas stripping reduces the partial pressure of the adsorbed component in the bulk stream through the introduction of an inert (unadsorbed) gas in the regeneration step. The reduction in partial pressure causes desorption to occur. Displacement desorption clears the bed of the undesired adsorbate (an adsorbate is an adsorbed molecule) by displacing it with a more strongly adsorbed molecule. Finally, it is sometimes possible to combine these techniques for bed regeneration, depending on the particulars of a given separation problem.

As might be expected, PSA has both advantages and disadvantages when compared to the other regenerative possibilities. PSA has advantages over inert gas stripping and displacement desorption in that it does not require any supplemental gases. It has advantages over TSA because the faster cycle times available in PSA allow more cycles to be completed in a day, which corresponds to more product generated (due to the thermal inertia of most systems, it is faster to reduce bed pressure than to heat and cool a bed). The main disadvantage in PSA is the cost of gas compression. In fact, for large amounts of feed, it is this compression cost that makes PSA less economical than cryogenic distillation.

The most basic continuous PSA system is shown in Figure 1. Here a switching valve imposes supply pressure on Bed #1 and exhaust pressure on Bed #2. As the pressure in Bed #1 is higher, product is withdrawn from this bed. Bed #2, at low pressure, is ready for its next production cycle. In many of these systems, a portion of the product gas is throttled (through the bypass) to strip the lower pressure bed. Thus, the diagram shows a combined PSA/stripping system. This type of arrangement, while less efficient than simple PSA systems (i.e., it produces less oxygen per pound of oxygen fed), is capable of producing higher purity product. As Bed #1 becomes saturated, supply pressure is switched to Bed #2. For a transitory period, Bed #1 still has a higher pressure than Bed #2 and therefore has flow out of both ends (both to exhaust and product). During

this same period, Bed #2 has flow into both ends. As the pressure in Bed #2 rises (and that in Bed #1 falls), it eventually becomes the high pressure bed and produces product. At this time, Bed #1 is exhausted and stripped.

The system in Figure 1 is only capable of producing the light (less strongly adsorbed) species in high purity. The other components in the mixture are not as enriched because of the stripping process and also because the light species are still adsorbed (although not to the extent of the heavy species). Therefore, the light species are exhausted along with the heavies when bed pressure is reduced. Of course, a series of PSA devices could be assembled to separate a multicomponent mixture, but the success of such a system would largely depend on the available adsorbents.

Adsorbents

Adsorbent characteristics are crucial to the success of any adsorptive separation process. The adsorbent dictates both the possible separations as well as the regeneration cycles possible. Thus, in applying an adsorptive process to a separation problem, it is important to be aware of available adsorbents.

There are five commercially significant adsorbents: silica gel, activated alumina, activated carbon, carbon molecular sieves, and zeolites (Ruthven, 1984). These adsorbents generally rely on physical adsorption to retain adsorbates. Typically, physical adsorption has a low heat of adsorption and is non-specific, reversible, and non-activated (Ruthven, 1984). Thus, the interactive forces are typically weak, especially compared to chemisorption. These weak forces make the regeneration process simple and inexpensive. However, the non-specificity has historically made physical adsorption of little use for separations, especially in systems with similar molecules. The classic adsorbents adsorb a type of molecule; silica gel and activated alumina adsorb polar molecules, and have been used almost exclusively for removing water from process streams, while activated carbon (which is hydrophobic) adsorbs almost any organic molecule. These adsorbents cannot separate molecules with similar electronegative properties.

The development of adsorbents with very small (or even non-existent) pore distributions has made separations based on physical adsorption possible. These separations are based primarily on steric influences. Molecules that cannot fit into the pore structure of the adsorbent cannot interact with it. Zeolites and carbon molecular sieve are adsorbents capable of separating gas mixtures in this manner.

Carbon molecular sieves are produced through special activation and oxidation of carbon to yield a narrow pore size distribution (which is not characteristic of a normal activated carbon). Although reproducibility is a problem in manufacture, sieves with pore sizes from 4 to 9 Å can be produced (Ruthven, 1984). However, because a pore size distribution still exists, separations are usually better using zeolites. Nevertheless, carbon molecular sieves have found large scale application in generation of nitrogen from air.

Zeolites are fundamentally different from the other adsorbents discussed because they have no distribution in pore size. These adsorbents are formed from a network of SiO_4 and AlO_4 tetrahedra that share corner oxygens. The tetrahedra are assembled into a porous, regular crystalline structure with very well defined pores. The crystal structure allows for no variation in pore size. This makes highly specific separations possible.

The adsorptive properties of a zeolite depend on its structure. Although many zeolites occur naturally, their lattice structure is not as regular as synthetically produced zeolites. Furthermore, many structures are only available as synthetic zeolites. Different zeolite structures are obtained through different arrangements of the silica and aluminum based tetrahedra. The ratio of silica to aluminum is therefore one way to control structure during synthesis. Synthetically produced zeolites are classed as to structure type using letters. The numbers associated with zeolite designations typically give a relative measure of the size of the pores. Structures named in this way are A, X, and Y, with 4A, 5A, 10X, and 13X as specific examples. Yet not all zeolites are named this way (ZSM-5, for example).

Zeolite adsorptive properties are further affected by the choice of cation used to balance out the negative charge that each aluminum atom gives to the network. The cations

can partially occlude pore openings depending on their location in the lattice. For example, zeolite 4A has a smaller pore size than zeolite 5A (about 4 Å compared to 5 Å) because the Na^{1+} ions in 4A partially occlude pore openings, while the Ca^{2+} ions in 5A do not. Because the Ca^{2+} ion is divalent, less cations are needed to satisfy the electronegativity requirements of the lattice. In the A structure, these divalent cations occupy sites that leave the pores unobstructed. However, this may not always be the case. In the X structure, the 10X variant has smaller pores than 13X, even though 10X contains the divalent Ca^{2+} cation (and therefore less cations). In this structure, the Ca^{2+} ions occupy sites that occlude pores while the monovalent Na^{1+} ions do not, even though there are more Na^{1+} ions in the lattice. In conclusion, cation type and size affect the pore size in zeolites, but not necessarily in a straightforward way.

It is interesting to note that many important separations are not based on the steric or sieving properties of the zeolite, but on a specific interaction between the adsorbent and an adsorbate. For example, the air separation using zeolite 5A produces oxygen, even though oxygen is smaller than nitrogen and is expected to penetrate the pores more easily. This separation works because the quadrupole moment of the nitrogen molecule interacts with polar adsorbent sites. Because the interaction between the adsorbent and the nitrogen molecule is stronger than the adsorbent-oxygen interaction, oxygen is produced (Yang, 1987). This effect can be overcome, however, as the production of nitrogen from air using a specially treated 4A zeolite is possible. In this case, the higher mass transfer rate of oxygen to the zeolite offsets the greater equilibrium capacity of the zeolite for nitrogen.

Commercially available zeolitic adsorbents are generally in the form of pellets consisting of zeolite crystals in an inert binder. The binder usually accounts for about 20% of the pellet weight and is used to enhance the durability and effectiveness of the zeolite crystals. Although the exact composition of the binder is often proprietary, it has a negligible effect on the separative properties of the sieve.

Motivation for Study of PSA Systems

PSA systems are poorly understood, complex, and highly interactive. Design of PSA systems promises to be an important problem. PSA systems are currently used in industry to generate oxygen for increasing the efficiency of combustion processes. They are also used to generate oxygen for dissolution in sewage streams. Other uses can be expected to develop, provided reliable and efficient PSA systems can be designed and built economically.

Of course, of primary interest in this work is the application of PSA oxygen enrichment of air to military needs. These needs are basically to provide crew breathing air for aircraft (On Board Oxygen Generation System, or OBOGS) and to provide oxygen for field hospital use. The ability of a PSA process to fill these roles will free the military from the restrictions imposed by liquid oxygen and high pressure oxygen systems. However, the small size of these units make dynamic considerations quite important in the analysis and design of a suitable PSA unit. The current design methodology for aircraft units is empirical and often relies on trial and error. Larger industrial units usually use a steady state modeling approach with empirical correlations in design. Nevertheless, operational PSA units are available and are in use by the military. It is not known, however, how optimal these units are.

Thus, the motivation for study of PSA systems is the desire to efficiently design efficient PSA units.

Modeling Approaches

The PSA modeling problem is not a trivial one because of the interaction that exists between the system components. The result is a complex, dynamic system of linked, non-linear equations that are difficult to solve. While many mathematical descriptions and solutions exist of breakthrough curves on single beds with pure component feeds, many of the extant solutions are subject to unrealistic assumptions concerning column behavior. Therefore, their utility in an overall PSA system model is questionable.

Essential to any PSA modeling effort are five components: bed flow equations, interparticle mass transfer expressions, intraparticle mass transfer expressions, isotherm expressions, and heat transfer equations. The combined set of equations is formidable:

Fluid Mass Balance (one for each component)

$$-D_L \frac{\partial^2 C_i}{\partial z^2} + \frac{\partial}{\partial z} (u C_i) + \frac{\partial C_i}{\partial t} + \left(\frac{1 - \epsilon}{\epsilon} \right) \frac{\partial q_i}{\partial t} = 0 \quad (1)$$

Mass Transfer (one for each component)

$$\frac{\partial q_i}{\partial t} = f(T_s, C_i, C_i^*, C_{j \neq i}) \quad (2)$$

Equilibrium Isotherm (one for each component)

$$C_i^* = f(T_s, q_{j \neq i}, q_i) \quad (3)$$

Fluid Heat Balance (non-isothermal)

$$\begin{aligned} \frac{-\lambda_L}{C_f} \frac{\partial^2 T_f}{\partial z^2} + u \frac{\partial T_f}{\partial z} + \frac{\partial T_f}{\partial t} + \left(\frac{1 - \epsilon}{\epsilon} \right) \left(\frac{C_s}{C_f} \right) \frac{\partial T_s}{\partial t} = \\ \left(\frac{1 - \epsilon}{\epsilon} \right) \left(\frac{-\Delta H}{C_f} \right) \frac{\partial q}{\partial t} - \frac{4h_w}{\epsilon D_b C_f} (T_f - T_w) \end{aligned} \quad (4)$$

Particle Mass Balance (one for each component)

$$\frac{1}{R^2} \frac{\partial}{\partial R} \left(R^2 D_p \frac{\partial C_i}{\partial R} \right) = \frac{\partial C_i}{\partial t} + \left(\frac{1 - \alpha}{\alpha} \right) \frac{\partial q_i}{\partial t} \quad (5)$$

Particle Heat Balance

$$C_s \frac{\partial T_s}{\partial t} = \frac{3h}{R_p} (T_f - T_s) + (-\Delta H) \frac{\partial q}{\partial t} \quad (6)$$

Note that these equations already assume negligible pressure drop through the bed as well as negligible radial gradients.

The differences between modeling approaches are primarily in the assumptions made in order to simplify the above equation set so as to obtain a solution within some set of constraints. These differences are not minor, nor is any single set of assumptions expected to be valid for all separation problems. The model described below was designed to be applicable to air separation with a dilute contaminant in the feed stream. It was also designed to run reasonably quickly on a DEC® Vax® computer. Details are presented below.

Bed Equations

The equations in the model described in this report have been presented in Beaman (1985), Beaman, et al (1983), and Yang (1986). They are:

$$\frac{\partial u}{\partial z} = \beta \left(\frac{C^*}{C} - 1 \right) \quad (7)$$

$$\frac{\partial y_A}{\partial z} = \beta \left(C_A^* - \frac{y_A}{u} \right) \quad (8)$$

$$\frac{\partial y_B}{\partial z} = \beta \left(C_A^* - \frac{y_B}{u} \right) \quad (9)$$

$$\frac{\partial q}{\partial t} = -k(C^* - C) \quad (10)$$

$$\frac{\partial q_A}{\partial t} = -k(C_A^* - C_A) \quad (11)$$

$$\frac{\partial q_B}{\partial t} = -k(C_B^* - C_B) \quad (12)$$

$$C_A^* = K_A q_A \quad (13)$$

$$C_B^* = \frac{K_B q_B}{1 - \frac{q_B}{b_B}} \quad (14)$$

$$C_C^* = \frac{K_C q_C}{1 - \frac{q_C}{b_C}} \quad (15)$$

Note that this set of equations includes the following assumptions:

1. Isothermal behavior
2. No intraparticle gradients

3. Negligible radial gradients
4. Negligible axial dispersion
5. Negligible pressure drop through the bed
6. Common mass transfer coefficients
7. Constant mass transfer coefficient
8. Uncoupled mass transfer

Furthermore, the ideal gas law is assumed to be valid.

Isotherms

Equations (13)-(15) above are the isotherms currently used in the model. The oxygen isotherm (component A) is linear, while the contaminant (B) and nitrogen (C) isotherms are modified Langmuir type isotherms. The isotherms are very important in oxygen generation from air because the process is equilibrium dominated. Beaman coupled the isotherms for the binary feed case using the Ideal Adsorbed Solution Theory of Myers and Prausnitz (1965), but the isotherms for the ternary feed model are uncoupled. This has been done to reduce the computational time required. As long as the contaminant is only in low concentration, this is a reasonable assumption. For the binary oxygen-nitrogen case, the linking of the isotherms did not make a significant difference in the product stream composition calculated. Improvements in equilibrium are possible and will be discussed later.

Temperature Correlation for Isotherms in Model

Although the model assumes isothermicity, there is no reason that temperatures other than ambient cannot be modeled. Therefore, the model has been modified to include temperature effects. These changes have been in the ideal gas law conversions and in the isotherm expressions. Experimental results and a theoretical analysis follow. Because the contaminant is unknown, experiments were only done to quantify the temperature dependence of the oxygen and nitrogen isotherms.

Power Law

As determined by Beaman, et al (1983), one can analyze nitrogen breakthrough experimental results from a single bed packed with zeolite to identify model system parameters, specifically the isotherm coefficients K_A , K_B , and b_B .

During a nitrogen breakthrough (or wash-out) the single bed system is set up such that pure oxygen is flowing through the bed at a constant mass flowrate, ω_0 . At time $t=0$, a valve is switched to cause the inlet gas to change from pure oxygen to pure nitrogen with the inlet mass flowrate remaining constant. Typically, the bed outlet flowrate drops almost instantaneously to a lower value, ω . The outlet mass flow remains at this depressed level for a measurable period of time.

Using the results from Beaman, et al., the following procedure for determining the model parameters was developed.

For each test the following data were obtained: temperature, bed pressure, steady state mass flowrate, droop mass flowrate, and time to 50% breakthrough (t_{50}).

All tests run at one temperature were used to obtain a value for $\mu (=1-K_B/K_A)$ and $\gamma (=q_B/b_B)$ for that particular temperature.

To accomplish this an equation containing two unknowns (γ and μ) was obtained for each test. The equations were of the form:

$$\mu - Q^*\gamma = F, \quad (16)$$

where $Q=P(I)/P(1)$ ($P(I)$ is the bed pressure of that particular test and $P(1)$ is the lowest bed pressure of any test run at a particular temperature), and $F=M_r^*(\omega/\omega_0)$ (M_r is the ratio of the molecular weight of oxygen to that of nitrogen and ω/ω_0 is the ratio of the droop mass flowrate to the steady state mass flowrate).

Thus, a matrix solution was developed for the equation:

$$\underline{Z} = \underline{H} \underline{X} + \underline{V} \quad (17)$$

where $\underline{Z}$ is an $n \times 1$ vector containing each F (n is the number of tests run at this particular temperature), $\underline{H}$ is an $n \times 2$ matrix containing each coefficient for μ and γ , $\underline{X}$ is the 2×1

vector of μ and γ , and $\underline{Y}$ is an $n \times 1$ vector containing the noise about the average for each test equation.

Once μ and γ are obtained, K_A , K_B , and b_B may be found. Again, for a given temperature, K_A was obtained from the results of each test by using the following equation:

$$K_A = L \cdot A \cdot \rho \cdot (1 - \epsilon) / (\omega_0 \cdot t_{50} \cdot (1 - \mu)) \quad (18)$$

where $\rho = P(1) \cdot M_{O_2} / (R \cdot T)$, L = bed length, A = bed cross-sectional area, and R is the gas constant.

The K_A 's obtained for each test were then averaged to arrive at a K_A for that specific temperature.

K_B and b_B were then easily found:

$$K_B = (1 - \mu) / K_A \quad (19)$$

and

$$b_B = P / (R \cdot T \cdot K_A \cdot \gamma) \quad (20)$$

where P is the lowest pressure $P(1)$ which was used in determining a value for Q (the coefficient of γ) at each particular temperature.

Experiments were done from temperatures of -55°C to 120°C . A cryogenic altitude chamber was used for low temperature experiments; a bath filled with old (contaminated) zeolite was used for the high temperature experiments.

The bed had an inside diameter of 15/16 inches and an internal length of 6.0 inches. The bed was packed with 49.0 grams of Linde 5A molecular sieve sifted through a screen with 20 to 40 meshes per inch. This corresponded to zeolite particles 340 to 833 microns in diameter. The density of the zeolite was assumed to be 1.15 grams per cubic centimeter. Thus, the void fraction in the bed (ϵ) was approximately 0.37 (dimensionless).

Cylinders of pressurized oxygen and nitrogen were located upstream from the bed. Each cylinder exhausted through a standard reducing valve (regulator) and a solenoid valve to a tee-junction. The pressure upstream of the flow valve was maintained at 90 psia to cause choked flow throughout the length of the bed, regardless of the mass flowrate. The

flow valve was adjusted to maintain a constant mass flow through the bed. Thus, the steady-state mass flowrate was eliminated as a variable in this experiment.

The pressure drop across the bed was monitored by a differential pressure transducer which had one input from the line just before the entrance to the bed and the second input in the line just after the exit from the bed. The output from the pressure transducer was recorded using a strip chart recorder.

A rotameter was placed in line along with a hot wire anemometer. The rotameter was used to calibrate the output of the hot wire anemometer at 20°C and 1 atmosphere. The calibrated anemometer output was used to set the steady-state flowrate at all subsequent temperatures and pressures.

The line then exhausted into a plenum which could be regulated to vary the system pressure between 1/2 and 3 atmospheres while maintaining the steady-state flowrate. As a check on the validity of the anemometer at elevated pressures, another rotameter was placed at the exhaust of the plenum. The readings from this second rotameter verified that the readings of the anemometer were correct.

The bed pressure was monitored by placing an absolute pressure gauge just downstream from the bed.

Two thermocouples, one placed upstream from the bed and a second placed just downstream from the bed, showed the temperature of the gas flowing through the system. Tests were not run until the temperature readings of the two thermocouples became identical and remained stable for several minutes to obtain steady-state conditions.

Second order least square approximations for both K_A and K_B were obtained (T in degrees Celsius):

$$K_A = 7.2443 \times 10^{-6} T^2 + 1.8347 \times 10^{-3} T + 0.14233 \quad (21)$$

$$K_B = 1.321 \times 10^{-5} T^2 + 9.5775 \times 10^{-4} T + 0.036546$$

The correlations of the data to these approximations were satisfactory for use in the system model and are shown in Figures 3 and 4.

For mid-range temperatures b_B is relatively constant and this average is used in the system model:

$$b_B = 3.0591 \quad (22)$$

Theoretical Analysis

Beaman identified three parameters for the oxygen-nitrogen system, as well as a method for evaluating these parameters. In Beaman's notation, the component isotherms are:

$$C_A^* = K_A q_A \quad (23)$$

$$C_B^* = \frac{K_B q_B}{\left(1 - \frac{q_B}{b_B}\right)} \quad (24)$$

K_A = oxygen equilibrium constant $\left[\frac{\text{kg-mol gas phase}}{\text{kg-mol adsorbed}}\right]$

K_B = nitrogen equilibrium constant $\left[\frac{\text{kg-mol gas phase}}{\text{kg-mol adsorbed}}\right]$

C_A^* = interfacial oxygen concentration $\left[\frac{\text{kg-mol}}{\text{m}^3}\right]$

C_B^* = interfacial nitrogen concentration $\left[\frac{\text{kg-mol}}{\text{m}^3}\right]$

q_A = adsorbed phase oxygen concentration $\left[\frac{\text{kg-mol adsorbed}}{\text{m}^3}\right]$

q_B = adsorbed phase nitrogen concentration $\left[\frac{\text{kg-mol adsorbed}}{\text{m}^3}\right]$

b_B = limiting nitrogen adsorption $\left[\frac{\text{kg-mol}}{\text{m}^3}\right]$

Note that the units for b_B are incorrect in the 1985 paper (where $b_B = \beta$).

Predicting the temperature dependence of these parameters is not trivial because these expressions, while similar to the classic Langmuir expressions, are significantly different. Specifically, the bed density of the adsorbent is imbedded in these parameters and, because the parameters are in terms of concentrations, an equation of state must be assumed. This

equation of state causes the temperature dependence of the parameters to be different from those of the Langmuir expression.

The classic Langmuir expression is:

$$\theta = \frac{m}{m_s} = \frac{kp}{1 + kp} \quad (25)$$

θ = fraction of surface coverage [dimensionless]

m = amount adsorbed $\left[\frac{\text{kg-mol adsorbed}}{\text{gm adsorbent}} \right]$

m_s = amount adsorbed at saturation $\left[\frac{\text{kg-mol adsorbed}}{\text{gm adsorbent}} \right]$

k = equilibrium constant $\left[\frac{1}{\text{Pa}} \right]$

p = partial pressure of component $\left[\frac{1}{\text{Pa}} \right]$

The temperature dependence of the equilibrium constant k follows the van't Hoff relation:

$$k = k_0 \exp \left(\frac{-\Delta H_0}{RT} \right) \quad (26)$$

where R is the universal gas constant [8.3144 J/gm-mol K], $(-\Delta H_0)$ is the isosteric heat of adsorption [J/gm-mol], and T is the temperature in degrees K.

From the ideal gas law, we can find:

$$C_i^* = \frac{p_i}{RT} \quad (27)$$

Substitution of (27) into (25) and rearranging:

$$m = \frac{km_s C_i^* RT}{1 + k C_i^* RT} \quad (28)$$

Solving for C_i^* ,

$$C_i^* = \frac{\left(\frac{1}{km_s RT} \right) mp}{\rho - \frac{m \rho}{m_s}} \quad (29)$$

Where ρ is the bed density of adsorbent [gm/m³].

In Beaman's notation, $q_i = mp$. Substitution into (29) and rearrangement yields:

$$C_i^* = \frac{\left(\frac{1}{k m_s R T \rho}\right)^{q_i}}{1 - \frac{q_i}{m_s \rho}} \quad (30)$$

Comparison to (24) gives the desired result:

$$K_i = \frac{1}{k_0 \exp\left(\frac{-\Delta H_0}{RT}\right) m_s R T \rho} \quad (31)$$

$$b_B = m_s \rho$$

Therefore, we find equations (31) give the expected theoretical dependence of K_i on temperature as well as show that b_B is expected to be independent of temperature. The experiments confirm this latter observation. However, the power law in positive powers of temperature (in °C) fit to K_i using three parameters is not correct from a theoretical standpoint. A more valid approach uses the expression for K_i in (31) to obtain a two parameter fit to the data. We can rewrite the expression for K_i as:

$$K_i = \frac{1}{A T \exp\left(\frac{-B}{T}\right)} \quad (32)$$

Where $A = k_0 m_s R \rho$ and $B = (\Delta H_0/R)$. Taking the natural logarithm of both sides of (32) gives:

$$\ln(K_i) = -\ln(AT) + \frac{B}{T} = -\ln(A) - \ln(T) + \frac{B}{T} \quad (33)$$

Rearranging,

$$\ln(K_i) + \ln(T) = -\ln(A) + \frac{B}{T} \quad (34)$$

or

$$\ln(K_i T) = -\ln(A) + \frac{B}{T} \quad (35)$$

Thus, if $\ln(K_i T)$ is plotted against $(1/T)$, a line should be obtained. This is shown in Figure 5 for oxygen and in Figure 6 for nitrogen. The coefficients from these figures were used to generate the GDM curves in Figures 7 and 8. From these graphs, it is clear that the three parameter curve is slightly superior for interpolation. However, the figures show that

the GDM and WAL curves diverge outside the temperature range of the data. Because of its theoretical background, extrapolation should be more reliable with the GDM curve.

Mass Transfer

Presently, the model uses a common mass transfer coefficient for all components. This is only reasonable for contaminants with sizes that are similar to oxygen and nitrogen. Furthermore, the use of the linear concentration driving force with a constant coefficient is not considered reasonable for zeolites (Yang, 1987, Ruthven, 1984). However, both of these assumptions have worked well in the model. Both assumptions have worked because air separation on zeolite 5A is basically an equilibrium process. Consequently, diffusional resistances are relatively unimportant. Beaman (1983,1985) found that the binary nitrogen-oxygen system was very insensitive to the choice of mass transfer coefficient and had some difficulty in determining one uniquely. Therefore, the mass transfer assumptions in the model are acceptable, as long as the contaminants are not large molecules. For large molecules, diffusional resistance will be significant, and the model may fail.

Solution of Model Equations

The system equations are solved as described in Beaman (1985). Note, however, that the solution algorithm supplied in that paper inherently assumes the bed geometry outlined in the paper, where a cross-sectional area change occurs at one half the total bed length. The equations needed for the valves and the plenum are described below.

Valve Equations

The basic equations for the flow of a gas through standard compressible flow pneumatic orifices have the following two forms (Blackburn, et al, 1960):

1. $W = C_d \cdot A_d \cdot C_1 \cdot P_u / (T^{0.5}) \cdot (P_r^{1/k}) \cdot ((1 - P_r^{(1 - (1/k))})^{0.5})$ for pressure ratios $(P_r = P_d/P_u)$ greater than the choked flow ratio (0.528) and less than unity.

2. $W = C_d * A_d * C_2 * P_u / (T^{0.5})$ for pressure ratios less than the choked flow ratio.

W = mass flowrate through the valve

A_d = orifice area

C_d = dimensionless discharge coefficient

T = upstream stagnation temperature

P_u = upstream stagnation pressure

P_d = downstream pressure

k = ratio of specific heats $\left(\frac{C_p}{C_v}\right)$

C_1 and C_2 are functions of the gas constant and the ratio of the specific heats of the gas.

Plenum Equation

The plenum was modeled as a simple two stage isobaric mixer. As defined by Beaman, the transfer function between the inlet (x_i) and outlet (x_o) mole fractions is given by:

$$\frac{x_o}{x_i} = \frac{1}{(\tau_s + 1)^2} \quad (36)$$

where

$$\tau = \frac{C_p V_{BR} M_{AV}}{W_{BR}} \quad (37)$$

C_p is the plenum concentration (assumed constant), W_{BR} is approximately half the total plenum volume ("approximately" because of corrections for mixing in piping), M_{AV} is an average molecular weight in the plenum (due to the similarity of the molecular weights of oxygen and nitrogen, the fact that M_{AV} is related to x_o and x_i is ignored), and W_{BR} is the breathing flowrate, in mass units.

Discussion

As shown in Beaman's (1985) paper, the above model works well for the binary system (which uses a subset of the bed equations above). Even for high product flowrates,

the model matched experiment for the Bendix prototype unit tested. Thus, we have confidence that the model is a good approximation of the operation of that unit. The multicomponent model did not exist at that time, however, and no tests with contaminants in the feed were attempted.

It should be noted that several equations and definitions in the 1985 paper are incorrect. Using the equation numbers and notation from that paper, equations (6b), (11), and the defining equations for u and γ_A in section 3.2 (iii) should read:

$$\gamma_B^0 = \frac{K_B n_B^0}{\left(1 - \frac{n_B^0}{b}\right)} \quad (6b)$$

$$u_i = u_{i-1} + \beta \left(\frac{\gamma_i + \gamma_{i-1}}{2C} - 1 \right) \Delta z \quad (11)$$

and

$$u = r_0 + r_1(z - z_{i-1}), \quad r_0 = u_{i-1}, \quad r_1 = \frac{u_i - u_{i-1}}{\Delta z}$$

$$\gamma_A = r_2 + r_3(z - z_{i-1}), \quad r_2 = \gamma_{A,i-1}, \quad r_3 = \frac{\gamma_{A,i} - \gamma_{A,i-1}}{\Delta z}$$

The units on b in that paper should be [kg mole adsorbed/m³].

It is very important to integrate equations (8) and (9) (in this report) in the direction of flow. This is required because in deriving the solutions to (8) and (9) using the approximations immediately above, r_1 must be positive whenever r_0 is zero (at the zero velocity point in the bed). This is necessary to avoid division by zero in the analytical approximations to (8) and (9).

Experimental Approach

Responsibility for acquisition of experimental information for model verification was officially transferred to the University of Texas at the end of July, 1986. Although a mass spectrometer was in place at this time, it was not operational until the fall of 1986, when other responsibilities substantially interfered with experiments. Nevertheless, single and dual bed systems have been constructed, and some preliminary work has been completed.

Single Bed Apparatus

The single bed unit was constructed to allow determination of mass transfer and isotherm parameters. Because Beaman's approach relied on the specifics of the isotherm and flow equations outlined in his papers, a new, more general approach was desired for determining multicomponent isotherms. The new parameter estimation scheme was designed to yield isotherm parameters for a new isotherm type, the Vacancy Solution Model (Suwanayuen and Danner, 1980a, 1980b, Cochran, et al, 1985a, 1985b), and also axial dispersion and individual mass transfer coefficients. All of this information is derived from breakthrough experiments.

Construction/Description

The single bed system is shown schematically in Figure 9. Figure 10 is a photograph of the system. The bed is two feet long (61 cm), one inch O.D. (2.54 cm), with .035 inch (.089 cm) wall thickness. Construction is of 316 stainless steel. The bed is housed in a temperature controlled box that has an available range of room temperature to 70° C. The stainless steel tubing is packed with zeolite adsorbent. Feed gas flow rates are controlled and measured by mass flow controllers (SIERRA 840 series) and monitored continuously on an IBM PC-XT using Labmaster and DADIO boards (Scientific Solutions Inc.). The concentration from the exit of bed is analyzed by a mass selective detector (Hewlett Packard 5790 series). The inlet and outlet pressures are controlled by in-line regulators and measured by pressure transducers (Omega), and monitored continuously. The IBM PC-

XT is also used to drive the solenoid switching valve. The task of this experimental equipment is to give the response curve for a step change in feed gas concentration.

Parameter Estimation Scheme

The analysis of the breakthrough curves involves matching the experimental breakthrough curves to curves predicted by a mathematical model by varying the parameters in the model. This approach to determining the isotherm parameters is a chromatographic method, as opposed to the more classic gravimetric and volumetric methods. In determining multicomponent isotherms, both gravimetric and volumetric methods suffer from the fact that samples must be withdrawn from the system in order to determine gas composition. This is not necessary for chromatographic methods (although it may be necessary to add tracers).

Background

A chromatographic method of measuring parameters of a packed bed of porous adsorbents has advantages of speed and simplicity. The chromatographic method depends on matching the experimental chromatogram to the theoretical response curve predicted from an appropriate dynamic model.

The first and most widely used method is the method of moments of Kubin (1965a, b) and Kucera (1965) which was developed by Schneider and Smith (1968a, b). Ruthven and Kumar (1980) used the modified method of moments for study of a binary adsorption system. Previous studies based on this approach, have generally relied entirely on the first and second moments but this means that important information contained in the shape of the curve is ignored.

To overcome these difficulties, Sarma and Haynes (1974) have used an alternative method of analysis involving direct matching of the experimental and theoretical curves in the frequency domain. The method gives reliable results but it has the disadvantages of mathematical complexity since numerical conversion between the time and frequency domains is required.

Another approach based on numerical solution of the governing differential equations has been developed by Raghavan and Ruthven (1985), but this is also mathematically complicated. Hassan et al.(1985) have used this approach for a single component adsorption system with nonlinear equilibrium.

Here we present a curve fitting method to estimate the parameter values for a packed bed with the Vacancy Solution Model (Cochran, Kabel, and Danner, 1985).

Mathematical Model

To represent the dynamic response of the adsorption system a linear rate model is used with the familiar Danckwerts boundary conditions. Parameters included in the model are therefore (a) the equilibrium adsorption constants for linear adsorption model or (b) the parameters for Vacancy Solution Model, and the axial dispersion coefficients and the mass transfer rate coefficients.

Assumptions:

- a) The system is isothermal.
- b) The fluid velocity within the bed is constant.
- c) The flow pattern is described by the axially dispersed plug flow model.
- d) The mass transfer rate is represented by a linear driving force rate expression.

Mass balance for adsorbable component in external fluid phase

$$-D_L \frac{\partial^2 C}{\partial z^2} + u \frac{\partial C}{\partial z} + \frac{\partial C}{\partial t} + \left(\frac{1 - \epsilon}{\epsilon} \right) \frac{\partial q}{\partial t} = 0 \quad (38)$$

Mass transfer rate expression

$$\frac{\partial q}{\partial t} = k(q^* - q) \quad (39)$$

Boundary conditions

$$D_L \left(\frac{\partial C}{\partial z} \right)_{z=0} = -u(C_{z=0-} - C_{z=0+}) \quad (40)$$

$$D_L \left(\frac{\partial C}{\partial z} \right)_{z=L} = 0$$

Initial conditions

$$C(z,0) = q(z,0) = 0 \quad (41)$$

Two equilibrium models are considered:

a) Linear adsorption model

$$q^* = K_A C \quad (42)$$

b) Vacancy solution model (VSM) (Cochran, Kabel, and Danner, 1985)

$$P = \frac{n_1^\infty}{b_1} \frac{\theta}{1 - \theta} \exp\left(\frac{(\alpha_{1v})^2 \theta}{1 + \alpha_{1v} \theta}\right) \quad (43)$$

$$\theta = \frac{n_1}{n_1^\infty}$$

The mathematical model is non-dimensionalized for both the linear isotherm and VSM cases. First, for the linear case, the following dimensionless variables are defined:

$$Pe = \frac{uL}{D_L}, \quad T_f = \frac{kL}{u}, \quad \tau = \frac{tu}{L}, \quad C = \frac{C}{K_A C_0}, \quad q = \frac{q}{q_0}, \quad x = \frac{L}{z}$$

These variables give the following PDE:

$$-\frac{1}{Pe} \frac{\partial^2 C}{\partial x^2} + \frac{\partial C}{\partial x} + \frac{\partial C}{\partial \tau} + \left(\frac{1 - \epsilon}{\epsilon}\right) K_A \frac{\partial q}{\partial \tau} = 0 \quad (44)$$

$$\frac{\partial q}{\partial \tau} = T_f (q^* - q) \quad (45)$$

For the VSM, the following dimensionless variables and equations are obtained:

$$Pe = \frac{uL}{D_L}, \quad T_f = \frac{kL}{u}, \quad \tau = \frac{tu}{L}, \quad C = \frac{C}{C_0}, \quad q = \frac{q}{q_0}, \quad x = \frac{L}{z},$$

$$n_1^\infty = \frac{n_1^\infty}{q_0}, \quad b_1 = \frac{b_1 P_0}{q_0}, \quad n_1 = \frac{n_1}{q_0}, \quad q^* = \frac{q^*}{q_0}, \quad \theta = \frac{n_1}{n_1^\infty}$$

$$-\frac{1}{Pe} \frac{\partial^2 C}{\partial x^2} + \frac{\partial C}{\partial x} + \frac{\partial C}{\partial \tau} + \left(\frac{1 - \epsilon}{\epsilon}\right) K_A \frac{\partial q}{\partial \tau} = 0 \quad (46)$$

$$\frac{\partial q}{\partial \tau} = T_f (q^* - q) \quad (47)$$

$$C = \frac{n_1^\infty}{b_1} \frac{\theta}{1 - \theta} \exp\left(\frac{(\alpha_{1v})^2 q}{1 + \alpha_{1v} \theta}\right) \quad (48)$$

The equations were first reduced to ordinary differential equations by the method of orthogonal collocation as described by Villadsen and Stewart (1967), Karanth and Hughes (1974) and Finlayson (1980). The ordinary differential equations were then solved by using Gear's integration algorithm with full jacobian analysis.

Estimation

According to the present model the response curve $C(L,t)$ is given by the solution of the coupled partial differential equations with the boundary and initial conditions. From the numerical solution to these equations one may generate a family of adsorption and desorption curves with parameters. Alternatively the values of the parameters corresponding to the experimental conditions may be found by matching the experimental and theoretical curves. Because of the sensitivity of the regressed parameters to the experimental breakthroughs both adsorption models are used; the Henry's law constant, b_1 , is obtained using the linear adsorption model, while the VSM is used to determine the limiting amount of adsorption, n_1^∞ , and the gas-solid interaction term, α_{1v} .

For the linear adsorption model (with inlet gas concentration less than 1%), the adsorption equilibrium constant can be easily calculated by integrating the breakthrough curves. q_0/C_0 may be calculated from the known gas phase concentration and the total equilibrium capacity, as determined by integration of the breakthrough curve:

$$\frac{q_0}{C_0} = \left(\frac{\epsilon}{1 - \epsilon}\right) \left(\frac{u}{L} \left(t - \int_0^t C dt \right) - 1 \right) \quad (49)$$

When the concentration of feed gas is less than 1%, q_0/C_0 is the same as K_A .

The Peclet number can be calculated from the correlation of Edwards and Richardson (1968):

$$D_L = 0.703 D_m + \frac{R_p u}{1 + \frac{4.85}{R_p} D_m} \quad (50)$$

Here only one parameter, the dimensionless mass transfer coefficient T_f , remains to be determined by matching the theoretical and experimental breakthrough curves. T_f can be estimated by using the GRG2 optimization code (Lasdon, 1986).

For higher inlet gas concentrations, the VSM is used. The Peclet number, Pe , is still calculated by using the correlation of Edwards and Richardson (1968) and the mass transfer coefficient, T_f , can be estimated with the linear model, assuming k_f is independent of gas concentration. K_A is known from the linear experiment as well.

Henry's law constant, b_1 , is obtained:

$$b_1 = \frac{K_A}{\left(\frac{q_0}{C_0}\right)} \quad (51)$$

The ratio of q_0/C_0 is determined by integrating the effluent curve.

The limiting amount of adsorption, n_1^∞ , and parameter describing nonlinearity, α_{1v} , are estimated using the GRG2 optimization code with objective function:

$$\text{Min}_{n_1^\infty, \alpha_{1v}} \sum_{i=1}^{N_p} (C_{i, \text{cal}} - C_{i, \text{exp}})^2 \quad (52)$$

subject to constraint:

$$1 = \frac{n_1^\infty}{b_1} \frac{1}{n_1^\infty - 1} \exp\left(\frac{(\alpha_{1v})^2}{n_1^\infty + \alpha_{1v}}\right)$$

This optimization problem involves one degree of freedom.

Interactive programs using IMSL, Tektronix graphics libraries and the GRG2 code were developed for the parameter estimation and the handling of actual experimental data.

ISOBANK

Concurrent with the construction of the single bed apparatus and development of the estimation scheme was the creation of ISOBANK. This is an interactive program that has a collection of isotherm information from the literature imbedded in it. Users are able to use the program to generate isotherms and phase diagrams from available literature data. This program was used to generate data on which to test the above estimation scheme. The program, a user's manual, and example output from the program are included in an appendix to this report.

Discussion

The single bed apparatus and estimation scheme have been tested using a zeolite called RS-10. This is a 4A zeolite that has been specially treated to further restrict the pore openings. Parameters were obtained that are of the same order of magnitude as literature parameters for 4A, but no information is available on RS-10 in the literature. Therefore, the validity of the approach has not yet been proven.

Numerical experiments carried out on literature data in ISOBANK indicate that the results are quite sensitive to the choice of objective function. A least squares objective function is described above. From a statistical point of view, better options for objective functions exist, particularly if more information is known about a process. One of these alternative objective functions is known as the error-in-variables method (EVM) (High and Danaer, 1986, Reilly and Patino-Leal, 1981). This objective function uses covariance matrices to account for measurement errors that may influence results in subtle ways. The covariance matrices for various types of measurements have been assembled and are available in the literature (Bilgic, 1970). The EVM seems to match the literature isotherms better, but this conclusion is somewhat simplistic and is simply based on "eyeballing" the generated isotherms. Least squares will generate the best fit possible in the least squares sense. The EVM uses covariance matrices in an objective function to get the best fit possible in a different sense. Thus, they are both best fits, and any conclusion as to which

fit is better must be based on a more fundamental analysis of the objective functions (especially on error propagation in the least squares method).

Until the single bed experiments are done for a more known zeolite (such as 5A), and until the parameter estimation problems are resolved, it will not be possible to properly qualify the equipment and method.

Dual Bed Apparatus

The dual bed PSA unit was constructed to verify the three component feed OBOGS model. Feed mixtures are fed to the unit; outlet composition and axial penetration of the contaminant are measured.

Construction/Description

Figure 11 shows the schematic of the two bed unit. Figure 12 is a photograph of the same unit. The overall unit is based on the design of the Small Oxygen Concentrator (SOC) of Theis, Ikels, and Dornes (1985). The two bed system is constructed from 316 stainless steel. The total bed length is 20 inches (50.8 cm) with a packed length of about 18 inches (45.7 cm). Wall thickness is .065 inches (0.165 cm). One bed has taps every 4 inches (10.2 cm) welded onto the bed. The taps are threaded for 1/8 inch NPT and accept standard Swagelok® thermocouple fittings. Stainless tubing of 1/16 inch O.D. is used in place of thermocouples in these taps. The sample end of the tubing has very fine screen welded over it to prevent plugging of the probe by zeolite dust. The probes are joined to the main 1/16 inch O.D. sample line using a union. The main sample line leads to the mass spectrometer (HP 5970 Series MSD). ASCO solenoid valves are used to control supply and exhaust pressures on the beds. These valves have 7/32 inch orifices and are recommended for use with up to 40 psi pressure differences across the valve. The valves are controlled by a variable timer such that valves numbers 1 and 3 are open/closed at the same time, and valves 2 and 4 are closed/open at the same time. One-way check valves with 1/3 psi opening pressure are used to prevent backflow of product into the beds (in excess of the bypass). The bypass leg is a section of 316 stainless microbore tubing 12

inches (30.5 cm) long, 1/8 inch O.D., and 0.6 mm I.D. All other tubing is 1/4 inch O.D. x 0.028 inch wall 316 stainless steel. Product rate is controlled with a metering valve and monitored with a rotameter.

Qualification Experiments

Before the two bed system was qualified, problems were identified in two main areas. The first problem concerned the quality of the supply to the PSA unit. In previous studies using the Brooks AFB SOC, results obtained for steady state product composition did not match model predictions (binary feed model). Initially, it seemed that the model was inadequate to predict performance of this unit. However, in view of the model's success in modeling the Bendix unit, this conclusion appeared suspect. Further analysis indicates that the SOC is quite sensitive to fluctuations in inlet pressure. This is (in part) because of the very short cycle time of this unit (the supply pressure step of each cycle is only 4 seconds in oxygen generation with zeolite 5A). At 10 to 40 psig (common experimental conditions), the impedance of the supply line to the SOC was large enough to cause a significant variation in inlet pressure during the supply part of the cycle. It is believed that the changing supply pressure caused the poor model match to the data (the model assumes constant supply pressure during a supply step). Thus, the new system had to have a supply plenum to minimize this effect.

The second problem encountered was the nature of the contaminants of interest to the Army.

Contaminants Considered

Table 1 is a table of contaminants of interest to the Army. Carbon dioxide is the only contaminant that is non-toxic, non-flammable, and gaseous at the conditions of PSA operation. The condensible substances need to be examined at high temperature, and, in practice, should be removed with filters before the PSA unit. Because carbon monoxide has the same molecular weight as nitrogen, the mass spectrometer will not be able to distinguish nitrogen and carbon monoxide. Toxic, corrosive substances are unacceptable

safety risks because of their ability to corrode seals and leak. Exhaust of these substances is also a problem. Finally, ozone is unstable, and ammonia is too small and polar to be removed from the bed in an exhaust cycle. Thus, it was decided to work on carbon dioxide, methane, and ethane. The flammability hazard is manageable through proper design, construction, and operation of the system.

Table 1
Contaminant List

<u>Contaminant</u>	<u>Attributes</u>
Ammonia	Small, Polar, Irreversible (?)
Carbon Dioxide	Non-Toxic, Non-flammable
Carbon Monoxide	Same Molecular Weight as Nitrogen
Methanol	Condensible (Liquid)
Ethanol	Condensible (Liquid)
Aviation Fuel	Condensible (Liquid)
Methyl Bromide	Highly Toxic, Shipped as Liquid at 13 psig
N_2O_4 / $2NO_2$	Toxic, Corrosive
Ozone (O_3)	Unstable
Freons	Condensible
Methane	Flammable
Ethane	Flammable

Design Concerns

The design of the ancillary components to safely and effectively run the two bed PSA unit must consider possible explosion hazards and the supply requirements of the unit. The only suitable plenum available is a T-type gas cylinder. These cylinders are rated to 3000 psi. In order to form a flow-through plenum, a hole is drilled in the center bottom and

tapped to 3/4 inch NPT for the gas inlet-explosion relief assembly. Adiabatic flame calculations indicate that explosion of a methane (or ethane) with air mixture at experimental conditions could generate pressures of up to 1000 psi in the plenum. Thus, explosion relief disks with a burst pressure of 150 psi have been specified for inlet and outlet sides of the plenum. One way check valves rated to 1000 psi have been specified upstream and downstream of the plenum in an attempt to minimize any overpressures to other system components. Explosion-proof solenoids have also been specified to minimize the number of potential ignition sources. A schematic of the system is shown as Figure 13. Because the plenum is so much larger than the bed, inlet pressure variation is expected to be minimal.

Experimental Plan

Carbon dioxide should be run first to test the gas mixing system. If it is not possible to tightly control composition, it will be necessary to order custom mixtures of the methane (ethane) in air. Feeds in the explosive mixture region of composition will not be fed to the PSA unit. However, as the gas in the bed becomes enriched in oxygen, the mixture may become explosive if the contaminant (methane or ethane) is not retained by the bed.

Operation with a binary feed will be investigated first. The volumes on this unit are slightly larger than the Brooks SOC, so a longer cycle time may be merited. The contaminated feeds will be run the the unit at nominal conditions determined by the binary feed experiments.

Discussion

The two bed apparatus is constructed and only awaits completion of the supply network. The supply network has been specified and parts are under construction. Contaminant properties have hindered progress to this point.

Future Work/Recommendations

1. The verification work for the contaminant model needs to be completed. However, even without these data, it is possible to make the following recommendations regarding the model:
 - a. The model is best used on smaller diameter beds, in order to avoid problems with radial effects and wall heat transfer.
 - b. The model will be most reliable in modeling non-polar contaminants similar in size to oxygen and nitrogen.
 - c. Inlet contaminant concentrations greater than 5% may be too high for the model to give reliable predictions.
2. The parameter estimation problems must be addressed. Improvement of the model to include the Vacancy Solution Model for the equilibrium description cannot occur until the problems with estimating the parameters of this model are resolved. This may involve a detailed analysis of the effects of different objective functions in the optimization routine.
3. For larger units, the effects of radial gradients need to be studied. In particular, it is important to know the largest bed cross-section that can be reliably designed ignoring radial gradients. Industrial sources indicate that radial temperature gradients can be measured in beds with diameters as small as 4 inches (in air separation). However, the diameter where compositional gradients can exist is expected to be somewhat greater.
4. The Vacancy Solution Model needs to be implemented. This equilibrium model is better at predicting multicomponent equilibria than the simple uncoupled approach that is currently being used. While this is not particularly important for low concentration streams, it is critical for more concentrated streams.

5. The linear driving force approximation needs to be critically re-evaluated in the light of literature criticisms as to its applicability to zeolites. Utilization of more detailed mass transfer mechanisms will involve the introduction of more parameters; each new parameter will be subject to some uncertainty. It is not clear if the adoption of a more realistic mass transfer mechanism will result in greater overall model accuracy unless the uncertainties in the introduced parameters can be minimized.

Figures

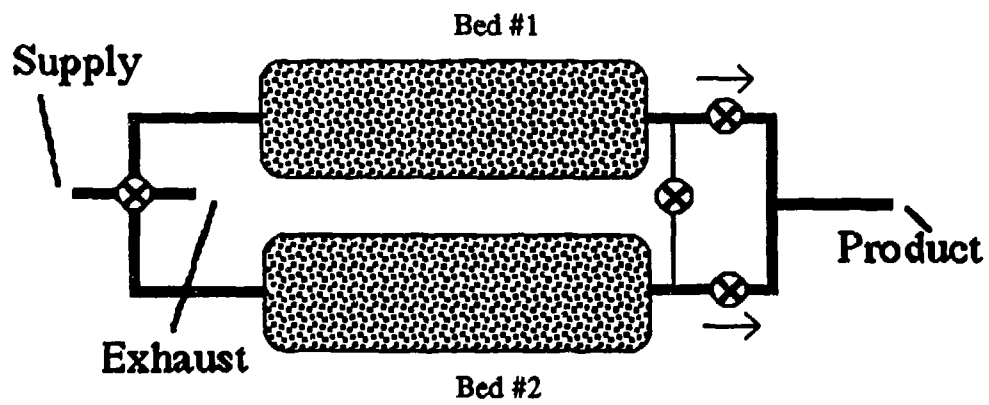
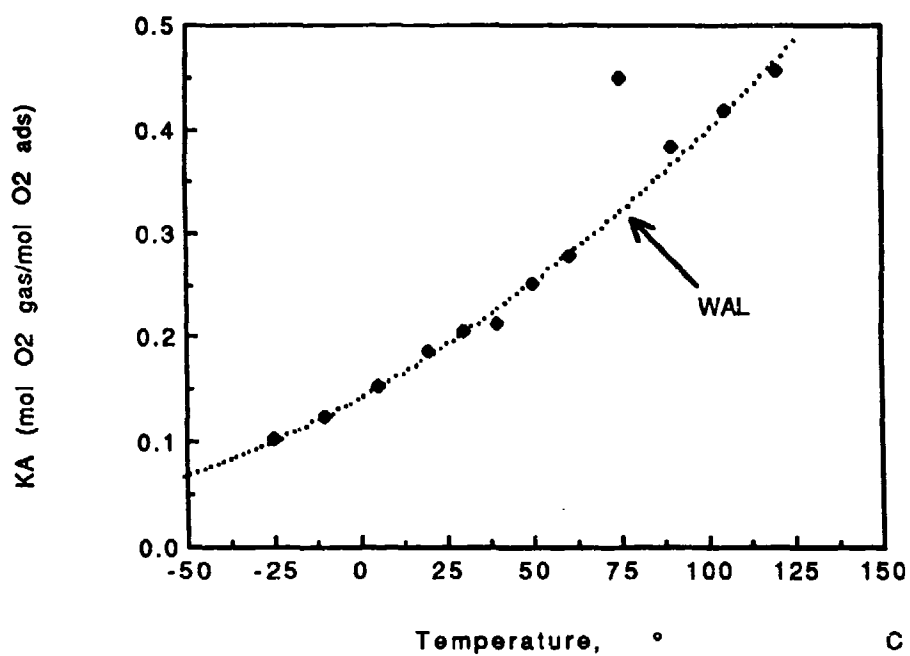


Figure 1: Simple two bed PSA system, with bypass

Figure 2: Zeolite A and X structures, left and right, respectively

Figure 3



wal #23

Figure 3: Temperature dependence of oxygen K_A isotherm parameter, power law fit

Figure 4

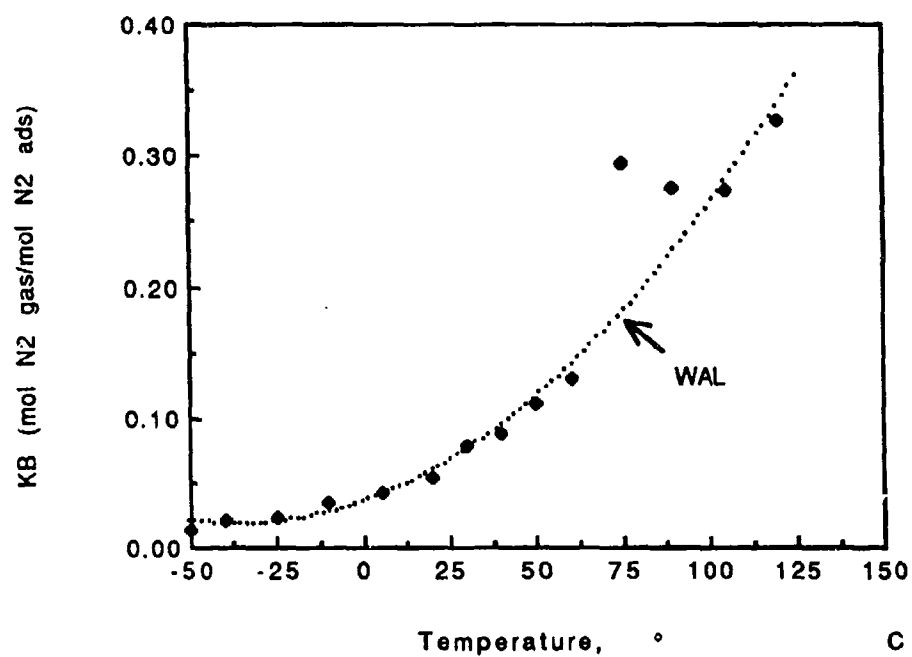
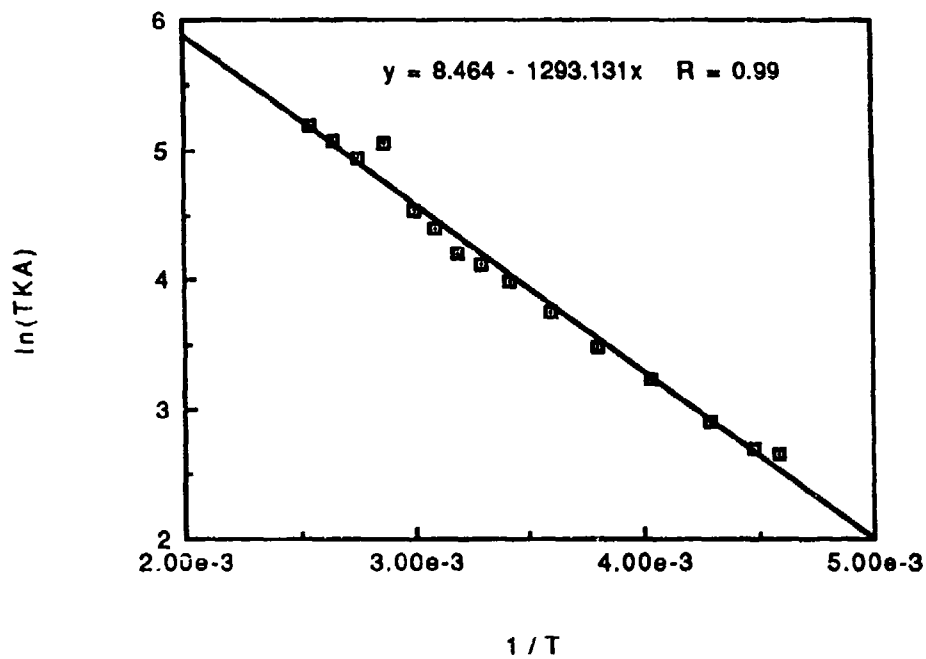


Figure 4: Temperature dependence of nitrogen K_B isotherm parameter, power law fit

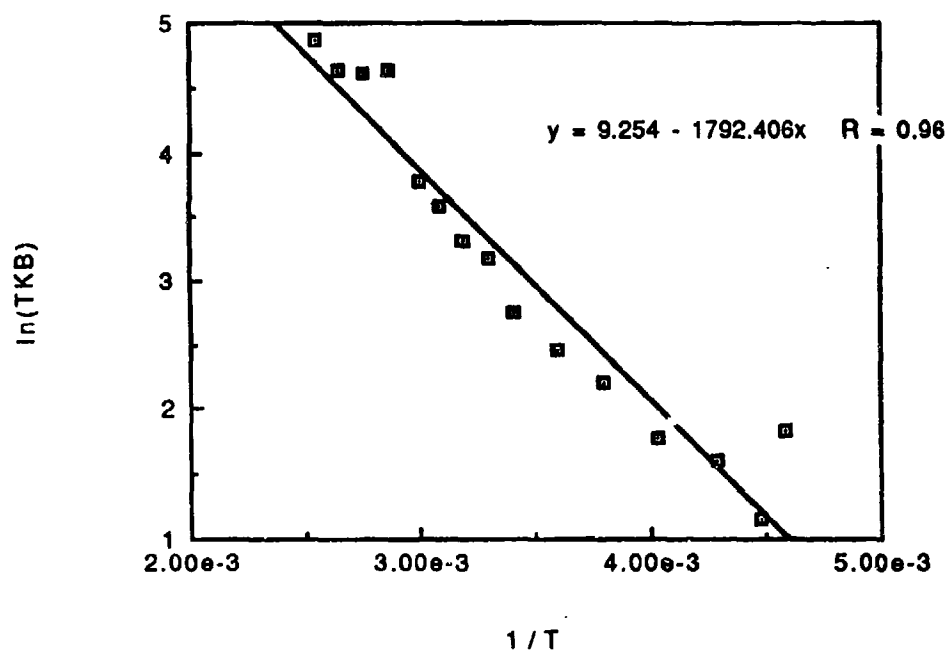
Figure 5



Wal #16

Figure 5: Linear fit of two parameter model to oxygen data

Figure 6



Wal #17

Figure 6: Linear fit of two parameter model to nitrogen data

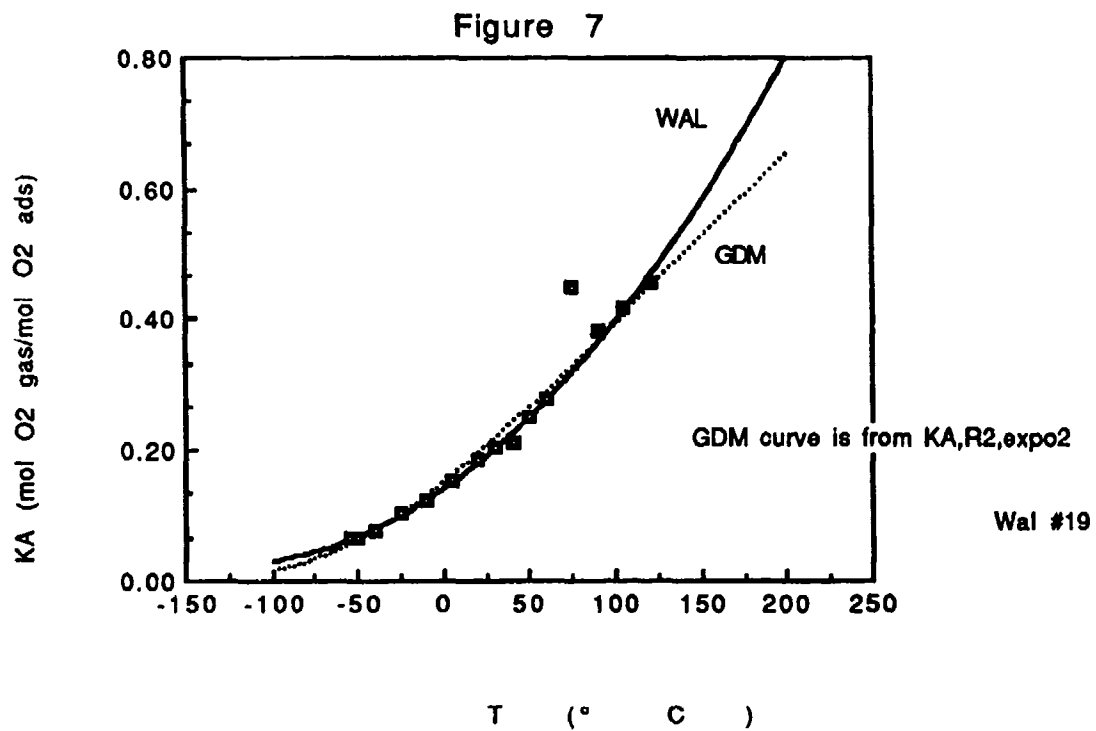


Figure 7: Comparison of two parameter (GDM) and power law (WAL) K_A correlation for oxygen

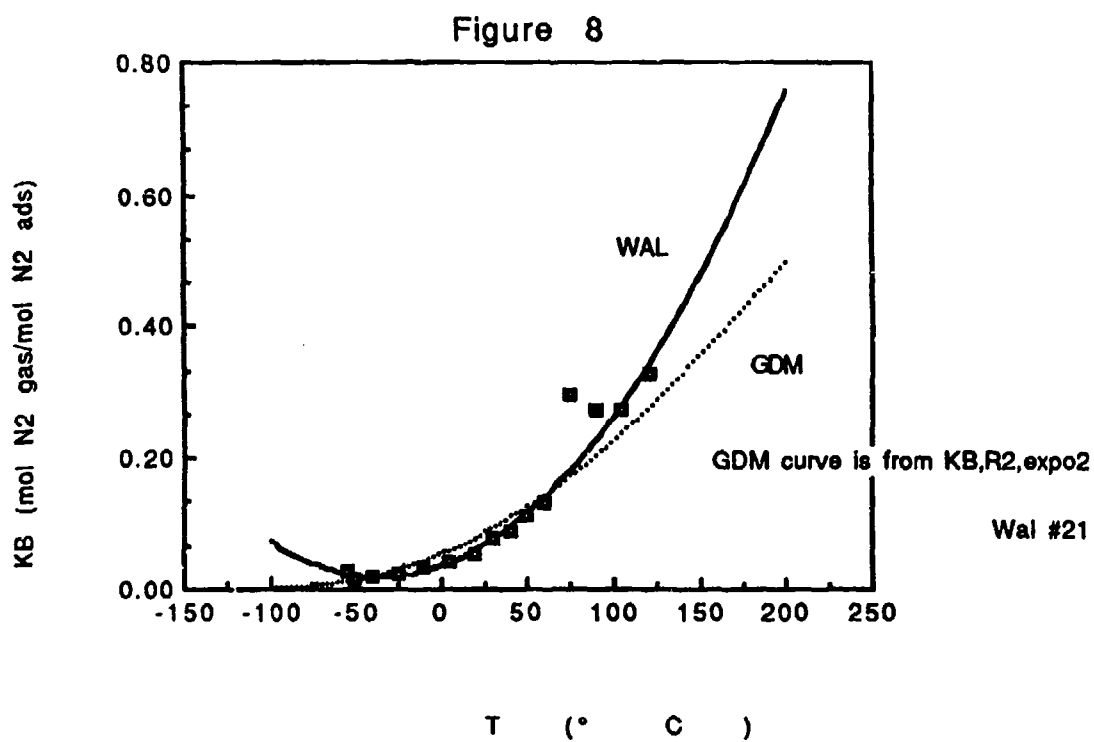


Figure 8: Comparison of two parameter (GDM) and power law (WAL) K_B correlation for nitrogen

Figure 9: Single bed schematic



Figure 10: Single bed apparatus photographs. Top photo is the overall system, showing (left to right) the mass spectrometer, the temperature box, the HP computer system, the IBM XT that collects pressure and temperature information, and the I/O unit. The bottom photo is the single bed inside the temperature box.

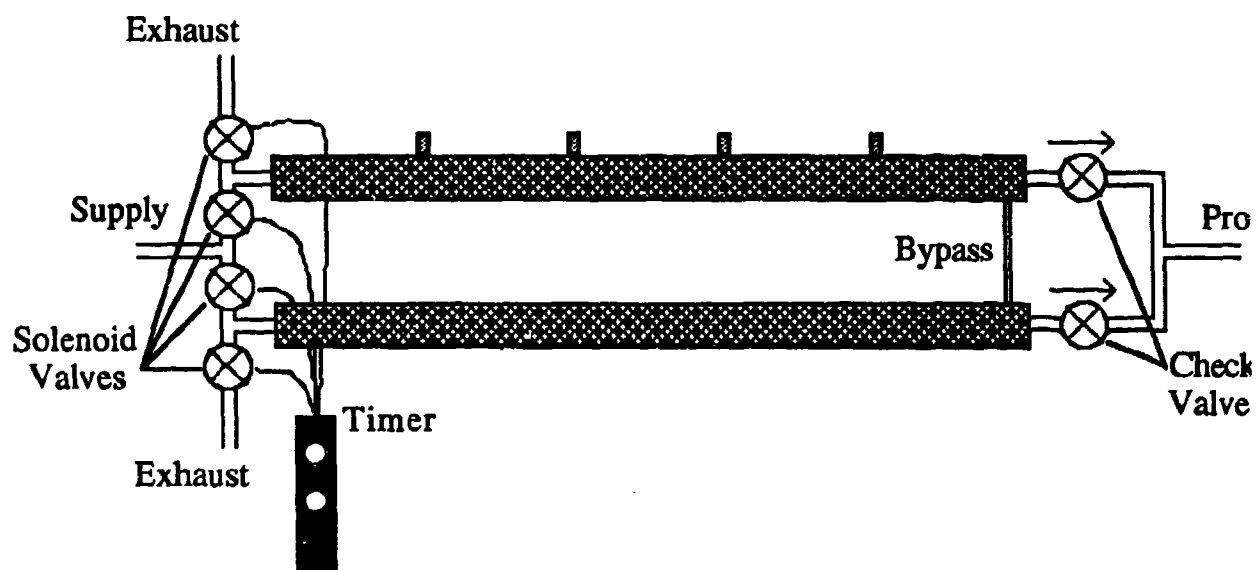


Figure 11: Dual bed apparatus schematic (Solenoid valves are numbered 1 to 4 from the top)

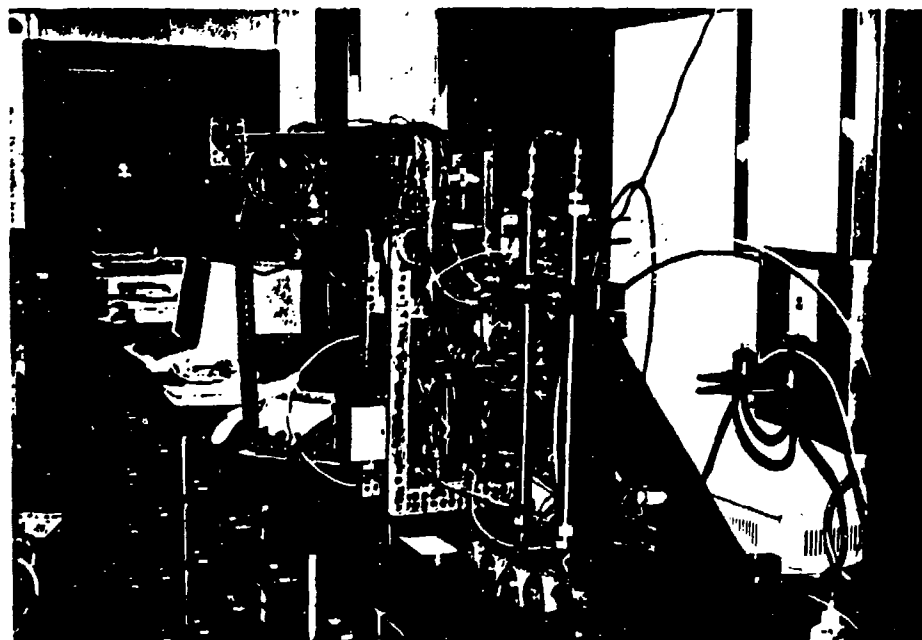
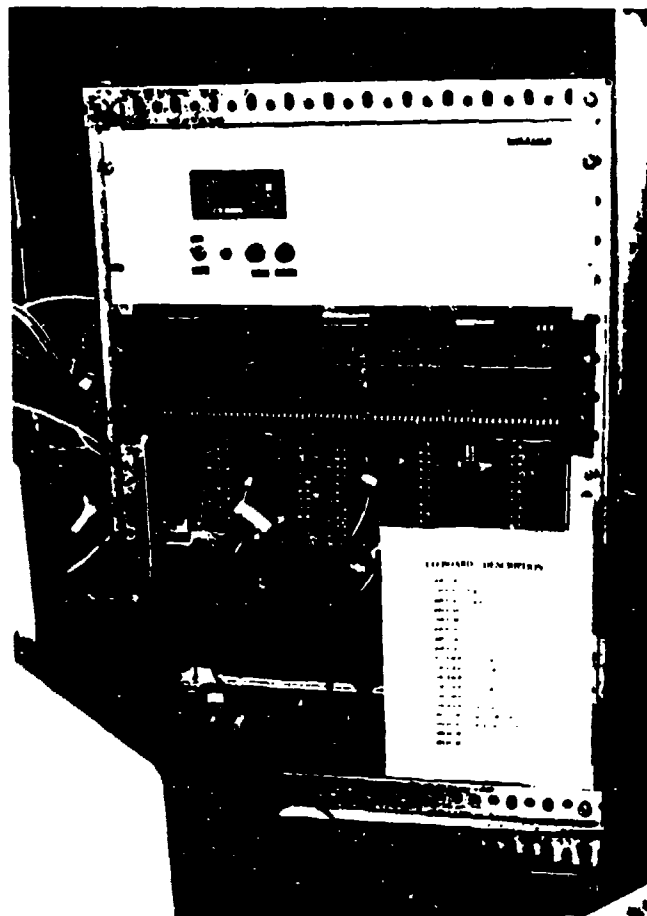


Figure 12: Dual bed apparatus photographs. Top photo is the I/O unit for pressures and temperatures. The bottom photo shows (right to left) the two bed system, mass flow controller set, and the rear of the I/O unit.

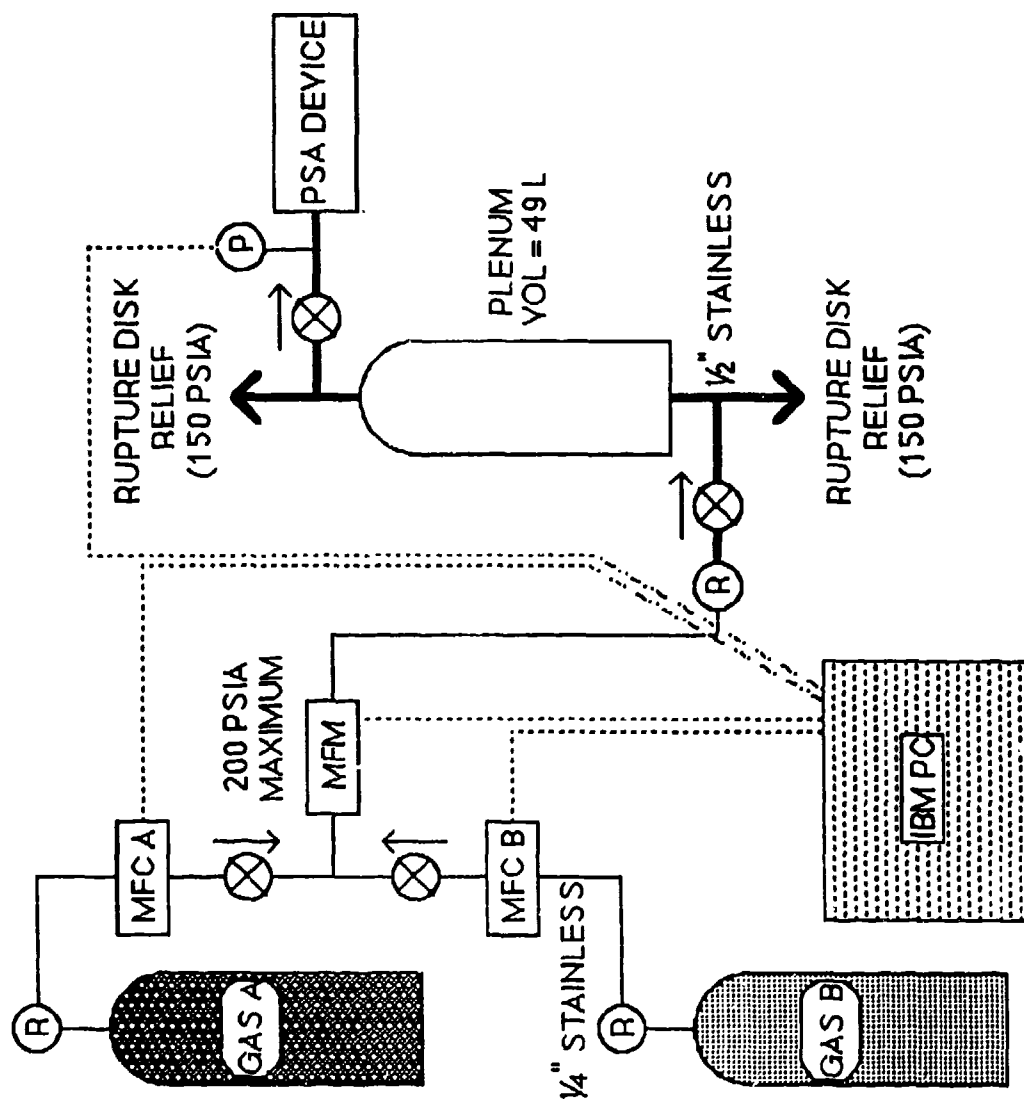


Figure 13: Schematic of feed system for contaminant experiments

Nomenclature

b_i	=	Limit adsorption coefficient for component i
C	=	Total gas phase concentration
C_i^*	=	Interfacial concentration of component i
C_f	=	Heat capacity of fluid
C_i	=	Gas phase concentration of component i
C_s	=	Heat capacity of solid
D_b	=	Diameter of bed
D_m	=	Binary molecular diffusivity
D_p	=	Dispersion coefficient for particle
D_L	=	Axial dispersion coefficient for fluid
F	=	$M_r * (\omega/\omega_0)$
h	=	Heat transfer coefficient at particle surface
h_w	=	Heat transfer coefficient at bed wall
k	=	Mass transfer coefficient
K_i	=	Isotherm coefficient for component i
M_r	=	Ratio of oxygen molecular weight to nitrogen molecular weight
n_1^∞	=	Saturation parameter in VSM
q	=	Total adsorbed phase concentration
q_i	=	Adsorbed phase concentration of component i
Q	=	Pressure ratio in temperature dependence of isotherm experiments
R	=	Radial direction in particle
R_p	=	Radius of particle
t	=	Time
t_{50}	=	Time to 50% breakthrough in temperature dependence of isotherm experiments

T_f	=	Temperature of fluid
T_s	=	Temperature of solid
T_w	=	Temperature of wall
u	=	Fluid velocity
y_i	=	Flux of component i
z	=	Bed distance

Greek Letters:

α	=	Intraparticle void fraction
α_{1v}	=	Non-ideality parameter in VSM
β	=	$k(1 - e)/e$
γ	=	qB/bB
ΔH	=	Heat of adsorption
ε	=	Void fraction
λ_L	=	Axial effective thermal conductivity
μ	=	$(1 - K_B/K_A)$
ω	=	Droop flowrate
ω_0	=	Initial mass flowrate
ρ	=	Bed zeolite density
θ	=	Fractional surface coverage

References

- Beaman, J.J., A. J. Healey, and J. Werlin, Trans. of ASME Journal of Dynamic Systems, Measurement, and Control, **105** (1983) 265.
- Beaman, J. J., Trans. of ASME Journal of Dynamic Systems, Measurement, and Control, **107** (1985) 111.
- Bilgic, A. H., Gas Mixture Adsorption Equilibrium Apparatus, Master's Thesis, Pennsylvania State University (1970).
- Blackburn, J. F., G. Reethof, and J. L. Shearer, Fluid Power Control, MIT Press (1960)
- Cochran, T. W., R. L. Kabel, and R. P. Danner, AIChE J., **31** (1985) 2075.
- Cochran, T. W., R. L. Kabel, and R. P. Danner, AIChE J., **31** (1985) 268.
- Edwards, M.F. and J. F. Richardson, Chem.Eng.Sci., **23** (1968) 109.
- Finlayson, B.A., Nonlinear Analysis in Chemical Engineering, McGraw-Hill (1980).
- Hassan, M. M., N. S. Raghavan, D. M. Ruthven, and H. A. Boniface, AIChE J., **31** (1985) 2008.
- High, M. S., and R. P. Danner, AIChE J., **32** (1986) 1138.
- Karant, N.G., and R. Hughes, Chem.Eng.Sci., **29** (1974) 197.
- Kubin, M., Collection Czech.Chem., **30** (1965) 1104.
- Kubin, M., Collection Czech.Chem., **30** (1965) 2900.
- Kucera, E., J. of Chromato., **19** (1965) ,237.
- Milton, R. M., U. S. Patents 2882243 and 2882244 (1959).
- Myers, A. L., and J. M. Prausnitz, AIChE J., **11** (1965) 121.
- Raghavan, N.S. and Ruthven, D.M., Chem.Eng.Sci., **40** (1985) 699.
- Reilly, P. M., and H. Patino-Leal, Technometrics, **23** (1981).
- Ruthven, D. M., and R. Kumar, Ind. Eng. Chem. Fundam., **19** (1980) 27.

- Ruthven, D. M., Principles of Adsorption and Adsorption processes, Wiley and Sons (1984).
- Sarma, P.M. and H.W. Haynes, Adv. Chem., 133 (1974) 205.
- Schneider, P., and J. M. Smith, AIChE J. 14 (1968a) 762.
- Schneider, P., and J. M. Smith, AIChE J. 14 (1968b) 886.
- Skarstrom, C. W., U. S. Patent 2944627 (1960).
- Suwanayuen, S., and R. P. Danner, AIChE J. 26 (1980) 68.
- Suwanayuen, S., and R. P. Danner, AIChE J. 26 (1980) 76.
- Theis, C. F., K. G. Ikels, and R. G. Dornes, USAFSAM-TR-85-18, December, 1985.
- Villadsen, J.V. and W.E. Stewart, Chem. Eng. Sci., 22 (1967) 1483.
- Yang, B., Master's Thesis, University of Texas (1986).
- Yang, R. T., Gas Separation by Adsorption Processes, Butterworth (1987).

List of Contributors

The following people contributed to research funded through this project:

Joseph J. Beaman, Principal Investigator

In-Won Kim

Glenn D. Munkvold

David Walshak

Brian Yang

Appendices

Appendix A: OBOGS Model Code, Temperature Correlated, for Oxygen/Nitrogen Feed

Appendix B: OBOGS Model Code, Three Component Feed

Appendix C: ISOBANK Code and Manual

Appendix D: On Board Oxygen Generation System (OBOGS) Simulator Manual

APPENDIX A: OBOGS Model Code, Temperature Correlated, for Oxygen/Nitrogen Feed

```

PROGRAM MSEV
  REAL LIN,KA,KB
  BYTE CONTINUE
  BYTE LOWC
  BYTE KEYIN
  BYTE NULL
  BYTE YES
  BYTE NO
  CHARACTER*64 TEMP
  DIMENSION W(3),X(3),CVA(3),PS(2),P(2)
  DIMENSION WV(300,2),TV(300),XV(300,3)
  DIMENSION DL(300),CA1(300),CA2(300),XA1(300,11),
1    XA2(300,11)
  DIMENSION IXABS1(101,11),IYABS1(101,11),IXABS2(101,11),
1    IYABS2(101,11)
  DIMENSION IX1(101),IY1(101),IX2(101),IY2(101)
  DIMENSION JX(6,11),JY(6,11),JXERSH(6,11),JYERSH(6,11),
1    JXERSL(6,11),JYERSL(6,11)
  DIMENSION TI(11),NY(11)
  DIMENSION XCA1(300,11),XCA2(300,11),XBRO2(200)
  COMMON/TIME/T
  COMMON/SYST/DT,PSUP,POUT,TF,WBRL,VS,VO,IIN,TCYC
  COMMON/GEOM/DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN
  COMMON/CAP/KA,KB,B,D,TEMP1
  COMMON/VOID/E
  COMMON/NFINISH/NEXIT
  COMMON/STEP/NSTEP,TSTEP1,TSTEP2,BRSTEP
  COMMON/NPRINT/NOUT1,TI,NY
  COMMON/BEDV/LUMPS,DZ
  COMMON/NBED/NBED,NB
  DATA CONTINUE/67/
  DATA LOWC/99/
  DATA NULL/0/
  DATA YES/89/
  DATA NO/78/
17 CALL MENU
  OPEN(UNIT=1,FILE='MOLEFRAC',STATUS='NEW')
  OPEN(UNIT=2,FILE='FLOWRATE',STATUS='NEW')
  OPEN(UNIT=3,STATUS='SCRATCH')
  OPEN(UNIT=4,FILE='PROFILE',STATUS='NEW')
  OPEN(UNIT=8,STATUS='SCRATCH')
  OPEN(UNIT=9,FILE='REGIS',STATUS='NEW')
  IF ( LUMPS.EQ.101 ) THEN
    IINC = 3
  ELSE
    IINC = 6
  END IF
  IF(NEXIT.EQ.1)GOTO 900
  IF(NSTEP.EQ.1)GOTO 20
  TSTEP1=10000000.
  TSTEP2=10000000.
  BRSTEP=WBRL
20 CONTINUE
  WRITE(6,7)
7 FORMAT(//,
1    .....),
  PRINT *

```

```

PRINT *, 'OBOGS SIMULATION'
PRINT *
IIN=0
ND=0
CDBY=.056/DBYIN
IF(CDBY.GT.1.)CDBY=1.
IF(CDBY.LT..6)CDBY=.6
DBYM=DBYIN*.0254
PRINT *, 'BYPASS VALVE DISCHARGE COEFFICIENT = ',CDBY
BYVA=3.14159*DBYM**2/4.
VS=.8*(.0254*DVSIN)**2*3.14159/4.
VO=.8*(.0254*DVOIN)**2*3.14159/4.
PRINT *, 'AREA OF BY-PASS (M**2) = ',BYVA
PRINT *, 'CD*AREA SUPPLY VALVE (M**2) = ',VS
PRINT *, 'CD*AREA OUTLET VALVE (M**2) = ',VO
PRINT *, 'SUPPLY PRESSURE(PSI)=',PSUP,'OUTLET PRESSURE(PSI)=',POUT
CVA(2)=CDBY*BYVA
T=0.
DT=.05
DATAP=200.
INC=IFIX(TF/(DATAP*DT))
NT=0
IM=0
P(1)=14.5
P(2)=14.5
X(2)=.20
PRINT *, 'BREATHING FLOW (STD LIT/MIN) = ',WBRL
XBR=.20
IF(NOUT1-1)35,35,36
35 CLOSE(UNIT=2,STATUS='DELETE')
CLOSE(UNIT=4,STATUS='DELETE')
CLOSE(UNIT=9,STATUS='DELETE')
WRITE(1,57)
57 FORMAT(' OXYGEN MOLE FRACTION',24X,'TIME')
GO TO 1
36 IF(NOUT1.EQ.4) GOTO 37
CLOSE(UNIT=1,STATUS='DELETE')
CLOSE(UNIT=4,STATUS='DELETE')
CLOSE(UNIT=9,STATUS='DELETE')
WRITE(2,58)
58 FORMAT(' INLET MASS FLOWRATE',T25,' OUTLET MASS FLOWRATE',T55,
1' TIME')
GO TO 1
37 CLOSE(UNIT=1,STATUS='DELETE')
CLOSE(UNIT=2,STATUS='DELETE')
CALL FRAME1
1 CONTINUE
X(1)=.21
X(3)=.21
CALL TIMEF(CVA,PH,PS,T,NBED)
CALL BEDS(P,W,X,XBR,CVA,PS,DT,CA1,CA2,C1,C2)
IF(NOUT1.EQ.4) GOTO 77
GO TO 78
77 DO 1000 K=1,11
IF(NT.EQ.NY(K)) GO TO 150
GO TO 1000
C 150 PRINT *, '*****'
C WRITE(6,170)TI(K)

```

```

C 170 FORMAT(' MOLE FRACTION PROFILE OF OXYGEN AT T=,F6.2,'SECONDS')
C      PRINT '.....'
C  PRINT *, ' BED COORDINATE(IN.):'  BED#1  ' '  BED#2'
150 CONTINUE
    IF(MOD(K,2)) 151,1000,151
151 IF(K.GT.1) THEN

    CALL INITIAL1 ( IY1(1), IY2(1), NB )
    DO J=2,LUMPS
      DLIN = LIN / (lumps-1)
      DL(J) = (J-1) * DLIN
      IF ( NB.EQ.0 ) THEN
        CALL DRAWF ( IX1(J), IY1(J), 1 )
      ELSE
        CALL DRAWF ( IX2(J), IY2(J), 1 )
      END IF
    END DO
    CALL INITIAL2 ( IY1(1), IY2(1), NB )
    DO M=2,LUMPS
      IF ( NB.EQ.0 ) THEN
        CALL DRAWB ( IX2(M), IY2(M), 1 )
      ELSE
        CALL DRAWB ( IX1(M), IY1(M), 1 )
      END IF
    END DO
    ELSE
    END IF

    DO J=2,LUMPS
      DLIN = LIN / (lumps-1)
      DL(J) = (J-1) * DLIN
    END DO
    IF (TTI.GT.0.0) THEN
      CALL NEWTIME(TTI,1)
      CALL NEWTIME(TI(K),0)
      IF (NBED.EQ.NB) THEN
        CONTINUE
      ELSE
        CALL FRAME2(NB,1)
        CALL FRAME2(NBED,0)
      END IF
    ELSE
      CALL NEWTIME(TI(K),0)
      CALL FRAME2(NBED,0)
    END IF
    TTI=TI(K)
    WRITE(4,110)TI(K)
110 FORMAT(' BED COORDINATE',T20,'MOLE FRACTION IN BED#1',T46,
  1'MOLE FRACTION IN BED#2',3X,'(TIME T=,F6.2,'SEC.),'/)
    DO L=1,LUMPS
      XCA1(L,K)=CA1(L)
      XCA2(L,K)=CA2(L)
      IF ( NBED.EQ.0 ) THEN
        XA1(L,K)=XCA1(L,K)/C1
        XA2(L,K)=XCA2(L,K)/C2
        IXABS1(L,K)=275-IINC*(L-1)
        IXABS2(L,K)=575-IINC*(L-1)
        CALL SCALE1(XA1(L,K),IYABS1(L,K))

```



```

CALL SCALE2(XA2(L,K),IYABS2(L,K))
ELSE
M = LUMPS+1-L
XA1(M,K)=XCA1(L,K)/C1
XA2(M,K)=XCA2(L,K)/C2
IXABS2(L,K)=275-IINC*(L-1)
IXABS1(L,K)=575-IINC*(L-1)
CALL SCALE1(XA1(M,K),IYABS1(M,K))
CALL SCALE2(XA2(M,K),IYABS2(M,K))
END IF
END DO

CALL INITIAL1 ( IYABS1(1,K), IYABS2(1,K), NBED )
DO I=2,LUMPS
IF ( NBED.EQ.0 ) THEN
CALL DRAWF ( IXABS1(I,K), IYABS1(I,K), 0 )
IX1(I) = IXABS1(I,K)
IY1(I) = IYABS1(I,K)
ELSE
CALL DRAWF ( IXABS2(I,K), IYABS2(I,K), 0 )
IX2(I) = IXABS2(I,K)
IY2(I) = IYABS2(I,K)
END IF
146 WRITE(4,300) DL(I), XA1(I,K), XA2(I,K)
300 FORMAT ( 5X,F6.3,14X,F10.6,16X,F10.6 )
END DO

CALL INITIAL2 ( IYABS1(1,K), IYABS2(1,K), NBED )
DO 255 N=2,LUMPS
IF ( NBED.EQ.0 ) THEN
CALL DRAWB ( IXABS2(N,K), IYABS2(N,K), 0 )
IX2(N) = IXABS2(N,K)
IY2(N) = IYABS2(N,K)
ELSE
CALL DRAWB ( IXABS1(N,K), IYABS1(N,K), 0 )
IX1(N) = IXABS1(N,K)
IY1(N) = IYABS1(N,K)
END IF
NB = NBED
255 CONTINUE

XBRO22=XBR/1.05
DY = XBRO22
DPY = DY * 280.0
IDY = IFIX ( DPY )
IABS = 412 - IDY
IF(K.GT.1) THEN
WRITE(9,500) IA
500 FORMAT(' P[695,'I3']')
IF(IABS.LT.IA) THEN
ID=IA-IABS
WRITE(9,510) ID
510 FORMAT(' W(R,N0)V[V[695,-'I3']')
ELSE
ID=IABS-IA
WRITE(9,520) ID
520 FORMAT(' W(R,N1)V[V[695,+'I3']')
END IF

```

```

ELSE
  WRITE(9,530) IABS
530  FORMAT(' P[695,412]V[, 'I3']')
  END IF
  WRITE(9,535)
535  FORMAT(' W(N0)')
C   WRITE(6,200)DL(I),XA1(I,K),XA2(I,K)
C 200  FORMAT(8X,F6.3,11X,E10.4,7X,E10.4)
  IA=IABS
1000 CONTINUE
  IF(NT.EQ.NY(11))THEN
    CLOSE(UNIT=9,STATUS='SAVE')
    REWIND 9
    OPEN(UNIT=9,FILE='REGIS',STATUS='OLD')
    CALL TXTERASE (0)
    CALL REGISTART
    CALL PLTERASE
9200  READ(9,9220,END=9999) TEMP
      WRITE(3,*) TEMP
      TYPE 9220,TEMP
9220  FORMAT(A)
      GO TO 9200
9999  CALL REGISOUT
      CLOSE(UNIT=9,STATUS='DELETE')
      WRITE(6,9450)
9450  FORMAT(' PRESS "1" TO CONTINUE :')
9400  READ(5,*) KKKK
      IF(KKKK.EQ.1) THEN
        CALL TXTERASE (0)
        CALL REGISTART
        CALL PLTERASE
        CALL REGISOUT
      ELSE
        GO TO 9400
      ENDIF
      GO TO 17
    ELSE
      GO TO 6
    ENDIF

78  IF(MOD(NT,INC))5,5,6
5   IM=IM+1
    XBRO2(IM) = XBR/1.05
    IF(NOUT1.EQ.1) THEN
      WRITE(1,84) XBRO2(IM),T
84  FORMAT (6X,F10.6,25X,F10.3)
82  IF(ND.EQ.0) THEN
      CALL TXTERASE (255)
      CALL REGISTART
      CALL PLTERASE
      CALL FRAME
      TYPE 80
80  FORMAT('IP[130,330]', $)
      GO TO 47
    ELSE
      CALL STIME(T,IX,TF,600.,130)
      CALL SCALEM(XBRO2(IM),IY,1.0,1.0)
      TYPE 85,IX,IY

```

```

85 FORMAT('IV['I3','I3']',$)
END IF
ELSE IF(NOUT1.EQ.2) THEN
IF(ND.EQ.0) THEN
CALL TXTERASE (255)
CALL REGISTART
CALL PLTERASE
CALL FRAME
TYPE 86
86 FORMAT('IP[130,330]',$)
GO TO 47
ELSE
CALL STIME(T,IX,TF,600.,130)
CALL SCALEM(WV(ND,1),IY,0.08,0.1)
TYPE 87,IX,IY
87 FORMAT('IV['I3','I3']',$)
WRITE(2,68)WV(ND,1),WV(ND,2),T
68 FORMAT(8X,F10.6,11X,F10.6,10X,F10.3)
END IF
ELSE IF(NOUT1.EQ.3) THEN
IF(ND.EQ.0) THEN
CALL TXTERASE (255)
CALL REGISTART
CALL PLTERASE
CALL FRAME
TYPE 88
88 FORMAT('IP[130,330]',$)
GO TO 47
ELSE
CALL STIME(T,IX,TF,600.,130)
CALL SCALEM(WV(ND,2),IY,0.08,0.1)
TYPE 89,IX,IY
89 FORMAT('IV['I3','I3']',$)
WRITE(2,69)WV(ND,1),WV(ND,2),T
69 FORMAT(8X,F10.6,11X,F10.6,10X,F10.6,6X,F10.3)
END IF
ELSE
GO TO 82
END IF
47 CONTINUE
ND=ND+1
IF(PH-180.)2,3,3
2 WV(ND,1)=W(3)
WV(ND,2)=W(1)
GO TO 4
3 WV(ND,1)=-W(1)
WV(ND,2)=-W(3)
4 XV(ND,1)=XBRO2(IM)
XV(ND,2)=0.
XV(ND,3)=1.
TV(ND)=T
6 NT=NT+1
T=T+DT

IF ( T.LT.TSTEP1 .AND. T.LT.TF ) THEN
WBRLD = WBRL
GO TO 1
ELSE IF ( T.GE.TSTEP1 .AND. T.LT.TSTEP2 ) THEN

```

```

WBRL = BRSTEP
GO TO 1
ELSE IF ( T.GE.TSTEP2 .AND. T.LE.TF ) THEN
WBRL = WBRLOLD
GO TO 1
ELSE
WBRL = WBRLOLD
GO TO 18
ENDIF

18 CONTINUE
CALL REGISOUT
WRITE(6,42) NULL
42 FORMAT(A,' PLEASE PRESS "1" TO CONTINUE :')
44 READ(5,'') KKKK
IF(KKKK.EQ.1) THEN
CALL TXTERASE (0)
CALL REGISTART
GO TO 43
ELSE
GO TO 44
ENDIF
43 CALL PLTERASE
CALL REGISOUT
100 FORMAT(14X,8E12.3)
GO TO 17
900 WRITE(6,910)
910 FORMAT('////////'//,
' PROGRAM TERMINATED BY OPERATOR',
2'////////'//)
END

BLOCK DATA
REAL LIN,KA,KB
COMMON/SYST/DT,PSUP,POUT,TF,WBRL,VS,VO,IIN,TCYC
COMMON/GEOM/DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN
COMMON/CAP/KA,KB,B,D,TEMP1
COMMON/VOID/E
DATA DT,PSUP,POUT,TF,WBRL,VS,VO,IIN,TCYC/.05,40.,15.,30.,
1 10.,0.,0.,0,10.7/
DATA DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN/
1 .075,306174.,43866,15.5,5.73,2.18/
DATA KA,KB,B,D/.153,.049,2.65,200./
DATA E/.37/,TEMP1/20./
END

SUBROUTINE MENU
REAL LIN,KA,KB,LIN1,KA1,KB1
DIMENSION TI(11),NY(11)
COMMON/SYST/DT,PSUP,POUT,TF,WBRL,VS,VO,IIN,TCYC
COMMON/GEOM/DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN
COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,RT,AI,AO,AMW01,AMWL1,AMW02,
AMWL2,WBR
COMMON/VOID/E
COMMON/CAP/KA,KB,B,D,TEMP1
COMMON/NFINISH/NEXIT
COMMON/STEP/NSTEP,TSTEP1,TSTEP2,BRSTEP
COMMON/NPRINT/NOUT1,TI,NY

```

```

NEXIT=0
NSTEP=0
10 CALL SYST1
CALL GEOM1
CALL CAP1
WRITE(7,107)
WRITE(6,107)
107 FORMAT(' ENTER CORRESPONDING PARAMETER# TO CHANGE PARAMETER',/
1' ENTER "0" TO RUN SIMULATION',
2' "59" TO STEP THE BREATHING FLOWRATE',/
4' ENTER "99" TO EXIT PROGRAM')
N=0
READ *,N
IF(N.EQ. 0) GOTO 600
IF(N.EQ.59) GOTO 500
IF(N.EQ.99) GOTO 900
EPSI=.0000001
WRITE(7,120)
WRITE(6,120)
120 FORMAT(///, '.....',
1  ///, ' ENTER THE PARAMETER AND PRESS "RETURN"',/,
2//T40, 'CURRENT VALUE',/)
IF(N.LT.6)GOTO 100
IF(N.LT.12)GOTO 200
IF(N.LE.16)GOTO 300
GOTO 800
100 CONTINUE
IF(N-2)159,155,155
155 IF(N-3)164,171,171
171 IF(N-4)170,175,174
159 WRITE(6,160) PSUP
WRITE(7,160) PSUP
160 FORMAT(' SUPPLY PRESSURE (PSIA)',T40,F8.4,/)
READ *,PSUP1
IF(PSUP1.LT.EPSI) GOTO 10
PSUP=PSUP1
GOTO 10
164 WRITE(6,166) POUT
166 FORMAT(' OUTLET PRESSURE (PSIA)',T40,F8.4,/)
READ *,POUT1
IF(POUT1.LT.EPSI) GOTO 10
POUT=POUT1
GOTO 10
170 WRITE(6,172) TF
172 FORMAT(' FINAL OBSERVATION TIME (SEC)',T40,F8.4,/)
READ *,TF1
IF(TF1.LT.EPSI) GOTO 10
TF=TF1
GOTO 10
175 WRITE(6,177) TCYC
177 FORMAT(' CYCLE TIME (SEC)',T40,F8.4,/)
READ *,TCYC1
IF(TCYC1.LT.EPSI) GOTO 10
TCYC=TCYC1
GOTO 10
174 WRITE(6,176) WBRL
176 FORMAT(' BREATHING MASS FLOWRATE (STD LIT/MIN)',T40,F8.4,/)
READ *,WBRL1

```

```

IF(WBRL1.LT.EPSI) GOTO 178
WBRL=WBRL1
GOTO 10
178 GO TO 10
200 CONTINUE
IF(N-7) 209,215,205
205 IF(N-9) 225,235,207
207 IF(N-10) 235,245,255
209 WRITE(6,210) DBYIN
210 FORMAT(' BY-PASS VALVE DIAMETER (IN)',T40,F8.4,/)
READ *,DBYIN1
IF(DBYIN1.LT.EPSI) GOTO 10
DBYIN=DBYIN1
GOTO 10
215 WRITE(6,220) DVSIN
220 FORMAT(' SUPPLY VALVE DIAMETER (IN)',T40,F8.4,/)
READ *,DVSIN1
IF(DVSIN1.LT.EPSI) GOTO 10
DVSIN=DVSIN1
GOTO 10
225 WRITE(6,230) DVOIN
230 FORMAT(' OUTLET VALVE DIAMETER (IN)',T40,F8.4,/)
READ *,DVOIN1
IF(DVOIN1.LT.EPSI) GOTO 10
DVOIN=DVOIN1
GOTO 10
235 WRITE(6,240) LIN
240 FORMAT(' BED LENGTH (IN)',T40,F8.4,/)
READ *,LIN1
IF(LIN1.LT.EPSI) GOTO 10
LIN=LIN1
GOTO 10
245 WRITE(6,250) DIAOIN
250 FORMAT(' OUTER BED DIAMETER (IN)',T40,F8.4,/)
READ *,DIAOIN1
IF(DIAOIN1.LT.EPSI) GOTO 10
DIAOIN=DIAOIN1
GOTO 10
255 WRITE(6,260) DIAIIN
260 FORMAT(' INNER BED DIAMETER (IN)',T40,F8.4,/)
READ *,DIAIIN1
IF(DIAIIN1.LT.EPSI) GOTO 10
DIAIIN=DIAIIN1
GOTO 10
300 CONTINUE
IF(N-13)309,325,305
305 IF(N-15)335,345,10
309 WRITE(6,310) TEMP1
310 FORMAT(' TEMPERATURE (C)',T40,F8.4,/)
READ *,TEMP2
IF(TEMP2.LT.EPSI) GOTO 10
TEMP1=TEMP2
GOTO 10
325 WRITE(6,330) B
330 FORMAT(' B',T40,F8.4,/)
READ *,B1
IF(B1.LT.EPSI) GOTO 10
B=B1

```

```

GOTO 10
335 WRITE(6,340) D
340 FORMAT(' DIFFUSION COEFFICIENT (>100.)',T40,F8.4,/)
  READ*,D1
  IF(D1.LT.EPSI) GOTO 10
  D=D1
  GOTO 10
345 WRITE(6,350) E
350 FORMAT(' VOID FRACTION (<1.)',T40,F8.4,/)
  READ*,E1
  IF(E1.LT.EPSI) GOTO 10
  E=E1
  GOTO 10
500 WRITE(6,510) TF
  WRITE(7,510) TF
  510 FORMAT('*****',/,
    1' ENTER THE TIME AT WHICH THE STEP CHANGE',/
    2' IN BREATHING FLOWRATE IS TO OCCUR',/,
    3' THE FINAL OBSERVATION TIME IS CURRENTLY ',F7.2,' SECONDS',//)
  NSTEP=1
  READ*,TSTEP1
515 WRITE(6,518) TF
  WRITE(7,518) TF
  518 FORMAT('*****',/,
    1' ENTER THE TIME AT WHICH THE STEP CHANGE',/
    2' IN BREATHING FLOWRATE IS TO END',/,
    3' THE FINAL OBSERVATION TIME IS CURRENTLY ',F7.2,' SECONDS',//)
  READ*,TSTEP2
  WRITE(7,520) TSTEP1,TSTEP2,WBRL
  WRITE(6,520) TSTEP1,TSTEP2,WBRL
  520 FORMAT('*****',/,
    1' ENTER THE NEW BREATHING FLOWRATE OCCURING AS',/
    2' A STEP CHANGE FROM T= ',F7.2,' TO T= ',F7.2,' SECONDS',/,
    3' THE CURRENT BREATHING FLOWRATE IS',F6.2,' STD LIT/MIN',//)
  READ*,BRSTEP
  WRITE(7,530) BRSTEP,TSTEP1,TSTEP2
  WRITE(6,530) BRSTEP,TSTEP1,TSTEP2
530 FORMAT('///,
  1' *****',/,
  2' THE BREATHING FLOWRATE OF ',F6.2,' STD LIT/MIN',/
  3' WILL BE INPUT AS A STEP AT T= ',F7.2,' SECONDS',/
  3' AND WILL END AT T= ',F7.2,' SECONDS'////,
  4' *****',//)
  GOTO 10
600 CONTINUE
  WRITE(7,610)
  WRITE(6,610)
  610 FORMAT('/////','*****',/,
    1' THE FOLLOWING PLOTS ARE AVAILABLE TO YOU ON THE TERMINAL',/,
    1' 1 MOLE FRACTION OF OXYGEN VS.TIME (DATA FILE=MOLEFRAC.DAT)',/,
    2' 2 INLET MASS FLOWRATE VS. TIME (DATA FILE=FLOWRATE.DAT)',/,
    3' 3 OUTLET MASS FLOWRATE VS. TIME (DATA FILE=FLOWRATE.DAT)',/,
    4' 4 A DYNAMIC SIMULATION FOR ONE CYCLE (OXYGEN VS. TIME',/,
    DATA FILE=PROFILE.DAT)',
    .,/,
    1' PRESS THE CORRESPONDING # TO HAVE THE OUTPUT SHOWN AS THE',/,
    1' SIMULATION TRANSPIRES',//)
  READ(5,630)NOUT1

```

```

630 FORMAT(I1)
   IF(NOUT1.EQ.4) GOTO 750
   GO TO 400
750 DTI=1.0
   DO J=1,10
     TI(J)=TI(J-1)+DTI
     NY(J)=IFIX(TI(J)/DT)
   END DO
   TI(11)=TCYC
   NY(11)=IFIX(TI(11)/DT)
   GO TO 400
400 WRITE(6,410)
   WRITE(7,410)
410 FORMAT(////,' YOU HAVE THE FOLLOWING OPTIONS :',
  .////,T10,'1. ENTER "0" TO RUN SIMULATION.',/,
  .T10,'2. ENTER "19" TO CHANGE THE PARAMETERS.',/,
  .T10,'3. ENTER "99" TO EXIT PROGRAM.',////,
  .ENTER THE CORRESPONDING # AND PRESS "RETURN",////)
   READ *, N69
   IF(N69.EQ.0) GOTO 820
   IF(N69.EQ.19) GOTO 10
   IF(N69.EQ.99) GOTO 900
800 CONTINUE
   WRITE(7,810)
   WRITE(6,810)
810 FORMAT(////,'*****',////,
  1'THE PARAMETER VALUE WAS NOT INPUT CORRECTLY',/
  2' TRY AGAIN',////,'*****',////)
   GOTO 10
820 WRITE(6,825)
   WRITE(7,825)
825 FORMAT(////,' DO YOU WISH TO DOUBLE THE NUMBER OF SPACE LUMPS IN',
  1'THE SIMULATION FOR BETTER ACCURACY ? ( CURRENT NUMBER USED ',
  2' IS 50 ) (Y/N)',////)
   READ (5,830) KEYIN
830 FORMAT (A)
   IF (KEYIN.EQ.YES) THEN
     LUMPS = 101
   ELSE
     LUMPS = 51
   END IF
   GO TO 1000
900 CONTINUE
   NEXIT=1
1000 RETURN
   END

SUBROUTINE SYST1
COMMON/SYST/DT,PSUP,POUT,TF,WBRL,VS,VO,IIN,TCYC
WRITE(6,350) PSUP,POUT,TF,TCYC,WBRL
WRITE(7,350) PSUP,POUT,TF,TCYC,WBRL
350 FORMAT(' CURRENT SYSTEM PARAMETERS ARE:',/,
  1' 1.,T5,' SUPPLY PRESSURE=',T30,F8.2,T40,'PSIA',/,
  2' 2.,T5,' OUTLET PRESSURE=',T30,F8.2,T40,'PSIA',/,
  3' 3.,T5,' FINAL OBSERVATION TIME=',T30,F8.2,T40,'SEC',/,
  4' 4.,T5,' CYCLE TIME=',T30,F8.2,T40,'SEC',/,
  5' 5.,T5,' BREATHING FLOWRATE=',T30,F8.2,T40,'STD LIT/MIN')
RETURN

```



```

END

SUBROUTINE GEOM1
REAL LIN
COMMON/GEOM/DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN
WRITE(6,450) DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN
WRITE(7,450) DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN
450 FORMAT(' CURRENT GEOMETRIC PARAMETERS ARE: ',
1' 6.',T5,' BY-PASS VALVE DIAMETER = ',T30,F8.4,T40,'IN'/,
2' 7.',T5,' SUPPLY VALVE DIAMETER = ',T30,F8.4,T40,'IN'/,
3' 8.',T5,' OUTLET VALVE DIAMETER = ',T30,F8.4,T40,'IN'/,
4' 9.',T5,' BED LENGTH = ',T30,F8.4,T40,'IN'/,
5' 10.',T5,' OUTER BED DIAMETER = ',T30,F8.4,T40,'IN'/,
6' 11.',T5,' INNER BED DIAMETER = ',T30,F8.4,T40,'IN')
RETURN
END

SUBROUTINE CAP1
REAL KA,KB
COMMON/CAP/KA,KB,B,D,TEMP1
COMMON/VOID/E
KA=.1423275+.00183744*TEMP1+.0000072443*TEMP1**2
KB=.03654573+.00095775*TEMP1+.00001321*TEMP1**2
B=3.0591250816
WRITE(7,550)TEMP1,B,D,E,KA,KB
WRITE(6,550)TEMP1,B,D,E,KA,KB
550 FORMAT(' THE CURRENT BED PARAMETERS ARE: ',
1' 12.',T5,' TEMPERATURE = ',T30,F8.3,T40,'DEGREES C'/,
2' 13.',T5,' B = ',T30,F8.4/,
3' 14.',T5,' DIFFUSION COEFFICIENT = ',T30,F8.2,T40,'1/SEC'/,
4' 15.',T5,' VOID FRACTION = ',T30,F8.3/,
5' T5,' AT THIS TEMPERATURE, KA = ',T30,F8.4,T40,
6' KGMOL/SEC 02 ADSORBED/KGMOL GAS'/,
7' T5,21X,' KB = ',T30,F8.4,T40,'KGMOL N2 ADSORBED/KGMOL GAS')
RETURN
END

SUBROUTINE TIMEF(CVA,PH,PS,T,NBED)
C*****CALCULATES CVA(T) AND PS(T)
DIMENSION CVA(3),PS(2)
DATA PHI,PH1,PH2,PH3,PH4,PH5,PH6/10.,20.,160.,180.,
1 200.,340.,360./
COMMON/SYST/DT,PSUP,POUT,TF,WBRL,VS,VO,IIN,TCYC
C.....
C
C PH = PHASE ANGLE OF CYCLE (DEGREES)
C TCYC = CYCLE TIME (SEC)
C.....
I=0
PH=AMOD(360.*T/TCYC+PHI,360.)
IF(PH-PH1)1,2,2
1 RA=PH/PH1
GO TO 11
2 IF(PH-PH2)3,4,4
3 RA=1.
GO TO 11
4 IF(PH-PH3)5,6,6

```

```

5  RA=(PH3-PH)/(PH3-PH2)
   GO TO 11
6  IF(PH-PH4)7,8,8
7  RA=(PH-PH3)/(PH4-PH3)
   I=1
   GO TO 11
8  IF(PH-PH5)9,10,10
9  RA=1.
   I=1
   GO TO 11
10 RA=(PH6-PH)/(PH6-PH5)
   I=1
11 NBED=I
   IF(I)12,12,13
12 PS(1)=PSUP
   PS(2)=POUT
   CVA(1)=VS*RA
   CVA(3)=VO*RA
   RETURN
13 PS(1)=POUT
   PS(2)=PSUP
   CVA(1)=VO*RA
   CVA(3)=VS*RA
   RETURN
   END

SUBROUTINE INITIAL(C1,C2,N1,N2,X,P)
  REAL L,MA,MB,NA01,NA02,KA,KB,N1,N2,NB01,NB02,LIN
  DIMENSION N1(300,2),N2(300,2),X(3),P(2)
  COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,RT,AI,AO,AMW01,AMWL1,AMW02
  1,AMWL2,WBR
  COMMON/CAP/KA,KB,B,D,TEMP1
  COMMON/VOID/E
  COMMON/GEOM/DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN
  C.....
  C
  C   RT = GAS CONSTANT*TEMP (N-M/KGMOLE)
  C   AITOT,AOTOT = INLET AND OUTLET CONCENTRIC BED AREAS (M**2)
  C   MA,MB = KGMOLECULAR WEIGHTS (KG/KGMOLE)
  C   0,L = SUBSCRIPTS FOR Z=0 AND Z=L
  C   AMW = AVERAGE MOLECULAR WEIGHT (KG/KGMOLE)
  C   1,2 = SUBSCRIPTS FOR BED 1 AND BED 2
  C   E = VOID FRACTION
  C   TEMP1 = TEMPERATURE (DEGREES C)
  C.....
  RT=8314.*(TEMP1+273.16)
  AITOT=3.14159*(DIAIIN*.0254)**2/4.
  AOTOT=3.14159*(DIAOIN**2-DIAIIN**2)*.0254**2/4.
  L=.0254*LIN
  PRINT *, ' BED LENGTH (M) = ',L
  PRINT *, ' INLET BED AREA (M**2) = ',AITOT
  PRINT *, ' OUTLET BED AREA (M**2) = ',AOTOT
  MA=32.
  PRINT *, ' VOID FRACTION = ',E
  MB=28.
  AI=AITOT*E
  AO=AOTOT*E
  AMW01=MA*X(1)+MB*(1.-X(1))

```

```

AMWL1=MA*X(2)+MB*(1.-X(2))
AMW02=AMWL1
AMWL2=MA*X(3)+MB*(1.-X(3))
C1=P(1)*6.895E3/RT
C2=P(2)*6.895E3/RT
CA1=C1*X(1)
CA2=C2*X(3)
CB1=C1-CA1
CB2=C2-CA2
NB01=(CB1/KB)/(1.+CB1/(KB*B))
NA01=CA1/KA
NB02=(CB2/KB)/(1.+CB2/(KB*B))
NA02=CA2/KA
DO 1 I=1,LUMPS
N1(I,1)=NA01+NB01
N1(I,2)=NA01
N2(I,1)=NA02+NB02
1 N2(I,2)=NA02
DZ=L/FLOAT(LUMPS-1)
RETURN
END

SUBROUTINE BEDS(P,W,X,XBR,CVA,PS,DTG,CA1,CA2,C1,C2)
C*****SOLVES BED EQUATIONS
REAL L,N1,N2,MA,MB
DIMENSION CA1(300),CA2(300),N1(300,2),N2(300,2),G1(300,2),
1 G2(300,2)
1 ,W(3),X(3),CVA(3),PS(2),PN(2),P(2),XT(2),XO(2)
COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,RT,AI,AO,AMW01,AMWL1,AMW02
1 ,AMWL2,WBR
COMMON/VOID/E
COMMON/SYST/DT,PSUP,POUT,TF,WBRL,VS,VO,IIN,TCYC
COMMON/CAP/KA,KB,B,D,TEMP1
COMMON/TIME/T
C*****
C
C C = TOTAL GAS STREAM CONCENTRATION (KGMOLE/M**3)
C CA = GAS A STREAM CONCENTRATION (KGMOLE/M**3)
C N = ADSORBED PHASE CONCENTRATION (KGMOLE/M**3)
C G = GAS PHASE CONCENTRATION (KGMOLE/M**3)
C U = STREAM VELOCITY (M/SEC)
C Y = MOLECULAR FLUX (KGMOLE/M**2-SEC)
C L = BED LENGTH (M)
C D = DIFFUSION COEFFICIENT (1/SEC)
C BETA = D*AREA RATIO (1/SEC)
C DT = TIME STEP (SEC)
C LUMPS = NUMBER OF SPACE LUMPS
C 1,2 = SUBSCRIPTS FOR BED 1 AND BED 2
C VBR = MIXING VOLUME FOR BREATHING FLOW (M**3)
C BR = SUBSCRIPT INDICATING BREATHING VARIABLE
C CP = PLENUM MOLE CONCENTRATION (KGMOLE/M**3)
C VP = BREATHING PLENUM VOLUME (M**3)
C*****
C
DT=DTG
DO 4 I=1,2
4 PN(I)=PS(I)*6.895E3
IF(IIN.EQ.1)GO TO 1

```

```

IIN=1
CALL INITIAL(C1,C2,N1,N2,X,P)
VPMUL=1.
VP=.0018*VPMUL
PRINT*, ' PLENUM VOLUME (M**2) = ',VP
DO I=1,7
PRINT*
END DO
WRITE(6,10) NULL
10 FORMAT(A, ' PLEASE WAIT.....')
VBR=.5*VP+.0003
CP=.95*AMAX1(PN(1),PN(2))/RT
DO 3 I=1,2
3 XT(I)=XBR
1 D1=D
D2=D
WBR=WBRL*4.72E-5/2.205
CALL CONCEN(C1,C2,CA1,CA2,W,G1,G2,X,N1,N2,PN,CVA)
XBR=XT(2)
AMWBR=MA*XBR+MB*(1.-XBR)
DO 5 I=1,2
5 XO(I)=XT(I)
DO 2 I=1,LUMPS
N1(I,1)=N1(I,1)-D1*(G1(I,1)-C1)*DT
N1(I,2)=N1(I,2)-D1*(G1(I,2)-CA1(I))*DT
N2(I,1)=N2(I,1)-D2*(G2(I,1)-C2)*DT
N2(I,2)=N2(I,2)-D2*(G2(I,2)-CA2(I))*DT
2 CONTINUE
FAC=CP*VBR*AMWBR
TAUI=WBR/FAC
XT(1)=XO(1)+TAUI*(X(2)-XO(1))*DT
XT(2)=XO(2)+TAUI*(XO(1)-XO(2))*DT
P(1)=C1*RT/6.895E3
P(2)=C2*RT/6.895E3
AMW01=MA*X(1)+MB*(1.-X(1))
AMWL1=MA*X(2)+MB*(1.-X(2))
AMW02=AMWL1
AMWL2=MA*X(3)+MB*(1.-X(3))
RETURN
END

SUBROUTINE CONCEN(CG1,CG2,CA1,CA2,W,G1,G2,X,N1,N2,PN,CVA)
C*****SOLVES FOR CONCENTRATIONS
REAL L,N1,N2,MA,MB
DIMENSION CA1(300),CA2(300),N1(300,2),N2(300,2),G1(300,2),
1 G2(300,2)
DIMENSION G1(300),G2(300),PN(2),W(3),X(3),CVA(3)
COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,RT,AI,AO,AMW01,AMWL1,AMW02
1,AMWL2,WBR
COMMON/VOID/E
COMMON/VBEQ/W1,W2,W3,U01,UL1,U02,UL2,PN1,PN2,VA1,VA2,VA3
1,BGI1L,BGI1L2,BGI2L,BGI2L2,BL21,BL22,P1,P2
C*****
C
C P = BED PRESSURE (N/M**2)
C PN = SUPPLY PRESURES (N/M**2)
C U = STREAM VELOCITY (M/SEC)
C W = MASS FLOWRATE (KG/SEC)

```

```

C  CVA = DISCHARGE COEFFICIENT TIMES VALVE AREA (M**2)
C  C = TOTAL GAS STREAM CONCENTRATION (KGMOLE/M**3)
C  CA = GAS A STREAM CONCENTRATION (KGMOLE/M**3)
C  Z = BED DISTANCE COORDINATE (M)
C  Y = MOLECULAR FLUX OF GAS A = CA*U (KGMOLE/M**2-SEC)
C  X = MOLE FRACTION OF GAS A
C
C.....
C  VA1=CVA(1)
C  VA2=CVA(2)
C  VA3=CVA(3)
C  BETA1=(1.-E)*D1/E
C  BETA2=(1.-E)*D2/E
C  PN1=PN(1)
C  PN2=PN(2)
C  BL21=BETA1*.5*L
C  BL22=BETA2*.5*L
C  C1=CG1
C  C2=CG2
C*****CALCULATE G AND INTEGRAL OF G
C  GI1(1)=0.
C  GI2(1)=0.
C  DO 1 I=1,LUMPS
C    CALL CAPZ(N1,G1,I)
C  1  CALL CAPZ(N2,G2,I)
C    DO 2 I=2,LUMPS
C      IM1=I-1
C      GI1(I)=GI1(IM1)+DZ*(G1(I,1)+G1(IM1,1))/2.
C  2  GI2(I)=GI2(IM1)+DZ*(G2(I,1)+G2(IM1,1))/2.
C    LUM2=(LUMPS+1)/2
C    BGI1L2=BETA1*GI1(LUM2)
C    BGI1L=BETA1*GI1(LUMPS)-BGI1L2
C    BGI2L2=BETA2*GI2(LUM2)
C    BGI2L=BETA2*GI2(LUMPS)-BGI2L2
C*****SOLVE BED/VALVE EQUATION FOR C
C  CALL VAQ(C1,C2)
C  W(1)=W1
C  W(2)=W2
C  W(3)=W3
C  CG1=C1
C  CG2=C2
C*****SOLVE FOR CONCENTRATION OF GAS A
C  CALL CONCA(C1,CA1,G1,GI1,U01,UL1,X(1),X(2),BETA1,AI/AO)
C  CALL CONCA(C2,CA2,G2,GI2,U02,UL2,X(2),X(3),BETA2,AO/AI)
C  RETURN
C  END

SUBROUTINE CONCA(CG,CA,G,GI,U0G,ULG,X0,XL,BETAG,ARG)
C*****SOLVES FOR CONCENTRATION OF GAS A
C  REAL L2,L
C  DIMENSION CA(300),G(300,2),GI(300),X(2),U(300),Y(300)
C  COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,RT,AI,AO,AMWO1,AMWL1,AMW02
C  1,AMWL2,WBR
C  COMMON/VOID/E
C  COMMON/TIME/T
C  C=CG
C  BETA=BETAG
C  UL=ULG

```

```

U0=U0G
X(1)=X0
X(2)=XL
AR=ARG
LUM2=(LUMPS+1)/2
GIL2=GI(LUM2)
L2=.5*L
IF(U0**2+UL**2)22,22,23
C*****ZERO FLOW SOLUTION
22 DO 24 I=1,LUMPS
24 CA(I)=G(I,2)
RETURN
23 CONTINUE
CA(1)=C*X(1)
U(1)=U0
Y(1)=U0*CA(1)
CA(LUMPS)=C*X(2)
Y(LUMPS)=UL*CA(LUMPS)
Z=0.
C*****ICR IS INDEX INDICATING U=0 IN BED
ICR=1
ICR1=ICR-1
C*****SOLVE FOR U IN BED
DO 3 I=2,LUMPS
IM1=I-1
Z=Z+DZ
IF(IM1-LUM2)35,34,34
35 U(I)=U(1)+BETA*(GI(I)/C-Z)
GO TO 36
34 U(I)=AR*U(LUM2)+BETA*((GI(I)-GIL2)/C-Z+L2)
36 CONTINUE
C*****CHECK SIGN CHANGE OF U
IF(U(I)*U(IM1))8,9,3
8 ICR=I
ICR1=ICR-1
C*****CALCULATE INTERIOR INITIAL FLUX (U0<0,UL>0)
ULAST=U(IM1)
IF(IM1.EQ.LUM2)ULAST=AR*ULAST
DZCR=DZ*U(I)/(U(I)-ULAST)
DTC=DZCR-DZ
R1=U(ICR)/DZCR
R3=(G(I,2)-G(IM1,2))/DZ
R2=G(IM1,2)-R3*DTC
A1=BETA*R2*R1/(BETA+R1)
A2=BETA*R3*R1/(BETA+2.*R1)
Y(ICR)=A1*DZCR+A2*DZCR*DZCR
Y(ICR1)=A1*DTC+A2*DTC*DTC
CA(ICR)=Y(ICR)/U(ICR)
CA(ICR1)=Y(ICR1)/ULAST
GO TO 3
9 ICR=I
ICR1=I
Y(ICR)=0.
IF(U0)3,25,3
25 ICR=1
ICR1=1
Y(1)=0.
3 CONTINUE

```

```

      UL=U(LUMPS)
C*****LOGIC FOR FORWARD INTEGRATION
      ISIG=1
      IF(UL)14,14,15
14   IST=1
      IF(U0)12,20,21
21   CONTINUE
      IT=ICR-2
      IF(IT)12,12,11
20   ICR=1
      ICR1=1
      GO TO 12
15   CONTINUE
      IST=ICR
      IT=LUMPS-ICR
      IF(IT)12,12,11
11   CONTINUE
C*****SOLVE FOR CA
      DO 10 J=1,IT
      I=IST+ISIG*J
      IM1=I-ISIG
      IF(IM1-LUM2)33,30,33
30   IF(ISIG)31,33,32
31   Y(IM1)=Y(IM1)/AR
      U(IM1)=U(IM1)/AR
      GO TO 33
32   U(IM1)=AR*U(IM1)
      Y(IM1)=AR*Y(IM1)
33   CONTINUE
      R0=U(IM1)
      IF(I.EQ.LUM2.AND.ISIG.EQ.-1)U(I)=AR*U(I)
      R1=(U(I)-R0)/DZ
      R2=G(IM1,2)
      R3=(G(I,2)-R2)/DZ
      IF(R1)6,7,6
7    A0=(R2-R3*R0/BETA)*R0
      A1=R3*R0
      BR=BETA/R0
      Y(I)=(Y(IM1)-A0)*EXP(-BR*DZ)+A0+A1*DZ
      GO TO 10
6    A2=BETA*R3*R1/(BETA+2.*R1)
      A1=(BETA*(R3*R0+R2*R1)-2.*A2*R0)/(BETA+R1)
      A0=(BETA*R0*R2-A1*R0)/BETA
      IF(R0)26,27,26
27   Y(I)=A1*DZ+A2*DZ*DZ
      GO TO 28
26   CONTINUE
      BR=BETA/R1
      Y(I)=(Y(IM1)-A0)*(1.+R1*DZ/R0)**(-BR)+A0+A1*DZ+A2*DZ*DZ
28   CONTINUE
      IF(U(I))18,19,18
19   CA(I)=C*G(I,2)/G(I,1)
      GO TO 10
18   CA(I)=Y(I)/U(I)

10   CONTINUE
      IF(ICR.EQ.1)GO TO 13
      IF(DZ)13,13,12

```

```

12 CONTINUE
C*****LOGIC FOR BACKWARD INTEGRATION
DZ=-DZ
ISIG=-1
IF(UL)16,16,17
16 IST=LUMPS
IT=LUMPS-ICR
IF(ICR.EQ.ICR1)IT=IT-1
IF(IT)13,13,11
17 CONTINUE
IST=ICR1
IT=ICR1-1
IF(IT)13,13,11
13 CONTINUE
X(1)=CA(1)/C
X(2)=CA(LUMPS)/C
DZ=FLOAT(ISIG)*DZ
X0=X(1)
XL=X(2)
RETURN
END
SUBROUTINE CAPZ(N,G,I)
REAL KA,KB,N,NB
DIMENSION N(300,2),G(300,2)
COMMON/CAP/KA,KB,B,D,TEMP1
NB=N(I,1)-N(I,2)
GB=KB*NB/(1.-NB/B)
G(I,2)=KA*N(I,2)
G(I,1)=G(I,2)+GB
RETURN
END
SUBROUTINE VALVE(A,PU,PD,W)
C*****CALCULATES VALVE MASS FLOWRATE KG/SEC
REAL KI
DATA KI,C1,C2,CK/.714,.00906,.00234,.286/
PUL=PU
PDL=PD
PR=PDL/PUL
SIG=1.
IF(1.-PR)4,3,2
3 W=0.
RETURN
4 PR=1./PR
PUL=PDL
SIG=-1.
2 CONTINUE
IF(PR.LT..528)GO TO 1
PRR=1.-PR**CK
W=SIG*A*C1*PUL*(PR**KI)*SQRT(PRR)
RETURN
1 W=SIG*A*C2*PUL
RETURN
END
SUBROUTINE VAQ(CG1,CG2)
C*****ITERATION AND FALSE POSITION TO SOLVE VALVE/BED EQUATIONS
EXTERNAL F1,F2
COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,RT,AI,AO,AMW01,AMWL1,AMW02
1,AMWL2,WBR

```



```

COMMON/VBEQ/W1,W2,W3,U01,UL1,U02,UL2,PN1,PN2,VA1,VA2,VA3
1 ,BGI1L,BGI1L2,BGI2L,BGI2L2,BL21,BL22,P1,P2
C1=CG1
C2=CG2
P1=C1*RT
PO1=P1
J=1
10 FC2=F2(C2)
1 CN2=C2*(1.-FC2)
IF(ABS(CN2-C2)/C2.LE.1.E-6)GO TO 2
FCN2=F2(CN2)
IF(FC2*FCN2)4,3,3
3 C2=CN2
FC2=FCN2
GO TO 1
4 C2=FP(F2,FCN2,CN2,FC2,C2)
2 CONTINUE
FC1=F1(C1)
5 CN1=C1*(1.-FC1)
IF(ABS(CN1-C1)/C1.LE.1.E-6)GO TO 6
FCN1=F1(CN1)
IF(FC1*FCN1)8,7,7
7 C1=CN1
FC1=FCN1
GO TO 5
8 C1=FP(F1,FCN1,CN1,FC1,C1)
6 IF(ABS(P1-PO1)/P1.LE.1.E-6)GO TO 9
J=J+1
PO1=P1
IF(J.LE.20)GO TO 10
PRINT 100
100 FORMAT(1X,"FAILED TO CONVERGE")
9 CG1=C1
CG2=C2
RETURN
END
FUNCTION F1(CG1)
C*****BED 1 VALVE/BED EQUATION
COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,RT,AI,AO,AMW01,AMWL1,AMW02
1 ,AMWL2,WBR
COMMON/VBEQ/W1,W2,W3,U01,UL1,U02,UL2,PN1,PN2,VA1,VA2,VA3
1 ,BGI1L,BGI1L2,BGI2L,BGI2L2,BL21,BL22,P1,P2
C1=CG1
P1=C1*RT
CALL VALVE(VA1,PN1,P1,W1)
CALL VALVE(VA2,P1,P2,W2)
IF(P1-P2)1,2,3
1 WBR1=0.
GO TO 4
2 WBR1=.5*WBR
GO TO 4
3 WBR1=WBR
4 CONTINUE
U01=W1/(C1*AMW01*AI)
UL1=(W2+WBR1)/(C1*AMWL1*AO)
AR=AI/AO
F1=1.-(AR*BGI1L2+BGI1L)/((UL1+BL21-AR*(U01-BL21))*C1)
RETURN

```

```

END
FUNCTION F2(CG2)
C*****BED 2 VALVE/BED EQUATION
COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,RT,AI,AO,AMW01,AMWL1,AMW02
1,AMWL2,WBR
COMMON/VBEQ/W1,W2,W3,U01,UL1,U02,UL2,PN1,PN2,VA1,VA2,VA3
1,BGI1L,BGI1L2,BGI2L,BGI2L2,BL21,BL22,P1,P2
C2=CG2
P2=C2*RT
CALL VALVE(VA2,P1,P2,W2)
CALL VALVE(VA3,P2,PN2,W3)
IF(P2-P1)1,2,3
1 WBR2=0.
GO TO 4
2 WBR2=.5*WBR
GO TO 4
3 WBR2=WBR
4 CONTINUE
U02=(W2-WBR2)/(C2*AMW02*AO)
UL2=W3/(C2*AMWL2*AI)
AR=AO/AI
F2=1.-(AR*BGI2L2+BGI2L)/((UL2+BL22-AR*(U02-BL22))*C2)
RETURN
END
FUNCTION FP(F,FXR,XR,FXL,XL)
C*****USES FALSE POSITION TO FIND ZERO
DATA EPS/1.E-6/
I=0
3 X=XL-FXL*(XL-XR)/(FXL-FXR)
FX=F(X)
I=I+1
IF(ABS(FX).LT.EPS)GO TO 4
IF(I.GT.50)GO TO 4
IF(FX*FXL)1,1,2
1 XR=X
FXL=FXL*FXR/(FXR+FX)
FXR=FX
GO TO 3
2 XL=X
FXR=FXR*FXL/(FXL+FX)
FXL=FX
GO TO 3
4 FP=X
RETURN
END
SUBROUTINE SCALE1(Y,YABS)
COMMON/NBED/NBED
INTEGER YABS
DY = 1.0 - Y
DPY = DY * 150.0
IDY = IFIX ( DPY )
YABS = IDY + 100
RETURN
END
SUBROUTINE SCALE2(Y,YABS)
COMMON/NBED/NBED
INTEGER YABS
DY = 1.0 - Y

```

```

DPY = DY * 150.0
IDY = IFIX ( DPY )
YABS = IDY + 300
RETURN
END
SUBROUTINE INITIAL1 ( YABS1, YABS2, NBED )
INTEGER YABS1, YABS2
IF ( NBED .EQ. 0 ) THEN
WRITE(9,100) YABS1
ELSE
WRITE(9,100) YABS2
END IF
100 FORMAT (3X, ' P[275,'I3']')
RETURN
END
SUBROUTINE INITIAL2 ( YABS1, YABS2, NBED )
INTEGER YABS1, YABS2
IF ( NBED .EQ. 0 ) THEN
WRITE(9,100) YABS2
ELSE
WRITE(9,100) YABS1
END IF
100 FORMAT (3X, ' P[575,'I3']')
RETURN
END
SUBROUTINE DRAWF ( XABS, YABS, NM )
C DRAW VECTORS FROM PREVIOUS CURSOR POSITIONS TO CURRENT POINTS
C FORWARD PLOTTING
INTEGER XABS, YABS
WRITE(9,100) NM, XABS, YABS
100 FORMAT (3X, ' W(R,N'I1')V[]V['I3', 'I3']')
RETURN
END
SUBROUTINE DRAWB ( XABS, YABS, NM )
C BACKWARD PLOTTING
INTEGER XABS, YABS
WRITE(9,100) NM, XABS, YABS
100 FORMAT (3X, ' W(R,N'I1')V[]V['I3', 'I3']')
RETURN
END
SUBROUTINE TXTERASE (NTICKS)
C ERASE THE VT125 TEXT SCREEN
LOGICAL ERASE
DIMENSION ERASE(4)
DATA ERASE / 27, 'I', '2', 'J' /
IF ( NTICKS.NE.0 ) THEN
DO I=1,7
CALL DELAY (NTICKS)
END DO
ELSE
END IF
TYPE 10, ERASE
10 FORMAT ( '+', 5A1, '$')
RETURN
END
SUBROUTINE REGISTART
C SETS VT125 INTO GRAPHICS(REGIS) MODE
LOGICAL GRAPHIC

```

```

    DIMENSION GRAPHIC(3)
    DATA GRAPHIC / 27, 'P', 'p' /
    TYPE 10, GRAPHIC
10  FORMAT ( '+', 5A1, $ )
    RETURN
    END
    SUBROUTINE PLTERASE
C  ERASES PLOTS
    LOGICAL ERASE
    DIMENSION ERASE(4)
    DATA ERASE / 'S', 'I', 'E', 'J' /
    TYPE 10, ERASE
10  FORMAT ( '+', 5A1, $ )
    RETURN
    END
    SUBROUTINE FRAME1
    WRITE(9,110)
110  FORMAT ( ' P[250,5]T(S2)"OBOGS SIMULATION"' )
    WRITE(9,112)
112  FORMAT ( ' P[30,30]T(S1)"TIME = SEC. "' )
    WRITE(9,114)
114  FORMAT ( ' P[260,95]T"1"P[260,245]T"0"' )
    WRITE(9,116)
116  FORMAT ( ' P[260,295]T"1"P[260,445]T"0"' )
    WRITE(9,120)
120  FORMAT ( ' P[400,80]T"BED #1"' )
    WRITE(9,130)
130  FORMAT ( ' P[400,280]T"BED #2"' )
    WRITE(9,140)
140  FORMAT ( ' P[20,255]T"INLET"' )
    WRITE(9,150)
150  FORMAT ( ' P[5,275]T"PRESSURE"' )
    WRITE(9,160)
160  FORMAT ( ' P[195,112]T"R.V."' )
    WRITE(9,170)
170  FORMAT ( ' P[95,430]T"EXHAUST"' )
    WRITE(9,180)
180  FORMAT ( ' P[115,450]T"GAS"' )
    WRITE(9,190)
190  FORMAT ( ' P[624,268]T"P.O."' )
    WRITE(9,192)
192  FORMAT ( ' P[680,112]T"B.P."' )
    WRITE(9,193)
193  FORMAT ( ' P[720,412]V000000P[730,403]T"0%"' )
    WRITE(9,195)
195  FORMAT ( ' P[720,356]V000000P[730,349]T"20%"' )
    WRITE(9,196)
196  FORMAT ( ' P[720,132]V000000P[730,123]T"100%"' )
    WRITE(9,200)
200  FORMAT ( ' P[275,100]V[575,100]V[575,250]V[275,250]V[275,100]' )
    WRITE(9,210)
210  FORMAT ( ' P[275,300]V[575,300]V[575,450]V[275,450]V[275,300]' )
    WRITE(9,300)
300  FORMAT ( ' P[575,162]V[670,162]V[670,132]V[720,132]V[720,412]' )
    WRITE(9,310)
310  FORMAT ( ' V[670,412]V[670,382]V[575,382]P[575,182]' )
    WRITE(9,400)
400  FORMAT ( ' V[600,182]V[600,267]V[605,272]V[600,277]V[600,362]' )

```

```

WRITE(9,410)
410 FORMAT ( ' V[575,362]P[620,182]V[670,182]V[670,362]V[620,362]' )
WRITE(9,450)
450 FORMAT ( ' V[620,277]V[615,272]V[620,267]V[620,182]' )
WRITE(9,460)
460 FORMAT ( ' P[720,262]V[750,262]V[750,282]V[720,282]' )
WRITE(9,500)
500 FORMAT ( ' P[275,162]V[240,162]V[240,132]V[180,132]' )
WRITE(9,510)
510 FORMAT ( ' V[180,162]V[118,162]V[118,262]' )
WRITE(9,520)
520 FORMAT ( ' P[275,182]V[240,182]P[180,182]V[138,182]V[138,262]
1 ' )
WRITE(9,540)
540 FORMAT ( ' P[118,282]V[118,420]V[138,420]V[138,382]' )
WRITE(9,600)
600 FORMAT ( ' V[180,382]V[180,412]V[240,412]V[240,382]V[275,382]' )
WRITE(9,610)
610 FORMAT ( ' P[180,262]V[80,262]V[80,282]' )
WRITE(9,700)
700 FORMAT ( ' V[180,282]V[180,362]V[138,362]V[138,282]P[275,362]' )
WRITE(9,710)
710 FORMAT ( ' P[275,362]V[240,362]V[240,182]P[180,182]V[180,262]' )
WRITE(9,730)
730 FORMAT ( ' P[630,162]V[630,167]V[640,162]' )
WRITE(9,735)
735 FORMAT ( ' P[630,182]V[630,177]V[640,182]' )
WRITE(9,740)
740 FORMAT ( ' P[630,362]V[630,367]V[640,362]' )
WRITE(9,750)
750 FORMAT ( ' P[630,382]V[630,377]V[640,382]' )
RETURN
END
SUBROUTINE NEWTIME ( TIME,NW )
WRITE(9,100) NW,TIME
100 FORMAT ( ' W(N'I1')P[110,30]T(S1)"F4.1"' )
RETURN
END
SUBROUTINE FRAME2 ( NBED,NW )
IF ( NBED.EQ. 1 ) THEN
WRITE(9,710)NW
710 FORMAT ( ' W(N'I1')P[180,262]V[240,362]' )
ELSE
WRITE(9,720)NW
720 FORMAT ( ' W(N'I1')P[180,262]V[240,162]' )
END IF
WRITE(9,730)NW
730 FORMAT ( ' W(N'I1')P[630,162]V[630,167]V[640,162]' )
WRITE(9,735)
735 FORMAT ( ' P[630,182]V[630,177]V[640,182]' )
WRITE(9,740)NW
740 FORMAT ( ' W(N'I1')P[630,362]V[630,367]V[640,362]' )
WRITE(9,750)
750 FORMAT ( ' P[630,382]V[630,377]V[640,382]' )
IF ( NBED.EQ. 1 ) THEN
WRITE(9,800)NW
800 FORMAT ( ' W(N'I1')P[180,282]V[240,382]' )
WRITE(9,805)

```

```

805 FORMAT ( ' P[240,162]V[180,162]P[240,182]V[180,182]' )
    WRITE(9,810)NW
810 FORMAT ( ' W(N'I1')P[113,272]V[143,272]' )
    WRITE(9,815)
815 FORMAT ( ' P[136,268]V[]V[143,272]V[136,276]' )
    WRITE(9,820)NW
820 FORMAT ( ' W(N'I1')P[590,372]V[625,372]' )
    WRITE(9,825)
825 FORMAT ( ' P[620,369]V[]V[625,372]V[620,375]' )
    WRITE(9,830)NW
830 FORMAT ( ' W(N'I1')P[610,320]V[610,290]' )
    WRITE(9,835)
835 FORMAT ( ' P[607,295]V[]V[610,290]V[613,295]' )
    WRITE(9,840)NW
840 FORMAT ( ' W(N'I1')P[610,172]V[585,172]' )
    WRITE(9,845)
845 FORMAT ( ' P[590,169]V[]V[585,172]V[590,175]' )
    WRITE(9,850)NW
850 FORMAT ( ' W(N'I1')P[128,352]V[128,392]' )
    WRITE(9,855)
855 FORMAT ( ' P[124,385]V[]V[128,392]V[132,385]' )
    WRITE(9,860)NW
860 FORMAT ( ' W(N'I1')P[638,172]C[632,166]' )
    WRITE(9,865)
865 FORMAT ( ' P[645,172]V11117777111177771111000' )
    WRITE(9,870)NW
870 FORMAT ( ' W(N'I1')P[650,372]C[643,372]' )
    WRITE(9,880)
880 FORMAT ( ' P[657,372]V11117777111100' )
    ELSE
    WRITE(9,900)NW
900 FORMAT ( ' W(N'I1')P[180,282]V[240,182]' )
    WRITE(9,905)
905 FORMAT ( ' P[180,362]V[240,362]P[180,382]V[240,382]' )
    WRITE(9,910)NW
910 FORMAT ( ' W(N'I1')P[113,272]V[143,272]' )
    WRITE(9,915)
915 FORMAT ( ' P[136,268]V[]V[143,272]V[136,276]' )
    WRITE(9,920)NW
920 FORMAT ( ' W(N'I1')P[590,172]V[615,172]' )
    WRITE(9,925)
925 FORMAT ( ' P[608,168]V[]V[615,172]V[608,176]' )
    WRITE(9,930)NW
930 FORMAT ( ' W(N'I1')P[610,222]V[610,252]' )
    WRITE(9,935)
935 FORMAT ( ' P[606,245]V[]V[610,252]V[614,245]' )
    WRITE(9,940)NW
940 FORMAT ( ' W(N'I1')P[610,372]V[580,372]' )
    WRITE(9,945)
945 FORMAT ( ' P[587,368]V[]V[580,372]V[587,376]' )
    WRITE(9,950)NW
950 FORMAT ( ' W(N'I1')P[159,372]V[128,372]V[128,390]' )
    WRITE(9,955)
955 FORMAT ( ' P[124,381]V[]V[128,390]V[132,381]' )
    WRITE(9,960)NW
960 FORMAT ( ' W(N'I1')P[650,172]C[643,172]' )
    WRITE(9,965)
965 FORMAT ( ' P[657,172]V11117777111100' )

```

```

WRITE(9,970)NW
970 FORMAT ( ' W(N'11')P[638,372]C[633,366]' )
WRITE(9,980)
980 FORMAT ( ' P[645,372]V11117777111177770000' )
END IF
RETURN
END
SUBROUTINE REGISOUT
C SETS VT125 BACK INTO TEXT MODE
LOGICAL ALPHA
DIMENSION ALPHA(2)
DATA ALPHA / 27, 'V' /
TYPE 10, ALPHA
10 FORMAT ( '+', 5A1, $ )
RETURN
END
SUBROUTINE DELAY ( NTICKS )
TYPE 100, NTICKS
100 FORMAT ( '+S(T<'13'>)', $ )
RETURN
END
SUBROUTINE SCALEM ( Y, IYABS, VV, RANGE )
DY = (( VV - Y ) * 300.0 ) / RANGE
IDY = IFIX ( DY )
IYABS = IDY + 90
RETURN
END
SUBROUTINE STIME ( TIME, IXABS, RANGE, AXLEN, SHIFT )
INTEGER SHIFT
DX = ( TIME * AXLEN ) / RANGE
IDX = IFIX ( DX )
IXABS = IDX + SHIFT
RETURN
END
SUBROUTINE FRAME
COMMON/SYST/DT, PSUP, POUT, TF, WBRL, VS, VO, IIN, TCYC
COMMON/NPRINT/NOUT1
DIMENSION T(10)
DELT = TF/10.0
DO I=1, 10
T(I)=I*DELT
END DO
TYPE 10
10 FORMAT ( 'IP[130,390]V[+60,]V[,+5]' , $ )
DO I=1, 9
TYPE 12
12 FORMAT ( 'IP[, -5]V[+60,]V[,+5]' , $ )
END DO
TYPE 20
20 FORMAT ( 'IP[120,398]T"0.0"' , $ )
TYPE 21, T(1), T(2)
21 FORMAT ( 'IP[160,398]T""F5.1""P[220,398]T""F5.1""' , $ )
TYPE 22, T(3), T(4)
22 FORMAT ( 'IP[280,398]T""F5.1""P[340,398]T""F5.1""' , $ )
TYPE 23, T(5), T(6)
23 FORMAT ( 'IP[400,398]T""F5.1""P[460,398]T""F5.1""' , $ )
TYPE 24, T(7), T(8)
24 FORMAT ( 'IP[520,398]T""F5.1""P[580,398]T""F5.1""' , $ )

```

```

TYPE 25 , T(9),T(10)
25  FORMAT ( 'IP[640,398]T"F5.1"P[700,398]T"F5.1" , $ )
TYPE 30
30  FORMAT ( 'IP[350,430]T"TIME (SEC.)" , $ )
TYPE 40
40  FORMAT ( 'IP[130,390]V[,-30]V[-5.]' , $ )
DO I=1,9
TYPE 50
50  FORMAT ( 'IP[+5.]V[,-30]V[-5.]' , $ )
END DO
TYPE 51
51  FORMAT ( 'IP[70,180]T[+0,+16]"OUTPUT" , $ )
TYPE 53 , PSUP
53  FORMAT ( 'IP[530,34]T[+9,+0]"PSUP = "T"F5.2"T" PSIA" , $ )
TYPE 54 , POUT
54  FORMAT ( 'IP[530,52]T"POUT = "T"F5.2"T" PSIA" , $ )
TYPE 55 , WBRL
55  FORMAT ( 'IP[530,70]T"WBRL = "T"F5.2"T" STD LIT/MIN" , $ )
IF(NOUT1.EQ.1)THEN
57  TYPE 60
60  FORMAT ( 'IP[290,5]T" OXYGEN MOLE FRACTION "' , $ )
TYPE 62
62  FORMAT ( 'IP[95,383]T"0.0"P[95,353]T"0.1"P[95,323]T"0.2" , $ )
TYPE 64
64  FORMAT ( 'IP[95,293]T"0.3"P[95,263]T"0.4"P[95,233]T"0.5" , $ )
TYPE 66
66  FORMAT ( 'IP[95,203]T"0.6"P[95,173]T"0.7"P[95,143]T"0.8" , $ )
TYPE 68
68  FORMAT ( 'IP[95,113]T"0.9"P[95,83]T"1.0" , $ )
ELSE IF(NOUT1.EQ.2) THEN
TYPE 80
80  FORMAT ( 'IP[290,5]T" INLET MASS FLOWRATE (KG/SEC) "' , $ )
TYPE 82
82  FORMAT ( 'IP[85,383]T".02"P[85,353]T".01"P[95,323]T"0.0" , $ )
TYPE 84
84  FORMAT ( 'IP[95,293]T".01"P[95,263]T".02"P[95,233]T".03" , $ )
TYPE 86
86  FORMAT ( 'IP[95,203]T".04"P[95,173]T".05"P[95,143]T".06" , $ )
TYPE 88
88  FORMAT ( 'IP[95,113]T".07"P[95,83]T".03" , $ )
ELSE IF(NOUT1.EQ.3) THEN
TYPE 90
90  FORMAT ( 'IP[290,5]T" OUTLET MASS FLOWRATE (KG/SEC) "' , $ )
TYPE 82
TYPE 84
TYPE 86
TYPE 88
ELSE
GO TO 57
END IF
RETURN
END

```


APPENDIX B: OBOGS Model Code, Three Component Model

```

PROGRAM MSEV
C
C OBOGS WITH CONTAMINANT
C A = COMPONENT A (OXYGEN)
C B = COMPONENT B (CONTAMINANT)
C C = COMPONENT C (NITROGEN)
C
  REAL LIN,KA,KB,KC
  REAL L50
  BYTE CONTINUE
  BYTE LOWC
  BYTE KEYIN
  BYTE NULL
  BYTE YES
  BYTE NO
  CHARACTER*64 TEMP
  DIMENSION W(3),XA(3),XB(3),CVA(3),PS(2),P(2)
  DIMENSION WV(502,2),TV(500),XV(502,3)
  DIMENSION DL(500),CA1(500),CA2(500),CB1(500),CB2(500),
1    XA1(502,11),XA2(502,11),XB1(502,11),XB2(502,11)
  DIMENSION IXABS1(501,11),IYABS1(501,11),IXABS2(501,11),
1    IYABS2(501,11)
  DIMENSION IX1(501),IY1(501),IX2(501),IY2(501)
  DIMENSION JX(6,11),JY(6,11),JXERSH(6,11),JYERSH(6,11),
1    JXERSL(6,11),JYERSL(6,11)
  DIMENSION TI(6),NY(6)
  DIMENSION XCA1(502,11),XCA2(502,11),XCB1(502,11),
1    XCB2(502,11),XBRO2(500)
  dimension range(4),xaxis(10),yaxis(10),range2(4),xaxis2(10),
1    yaxis2(10),xaxis3(10),yaxis3(10),range3(4),
1    xaxis50(10),yaxis50(10),range50(4)
  DIMENSION L50(402),T50(400),ERR(400)
  COMMON/TIME/T
  COMMON/SYST/DT,PSUP,POUT,TF,WBRL,VS,VO,IIN,TCYC
  COMMON/GEOM/DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN
  COMMON/CAP/KA,KB,KC,BB,BC,D
  COMMON/VOID/E
  COMMON/NFINISH/NEXIT
  COMMON/STEP/NSTEP,TSTEP1,TSTEP2,BRSTEP
  COMMON/NPRINT/NOUT1,TI,NY
  COMMON/BEDV/LUMPS,DZ
  COMMON/HILOBED/NBED,NB
  DATA CONTINUE/67/
  DATA LOWC/99/
  DATA NULL/0/
  DATA YES/89/
  DATA NO/78/
17 CALL MENU
  IF ( LUMPS.EQ. 101 ) THEN
    IINC = 3
  ELSE
    IINC = 6
  END IF
  IF ( NEXIT.EQ. 1 ) GOTO 900
  IF ( NSTEP.EQ. 1 ) GOTO 20
  TSTEP1 = 10000000.
  TSTEP2 = 10000000.

```

```

BRSTEP = WBRL
20 CONTINUE
DLIN = LIN / (LUMPS-1)
DO I = 1, LUMPS
  DL(I) = (I-1) * DLIN
END DO
OPEN (UNIT=1, FILE='MOLEFRAC', STATUS='NEW')
OPEN (UNIT=2, FILE='FLOWRATE', STATUS='NEW')
OPEN (UNIT=3, STATUS='SCRATCH')
OPEN (UNIT=4, FILE='PROFILE', STATUS='NEW')
OPEN (UNIT=8, STATUS='SCRATCH')
OPEN (UNIT=9, FILE='REGIS', STATUS='NEW')
OPEN (UNIT=10, FILE='CONTA', STATUS='NEW')
WRITE(6,7)
7 FORMAT(//,
1 .....),
PRINT *
PRINT *, 'OBOGS SIMULATION'
PRINT *
IIN = 0
ND = 0
N50 = 0
CDBY = .056 / DBYIN
IF (CDBY.GT.1.) CDBY = 1.
IF (CDBY.LT..6) CDBY = .6
DBYM = DBYIN * .0254
PRINT *, 'BYPASS VALVE DISCHARGE COEFFICIENT = ', CDBY
BYVA = 3.14159 * DBYM ** 2 / 4.
VS = .8 * (.0254 * DVSIN) ** 2 * 3.14159 / 4.
VO = .8 * (.0254 * DVOIN) ** 2 * 3.14159 / 4.
PRINT *, 'AREA OF BY-PASS (M**2) = ', BYVA
PRINT *, 'CD*AREA SUPPLY VALVE (M**2) = ', VS
PRINT *, 'CD*AREA OUTLET VALVE (M**2) = ', VO
PRINT *, 'SUPPLY PRESSURE(PSI)=', PSUP,
      'OUTLET PRESSURE(PSI)=', POUT
CVA(2) = CDBY * BYVA
T = 0.0
DT = .01
DATAP = 200.
INC = IFIX ( TF / (DATAP * DT) )
NT = 0
IM = 0
P(1) = 14.5
P(2) = 14.5
XA(2) = .20
XB(2) = 0.
PRINT *, 'BREATHING FLOW (STD LIT/MIN) = ', WBRL
XABR = .20
XBBR = 0.
IF(NOUT1-1)35,35,36
35 CLOSE(UNIT=2, STATUS='DELETE')
CLOSE(UNIT=4, STATUS='DELETE')
CLOSE(UNIT=9, STATUS='DELETE')
CLOSE(UNIT=10, STATUS='DELETE')
WRITE(1,57)
57 FORMAT(' OXYGEN MOLE FRACTION',24X,'TIME')
GO TO 1
36 IF(NOUT1.EQ.4) GOTO 37

```

```

IF(NOUT1.EQ.5) GOTO 170
CLOSE(UNIT=1,STATUS='DELETE')
CLOSE(UNIT=4,STATUS='DELETE')
CLOSE(UNIT=9,STATUS='DELETE')
CLOSE(UNIT=10,STATUS='DELETE')
WRITE(2,58)
58 FORMAT(' INLET MASS FLOWRATE',T25,' OUTLET MASS FLOWRATE',
1' T55,TIME')
GO TO 1
170 CLOSE(UNIT=1,STATUS='DELETE')
CLOSE(UNIT=2,STATUS='DELETE')
CLOSE(UNIT=4,STATUS='DELETE')
CALL FRAME1
GO TO 1
37 CLOSE(UNIT=1,STATUS='DELETE')
CLOSE(UNIT=2,STATUS='DELETE')
CLOSE(UNIT=10,STATUS='DELETE')
CALL FRAME1
1 CONTINUE
XA(1) = .21
XA(3) = .21
XB(1) = .21
XB(3) = .21
CALL TIMEF ( CVA,PH,PS,T )
CALL BEDS ( P, W, XA, XB, XABR, XBBR, CVA, PS, DT,
CA1, CA2, CB1, CB2, C1, C2 )
IF ( NOUT1 .EQ. 4 .OR. NOUT1 .EQ. 5 ) GO TO 77
GO TO 78
77 DO 1000 K=1,6
IF ( NT .EQ. NY(K) ) GO TO 150
GO TO 1000
150 CONTINUE
IF ( K.GT. 1 ) THEN

CALL INITIAL1 ( IY1(1), IY2(1), NB )
DO J=2,LUMPS
DLIN = LIN / (LUMPS-1)
DL(J) = (J-1) * DLIN
IF ( NB .EQ. 0 ) THEN
CALL DRAWF ( IX1(J), IY1(J), 1 )
ELSE
CALL DRAWF ( IX2(J), IY2(J), 1 )
END IF
END DO
CALL INITIAL2 ( IY1(1), IY2(1), NB )
DO M=2,LUMPS
IF ( NB .EQ. 0 ) THEN
CALL DRAWB ( IX2(M), IY2(M), 1 )
ELSE
CALL DRAWB ( IX1(M), IY1(M), 1 )
END IF
END DO
ELSE
END IF

DO J=2,LUMPS
DLIN = LIN / (LUMPS-1)
DL(J) = (J-1) * DLIN

```

```

END DO
IF ( TTI .GT. 0.0 ) THEN
CALL NEWTIME ( TTI,1 )
CALL NEWTIME ( TI(K),0 )
IF ( NBED .EQ. NB ) THEN
CONTINUE
ELSE
CALL FRAME2 ( NB,1 )
CALL FRAME2 ( NBED,0 )
END IF
ELSE
CALL NEWTIME ( TI(K),0 )
CALL FRAME2 ( NBED,0 )
END IF
TTI = TI(K)
IF (NOUT1 .EQ. 4) GO TO 120
WRITE(10,*) PH,NBED,T,NT,NY(K)
WRITE(10,110) TI(K)
GO TO 130
120 WRITE(4,110) TI(K)
110 FORMAT(' BED COORDINATE',T20,'MOLE FRACTION IN BED#1',
1T46,'MOLE FRACTION IN BED#2',3X,'(TIME T= ',F6.2,'SEC.),' /)
130 CONTINUE
DO L=1,LUMPS
XCA1(L,K) = CA1(L)
XCA2(L,K) = CA2(L)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
XCB1(L,K) = CB1(L)
XCB2(L,K) = CB2(L)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
IF ( NBED .EQ. 0 ) THEN
XA1(L,K) = XCA1(L,K) / C1
XA2(L,K) = XCA2(L,K) / C2
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
XB1(L,K) = XCB1(L,K) / C1
XB2(L,K) = XCB2(L,K) / C2
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
IXABS1(L,K) = 275 + IINC*(L-1)
IXABS2(L,K) = 575 - IINC*(L-1)
IF ( NOUT1 .EQ. 4 ) THEN
CALL SCALE1 ( XA1(L,K), IYABS1(L,K) )
CALL SCALE2 ( XA2(L,K), IYABS2(L,K) )
ELSE
CALL SCALE1 ( XB1(L,K), IYABS1(L,K) )
CALL SCALE2 ( XB2(L,K), IYABS2(L,K) )
END IF
ELSE
M = LUMPS+1-L
XA1(M,K) = XCA1(L,K) / C1
XA2(M,K) = XCA2(L,K) / C2

XB1(M,K) = XCB1(L,K) / C1
XB2(M,K) = XCB2(L,K) / C2

IXABS2(L,K) = 275 + IINC*(L-1)
IXABS1(L,K) = 575 - IINC*(L-1)
IF ( NOUT1 .EQ. 4 ) THEN
CALL SCALE1 ( XA1(M,K), IYABS1(M,K) )

```

```

      CALL SCALE2 ( XA2(M,K), IYABS2(M,K) )
    ELSE
      CALL SCALE1 ( XB1(M,K), IYABS1(M,K) )
      CALL SCALE2 ( XB2(M,K), IYABS2(M,K) )
    END IF
  END IF
END DO

  CALL INITIAL1 ( IYABS1(1,K), IYABS2(1,K), NBED )
  DO I=2,LUMPS
    IF ( NBED .EQ. 0 ) THEN
      CALL DRAWF ( IXABS1(I,K), IYABS1(I,K), 0 )
      IX1(I) = IXABS1(I,K)
      IY1(I) = IYABS1(I,K)
    ELSE
      CALL DRAWF ( IXABS2(I,K), IYABS2(I,K), 0 )
      IX2(I) = IXABS2(I,K)
      IY2(I) = IYABS2(I,K)
    END IF
  END DO
  DO I=1,LUMPS
    IF (NOUT1 .EQ. 4) GO TO 146
    GO TO 140
146  WRITE(4,300) DL(I), XA1(I,K), XA2(I,K)
    GO TO 160
140  WRITE(10,300)DL(I), XB1(I,K), XB2(I,K)
300  FORMAT ( 5X, F6.3, 14X, F10.6, 16X, F10.6 )
160  CONTINUE
  END DO

  CALL INITIAL2 ( IYABS1(1,K), IYABS2(1,K), NBED )
  DO 255 N=2,LUMPS
    IF ( NBED .EQ. 0 ) THEN
      CALL DRAWB ( IXABS2(N,K), IYABS2(N,K), 0 )
      IX2(N) = IXABS2(N,K)
      IY2(N) = IYABS2(N,K)
    ELSE
      CALL DRAWB ( IXABS1(N,K), IYABS1(N,K), 0 )
      IX1(N) = IXABS1(N,K)
      IY1(N) = IYABS1(N,K)
    END IF
    NB = NBED
255  CONTINUE

  XBRO22 = XABR / 1.05
  DY = XBRO22
  DPY = DY * 280.0
  IDY = IFIX ( DPY )
  IABS = 412 - IDY
  IF ( K.GT. 1 ) THEN
    WRITE(9,500) IA
500  FORMAT ( ' P[695,'I3'] )
    IF(IABS.LT.IA) THEN
      ID = IA - IABS
      WRITE(9,510) ID
510  FORMAT ( ' W(R,N0)V[V[695,'I3'] )
    ELSE
      ID = IABS - IA

```

```

WRITE(9,520) ID
520 FORMAT (' W(R,N1)V[V[695,+'I3']')
END IF
ELSE
WRITE(9,530) IABS
530 FORMAT (' P[695,412]V[,I3']')
END IF
WRITE(9,535)
535 FORMAT (' W(N0)')
C WRITE(6,200)DL(I),XA1(I,K),XA2(I,K)
C 200 FORMAT(8X,F6.3,11X,E10.4,7X,E10.4)

range(1) = 0.
range(2) = 18.
range(3) = 0.
range(4) = 1.
nxaxis = 20
encode (nxaxis,258,xaxis)
258 format ('BED COORDINATE (IN.)')
nyaxis = 13
encode (nyaxis,369,yaxis)
369 format ('MOLE FRACTION')
nxaxis50 = 11
encode (nxaxis50,256,xaxis50)
256 format ('TIME (SEC.)')
nyaxis50 = 9
encode (nyaxis50,366,yaxis50)
366 format ('L50 (IN.)')
IA = IABS
1000 CONTINUE
IF(((NOUT1.EQ.5).AND.(MOD(NT,INC).EQ.0)).OR.((NOUT1.EQ.5)
.AND.(AMOD(T,TCYC).EQ.0.))) THEN
N50 = N50 + 1
DO M=1,LUMPS
IF (NBED.EQ. 0) THEN
ERR(M)=(CB1(M)/C1)-.10
ELSE
L = LUMPS+1-M
ERR(M) = (CB2(L)/C2) - .10
END IF
IF (ABS(ERR(M)) .LE. 0.015) THEN
L50(N50) = DL(M)
ELSE
END IF
END DO
T50(N50) = T
WRITE(11,*) T50(N50),L50(N50),NBED,LUMPS
ELSE
END IF
IF (NT.EQ. NY(6)) THEN
RANGE50(1) = 0.
RANGE50(2) = 0.
RANGE50(3) = 0.
RANGE50(4) = 0.
CALL ZETAPLT(11,1,N50,RANGE50,T50,L50,N50+2,XAXIS50,
NXAXIS50,,YAXIS50,NYAXIS50)
call zetapit(10,6,lumps,range,DL,XB1,LUMPS+2,xaxis,
nxaxis,yaxis,nyaxis)

```

```

      call zetapt(20,6,lumps,range,DL,XB2,LUMPS+2,xaxis,
        nxaxis,yaxis,nyaxis)

      CLOSE ( UNIT=9, STATUS='SAVE' )
      REWIND 9
      OPEN ( UNIT=9, FILE='REGIS', STATUS='OLD' )
      CALL TXTERASE (0)
      CALL REGISTART
      CALL PLTERASE
9200  READ ( 9, 9220, END=9999 ) TEMP
      WRITE (3,*) TEMP
      TYPE 9220,TEMP
9220  FORMAT ( A )
      GO TO 9200
9999  CALL REGISOUT
      CLOSE ( UNIT=9, STATUS='DELETE' )
      WRITE(6,9450) NULL
9450  FORMAT ( A, ' PRESS "C" TO CONTINUE : ' )
9400  READ(5,9300) KEYIN
9300  FORMAT ( A )
      IF ( KEYIN.EQ. CONTINUE .OR. KEYIN.EQ. LOWC ) THEN
        CALL TXTERASE (0)
        CALL REGISTART
        CALL PLTERASE
        CALL REGISOUT
        go to 17
      ELSE
        GO TO 9400
      ENDIF
    ELSE
      GO TO 6
    ENDIF

78  IF ( MOD(NT,INC) ) 5,5,6
5   IM = IM + 1
    XBRO2(IM) = XABR / 1.05
    IF ( NOUT1.EQ. 1 ) THEN
      WRITE(1,84) XBRO2(IM), T
84  FORMAT ( 6X, F10.6, 25X, F10.3)
    82 IF ( ND.EQ. 0 ) THEN
      CALL TXTERASE ( 255 )
      CALL REGISTART
      CALL PLTERASE
      CALL FRAME
      TYPE 80
    80 FORMAT ( 'IP[130,330]', $ )
      GO TO 47
    ELSE
      CALL STIME ( T, IX, TF, 600., 130 )
      CALL SCALEM ( XBRO2(IM), IY, 1.0, 1.0 )
      TYPE 85, IX, IY
    85 FORMAT ( 'I'13', 'I3', $ )
      END IF
    ELSE IF ( NOUT1.EQ. 2 ) THEN
      IF ( ND.EQ. 0 ) THEN
        CALL TXTERASE ( 255 )
        CALL REGISTART
        CALL PLTERASE

```



```

CALL FRAME
TYPE 86
86 FORMAT ( 'IP[130,330]', $ )
GO TO 47
ELSE
CALL STIME ( T, IX, TF, 600., 130 )
CALL SCALEM ( WV(ND,1), IY, 0.08, 0.1 )
TYPE 87, IX, IY
87 FORMAT ( 'IV[I3', I3]', $ )
WRITE(2,68) WV(ND,1), WV(ND,2), T
68 FORMAT ( 8X, F10.6, 11X, F10.6, 10X, F10.3 )
END IF
ELSE IF ( NOUT1 .EQ. 3 ) THEN
IF ( ND .EQ. 0 ) THEN
CALL TXTERASE ( 255 )
CALL REGISTART
CALL PLTERASE
CALL FRAME
TYPE 88
88 FORMAT ( 'IP[130,330]', $ )
GO TO 47
ELSE
CALL STIME ( T, IX, TF, 600., 130 )
CALL SCALEM ( WV(ND,2), IY, 0.08, 0.1 )
TYPE 89, IX, IY
89 FORMAT ( 'IV[I3', I3]', $ )
WRITE(2,69) WV(ND,1), WV(ND,2), T
69 FORMAT ( 8X, F10.6, 11X, F10.6, 10X, F10.6, 6X, F10.3 )
END IF
ELSE
GO TO 82
END IF
47 CONTINUE
ND=ND+1
IF ( PH-180. ) 2,3,3
2 WV(ND,1)=W(3)
WV(ND,2)=W(1)
GO TO 4
3 WV(ND,1)=-W(1)
WV(ND,2)=-W(3)
4 XV(ND,1)=XBRO2(IM)
XV(ND,2)=0.
XV(ND,3)=1.
TV(ND)=T
6 NT=NT+1
T = NT * DT

IF ( T .LT. TSTEP1 .AND. T .LE. TF ) THEN
WBRLOLD = WBRL
GO TO 1
ELSE IF ( T .GE. TSTEP1 .AND. T .LT. TSTEP2 ) THEN
WBRL = BRSTEP
GO TO 1
ELSE IF ( T .GE. TSTEP2 .AND. T .LE. TF ) THEN
WBRL = WBRLOLD
GO TO 1
ELSE
WBRL = WBRLOLD

```

```

GO TO 18
END IF

18 CONTINUE
CALL REGISOUT
WRITE(6,42) NULL
42 FORMAT (A, 'PRESS "C" TO CONTINUE :')
44 READ(5,41) KEYIN
41 FORMAT (A)
IF ( KEYIN .EQ. CONTINUE .OR. KEYIN .EQ. LOWC ) THEN
CALL TXTERASE (0)
CALL REGISTART
GO TO 43
ELSE
GO TO 44
END IF
43 CALL PLTERASE
CALL REGISOUT
range2(1) = 0.
range2(2) = 0.
range2(3) = 0.
range2(4) = 1.
range3(1) = 0.
range3(2) = 0.
range3(3) = -.01
range3(4) = .04
nxaxis2 = 11
encode ( nxaxis2, 257, xaxis2 )
257 format ( 'TIME (SEC.)' )
nyaxis2 = 16
encode ( nyaxis2, 368, yaxis2 )
368 format ( 'O2 MOLE FRACTION' )
nxaxis3 = 11
encode ( nxaxis3, 852, xaxis3 )
852 format ( 'TIME (SEC.)' )
nyaxis3 = 18
encode ( nyaxis3, 963, yaxis3 )
963 format ( 'BREATHING FLOWRATE' )
call zetaplt ( 30, 3, nd, range2, tv, xv, nd+2, xaxis2,
nxaxis2, yaxis2, nyaxis2 )
call zetaplt ( 40, 2, nd, range3, tv, wv, nd+2, xaxis3,
nxaxis3, yaxis3, nyaxis3 )
100 FORMAT ( 14X, 8E12.3 )
GO TO 17
900 WRITE(6,910)
910 FORMAT(////////'*****',/////,
'1' PROGRAM TERMINATED BY OPERATOR',
2////////'*****',////////)
END

BLOCK DATA
REAL LIN,KA,KB,KC
COMMON/SYST/DT,PSUP,POUT,TF,WBRL,VS,VO,IIN,TCYC
COMMON/GEOM/DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN
COMMON/CAP/KA,KB,KC,BB,BC,D
COMMON/VOID/E
DATA DT, PSUP, POUT, TF, WBRL, VS, VO, IIN, TCYC /.01,
& 40.0, 15.0, 30.0, 10.0, 0.0, 0.0, 0, 10.7 /

```

```

DATA DBYIN, DVSIN, DVOIN, LIN, DIAOIN, DIAIIN /
& .075, .306174, .43866, 15.5, 5.73, 2.18 /
DATA KA, KB, KC, BB, BC, D / .2133, .007597, .05223,
& .6414, .6265, 200. /
DATA E / .37 /
END

SUBROUTINE MENU
  REAL LIN,KA,KB,KC,LIN1,KA1,KB1,KC1
  DIMENSION TI(6), NY(6), TI4(6), NY4(6), TI5(6), NY5(6)
  BYTE KEYIN
  BYTE YES
  BYTE loyes
  BYTE NO
  BYTE lono
  COMMON/SYST/DT,PSUP,POUT,TF,WBRL,VS,VO,IIN,TCYC
  COMMON/GEOM/DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN
  COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,MC,RT,AI,AO,AMW01,
    AMWL1,AMW02,AMWL2,WABR,WBBR
  COMMON/VOID/E
  COMMON/CAP/KA,KB,KC,BB,BC,D
  COMMON/NFINISH/NEXIT
  COMMON/STEP/NSTEP,TSTEP1,TSTEP2,BRSTEP
  COMMON/NPRINT/NOUT1,TI,NY
  DATA YES,loyes/89,121/
  DATA NO,lono/78,110/
  NEXIT=0
  NSTEP=0
10 CALL SYST1
  CALL GEOM1
  CALL CAP1
  WRITE(7,107)
  WRITE(6,107)
107 FORMAT('ENTER CORRESPONDING PARAMETER# TO CHANGE P/ARAMETER',/
1' ENTER "0" TO SIGNIFY PARAMETERS ARE CORRECT--RUN SIMULATION',/
2' ENTER "59" TO INSERT A STEP CHANGE IN BREATHING FLOWRATE',/
4' ENTER "99" TO EXIT PROGRAM')
  N=0
  READ *,N
  IF (N.EQ.0) GOTO 600
  IF (N.EQ.59) GOTO 500
  IF (N.EQ.99) GOTO 900
  EPSI = .0000001
  WRITE(7,120)
  WRITE(6,120)
120 FORMAT('///,*****',/
1' //, ENTER THE PARAMETER AND PRESS "RETURN",/,
2//T40,'CURRENT VALUE',/)
  IF (N.LT.6) GOTO 100
  IF (N.LT.12) GOTO 200
  IF (N.LE.16) GOTO 300
  GOTO 800
100 CONTINUE
  IF (N-2) 159,155,155
155 IF (N-3) 164,171,171
171 IF (N-4) 170,175,174
159 WRITE(6,160) PSUP
  WRITE(7,160) PSUP

```

```

160 FORMAT(' SUPPLY PRESSURE (PSIA)',T40,F8.4,/)
  READ *,PSUP1
  IF(PSUP1.LT.EPSI) GOTO 10
  PSUP=PSUP1
  GOTO 10
164 WRITE(6,166) POUT
166 FORMAT(' OUTLET PRESSURE (PSIA)',T40,F8.4,/)
  READ *,POUT1
  IF(POUT1.LT.EPSI) GOTO 10
  POUT=POUT1
  GOTO 10
170 WRITE(6,172) TF
172 FORMAT(' FINAL OBSERVATION TIME (SEC)',T40,F8.4,/)
  READ *,TF1
  IF(TF1.LT.EPSI) GOTO 10
  TF=TF1
  GOTO 10
175 WRITE(6,177) TCYC
177 FORMAT(' CYCLE TIME (SEC)',T40,F8.4,/)
  READ *,TCYC1
  IF(TCYC1.LT.EPSI) GOTO 10
  TCYC=TCYC1
  GOTO 10
174 WRITE(6,176) WBRL
176 FORMAT(' BREATHING MASS FLOWRATE (STD LIT/MIN)',T40,F8.4,/)
  READ *,WBRL1
  IF(WBRL1.LT.EPSI) GOTO 178
  WBRL=WBRL1
  GOTO 10
178 GO TO 10
200 CONTINUE
  IF(N-7) 209,215,205
205 IF(N-9) 225,235,207
207 IF(N-10) 235,245,255
209 WRITE(6,210) DBYIN
210 FORMAT(' BY-PASS VALVE DIAMETER (IN)',T40,F8.4,/)
  READ *,DBYIN1
  IF(DBYIN1.LT.EPSI) GOTO 10
  DBYIN=DBYIN1
  GOTO 10
215 WRITE(6,220) DVSIN
220 FORMAT(' SUPPLY VALVE DIAMETER (IN)',T40,F8.4,/)
  READ *,DVSIN1
  IF(DVSIN1.LT.EPSI) GOTO 10
  DVSIN=DVSIN1
  GOTO 10
225 WRITE(6,230) DVOIN
230 FORMAT(' OUTLET VALVE DIAMETER (IN)',T40,F8.4,/)
  READ *,DVOIN1
  IF(DVOIN1.LT.EPSI) GOTO 10
  DVOIN=DVOIN1
  GOTO 10
235 WRITE(6,240) LIN
240 FORMAT(' BED LENGTH (IN)',T40,F8.4,/)
  READ *,LIN1
  IF(LIN1.LT.EPSI) GOTO 10
  LIN=LIN1
  GOTO 10

```

```

245 WRITE(6,250) DIAOIN
250 FORMAT(' OUTER BED DIAMETER (IN)',T40,F8.4,/)
  READ *,DIAOIN1
  IF(DIAOIN1.LT.EPSI) GOTO 10
  DIAOIN=DIAOIN1
  GOTO 10
255 WRITE(6,260) DIAIIN
260 FORMAT(' INNER BED DIAMETER (IN)',T40,F8.4,/)
  READ *,DIAIIN1
  IF(DIAIIN1.LT.EPSI) GOTO 10
  DIAIIN=DIAIIN1
  GOTO 10
300 CONTINUE
  IF(N-13)309,315,305
305 IF(N-15)325,335,345
309 WRITE(6,310) KA
310 FORMAT(' KA', T40, F8.4, /)
  READ *, KA1
  IF ( KA1 .LT. EPSI ) GOTO 10
  KA = KA1
  GOTO 10
315 WRITE(6,320) KB
320 FORMAT(' KB', T40, F8.4, /)
  READ *, KB1
  IF ( KB1 .LT. EPSI ) GOTO 10
  KB = KB1
  GOTO 10
325 WRITE(6,330) KC
330 FORMAT(' KC', T40, F8.4, /)
  READ *, KC1
  IF( KC1 .LT. EPSI ) GOTO 10
  KC = KC1
  GOTO 10
335 WRITE(6,340) D
340 FORMAT(' DIFFUSION COEFFICIENT (>100.)',T40,F8.4,/)
  READ *,D1
  IF(D1.LT.EPSI) GOTO 10
  D=D1
  GOTO 10
345 WRITE(6,350) E
350 FORMAT(' VOID FRACTION (<1.)',T40,F8.4,/)
  READ *,E1
  IF(E1.LT.EPSI) GOTO 10
  E=E1
  GOTO 10
500 WRITE(6,510) TF
  WRITE(7,510) TF
  510 FORMAT('','','',/,
  1' ENTER THE TIME AT WHICH THE STEP CHANGE' /
  2' IN BREATHING FLOWRATE IS TO OCCUR' //,
  3' THE FINAL OBSERVATION TIME IS CURRENTLY',F7.2,' SECONDS' //)
  NSTEP=1
  READ *,TSTEP1
515 WRITE(6,518) TF
  WRITE(7,518) TF
  518 FORMAT('','','',/,
  1' ENTER THE TIME AT WHICH THE STEP CHANGE' /
  2' IN BREATHING FLOWRATE IS TO END' //,

```

```

3' THE FINAL OBSERVATION TIME IS CURRENTLY ',F7.2,' SECONDS',///)
READ*,TSTEP2
WRITE(7,520) TSTEP1,TSTEP2,WBRL
WRITE(6,520) TSTEP1,TSTEP2,WBRL
520 FORMAT(/'.....'///,
1' ENTER THE NEW BREATHING FLOWRATE OCCURRING AS',/,
2' A STEP CHANGE FROM T= ',F7.2,' TO T= ',F7.2,' SECONDS',/,
3' THE CURRENT BREATHING FLOWRATE IS',F6.2,' STD LIT/MIN',///)
READ*,BRSTEP
WRITE(7,530) BRSTEP,TSTEP1,TSTEP2
WRITE(6,530) BRSTEP,TSTEP1,TSTEP2
530 FORMAT(///,
1' .....',///,
2' THE BREATHING FLOWRATE OF ',F6.2,' STD LIT/MIN',/
3' WILL BE INPUT AS A STEP AT T= ',F7.2,' SECONDS',/
3' AND WILL END AT T= ',F7.2,' SECONDS'///,
4' .....')
GOTO 10
600 CONTINUE
WRITE(7,610)
WRITE(6,610)
610 FORMAT(/////,'.....',///,
: THE FOLLOWING PLOTS ARE AVAILABLE TO YOU ON THE TERMINAL',///,
: 1 MOLE FRACTION OF OXYGEN VS.TIME (DATA FILE=MOLEFRAC.DAT)',/,
: 2 INLET MASS FLOWRATE VS. TIME (DATA FILE=FLOWRATE.DAT)',/,
: 3 OUTLET MASS FLOWRATE VS. TIME (DATA FILE=FLOWRATE.DAT)',/,
: 4 A DYNAMIC SIMULATION FOR ONE CYCLE (OXYGEN VS. TIME',/,
: DATA FILE=PROFILE.DAT)',/,
: 5 DYNAMIC SIMULATION OF PROPAGATION OF CONTAMINANT ALONG',/,
: THE BED LENGTH',/,
: ///,
: PRESS THE CORRESPONDING # TO HAVE THE OUTPUT SHOWN AS THE',/,
: SIMULATION TRANSPIRES',/////))
READ(5,630)NOUT1
630 FORMAT(I1)
IF (NOUT1.EQ. 4) GOTO 750
IF (NOUT1.EQ. 5) GOTO 760
GOTO 400
750 DO J=1,5
FJ= FLOAT(J)
TI4(J) = 2.*FJ - 1.
NY4(J) = IFIX ( TI4(J)/DT )
END DO
TI4(6) = TCYC
NY4(6) = IFIX ( TI4(6)/DT )
DOM=1,6
TI(M) = TI4(M)
NY(M) = NY4(M)
ENDDO
GO TO 400
760 DO J=2,5
FJ= FLOAT(J)
TI5(J) = ( TF/5. ) * (FJ-1.)
NY5(J) = IFIX ( TI5(J)/DT )
END DO
TI5(1) = 1.0
NY5(1) = IFIX ( TI5(1)/DT )
TI5(6) = TF

```

```

      NY5(6) = IFIX ( TI5(6)/DT )
      DOM=1,6
      TI(M) = TI5(M)
      NY(M) = NY5(M)
      ENDDO
400 WRITE(6,410)
      WRITE(7,410)
410 FORMAT(////,' YOU HAVE THE FOLLOWING OPTIONS :',
      .////,T10,'1. ENTER "0" TO RUN SIMULATION.',/,
      .T10,'2. ENTER "19" TO CHANGE THE PARAMETERS.',/,
      .T10,'3. ENTER "99" TO EXIT PROGRAM.',////,
      .ENTER THE CORRESPONDING # AND PRESS "RETURN",////)
      READ *, N69
      IF(N69.EQ.0) GOTO 820
      IF(N69.EQ.19) GOTO 10
      IF(N69.EQ.99) GOTO 900
800 CONTINUE
      WRITE(7,810)
      WRITE(6,810)
810 FORMAT(////,'*****',////,
      1'THE PARAMETER VALUE WAS NOT INPUT CORRECTLY',/
      2' TRY AGAIN',////,'*****',////)
      GOTO 10
820 WRITE(6,825)
      WRITE(7,825)
825 FORMAT(////,' DO YOU WISH TO DOUBLE THE NUMBER OF SPACE LUMPS IN',
      1'THE SIMULATION FOR BETTER ACCURACY ? ( CURRENT NUMBER USED',/
      2' IS 50 ) (Y/N)',////)
      READ (5,830) KEYIN
830 FORMAT ( A )
      IF ( (KEYIN .EQ. YES) .OR. (KEYIN .EQ. loyes) ) THEN
        LUMPS = 101
      ELSE
        LUMPS = 51
      END IF
      GO TO 1000
900 CONTINUE
      NEXIT = 1
1000 RETURN
      END

```

```

SUBROUTINE SYST1
COMMON/SYST/DT,PSUP,POUT,TF,WBRL,VS,VO,IIN,TCYC
WRITE(6,350) PSUP,POUT,TF,TCYC,WBRL
WRITE(7,350) PSUP,POUT,TF,TCYC,WBRL
350 FORMAT(' CURRENT SYSTEM PARAMETERS ARE:',/
1' 1,'T5,' SUPPLY PRESSURE=',T30,F8.2,T40,'PSIA',/,
2' 2,'T5,' OUTLET PRESSURE=',T30,F8.2,T40,'PSIA',/,
3' 3,'T5,' FINAL OBSERVATION TIME=',T30,F8.2,T40,'SEC',/,
4' 4,'T5,' CYCLE TIME=',T30,F8.2,T40,'SEC',/,
5' 5,'T5,' BREATHING FLOWRATE=',T30,F8.2,T40,'STD LIT/MIN')
RETURN
END

```

```

SUBROUTINE GEOM1
REAL LIN
COMMON/GEOM/DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN
WRITE(6,450) DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN

```

```

WRITE(7,450) DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN
450 FORMAT(' CURRENT GEOMETRIC PARAMETERS ARE: ',
1' 6.',T5,' BY-PASS VALVE DIAMETER = ',T30,F8.4,T40,'IN'./,
2' 7.',T5,' SUPPLY VALVE DIAMETER = ',T30,F8.4,T40,'IN'./,
3' 8.',T5,' OUTLET VALVE DIAMETER = ',T30,F8.4,T40,'IN'./,
4' 9.',T5,' BED LENGTH = ',T30,F8.4,T40,'IN'./,
5' 10.',T5,' OUTER BED DIAMETER = ',T30,F8.4,T40,'IN'./,
6' 11.',T5,' INNER BED DIAMETER = ',T30,F8.4,T40,'IN' )
RETURN
END

SUBROUTINE CAP1
REAL KA,KB,KC
COMMON/CAP/KA,KB,KC,BB,BC,D
COMMON/VOID/E
WRITE(7,550)KA,KB,KC,D,E
WRITE(6,550)KA,KB,KC,D,E
550 FORMAT(' THE CURRENT BED PARAMETERS ARE: ',
1' 12.',T5,' KA = ',T30,F8.4,T40,'KGMOL/02 ADSORBED/KGMOL GAS'./,
2' 13.',T5,' KB = ',T30,F8.4,T40,'KGMOL/CO ADSORBED/KGMOL GAS'./,
3' 14.',T5,' KC = ',T30,F8.4,T40,'KGMOL/N2 ADSORBED/KGMOL GAS'./,
4' 15.',T5,' DIFFUSION COEFFICIENT = ',T30,F8.2,T40,'1/SEC'./,
5' 16.',T5,' VOID FRACTION = ',T30,F8.3)
RETURN
END

```

```

SUBROUTINE TIMEF(CVA,PH,PS,T)
C*****CALCULATES CVA(T) AND PS(T)
DIMENSION CVA(3),PS(2)
DATA PHI,PH1,PH2,PH3,PH4,PH5,PH6/10.,20.,160.,180.,
1 200.,340.,360./
COMMON/SYST/DT,PSUP,POUT,TF,WBRL,VS,VO,IIN,TCYC
COMMON/HILOBED/NBED,NB
C*****
C
C PH= PHASE ANGLE OF CYCLE (DEGREES)
C TCYC= CYCLE TIME (SEC)
C*****
C
I=0
PH=AMOD(360.*T/TCYC+PHI,360.)
IF(PH-PH1)1,2,2
1 RA=PH/PH1
GO TO 11
2 IF(PH-PH2)3,4,4
3 RA=1.
GO TO 11
4 IF(PH-PH3)5,6,6
5 RA=(PH3-PH)/(PH3-PH2)
GO TO 11
6 IF(PH-PH4)7,8,8
7 RA=(PH-PH3)/(PH4-PH3)
I=1
GO TO 11
8 IF(PH-PH5)9,10,10
9 RA=1.
I=1
GO TO 11

```


10 RA = (PH6-PH1) / (PH6-PH5)

I = 1

11 NBED = I

IF (I) 12,12,13

12 PS(1) = PSUP

PS(2) = POUT

CVA(1) = VS*RA

CVA(3) = VO*RA

RETURN

13 PS(1) = POUT

PS(2) = PSUP

CVA(1) = VO*RA

CVA(3) = VS*RA

RETURN

END

SUBROUTINE INITIAL (C1, C2, N1, N2, XA, XB, P)

REAL L,MA,MB,MC,NA01,NA02,KA,KB,KC,N1,N2,NB01,NB02,LIN,NC01,NC02

DIMENSION N1(300,3),N2(300,3),XA(3),XB(3),P(2)

COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,MC,RT,AI,AO,AMW01,AMWL1,AMW02

1,AMWL2,WBR

COMMON/CAP/KA,KB,KC,BB,BC,D

COMMON/VOID/E

COMMON/GEOM/DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN

C.....

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

RT = GAS CONSTANT*TEMP (N-M / KGMOLE)

AITOT,AOTOT = INLET AND OUTLET CONCENTRIC BED AREAS (M**2)

MA,MB = KGMOLECULAR WEIGHTS (KG / KGMOLE)

0,L = SUBSCRIPTS FOR Z=0 AND Z=L

AMW = AVERAGE MOLECULAR WEIGHT (KG / KGMOLE)

1,2 = SUBSCRIPTS FOR BED 1 AND BED 2

E = VOID FRACTION

RT = 8314. * 300.

AITOT = 3.14159 * (DIAIIN*.0254)**2 / 4.

AOTOT = 3.14159 * (DIAOIN**2-DIAIIN**2) *.0254**2 / 4.

L = .0254 * LIN

PRINT *, 'BED LENGTH (M) = ', L

PRINT *, 'INLET BED AREA (M**2) = ', AITOT

PRINT *, 'OUTLET BED AREA (M**2) = ', AOTOT

PRINT *, 'VOID FRACTION = ', E

MA = 32.

MB = 28.

MC = 28.

AI = AITOT*E

AO = AOTOT*E

AMW01 = MA*XA(1) + MB*XB(1) + MC*(1.-XA(1)-XB(1))

AMWL1 = MA*XA(2) + MB*XB(2) + MC*(1.-XA(2)-XB(2))

AMW02 = AMWL1

AMWL2 = MA*XA(3) + MB*XB(3) + MC*(1.-XA(3)-XB(3))

C1 = P(1)*6.895E3 / RT

C2 = P(2)*6.895E3 / RT

CA1 = C1*XA(1)

CA2 = C2*XA(3)

CB1 = C1*XB(1)

CB2 = C2*XB(3)

```

CC1 = C1-CA1-CB1
CC2 = C2-CA2-CB2
NA01 = CA1 / KA
NA02 = CA2 / KA
NB01 = (CB1/KB) / (1.+CB1/(KB*BB))
NB02 = (CB2/KB) / (1.+CB2/(KB*BB))
NC01 = (CC1/KC) / (1.+CC1/(KC*BC))
NC02 = (CC2/KC) / (1.+CC2/(KC*BC))
DO 1 I=1,LUMPS
N1(I,1) = NA01 + NB01 + NC01
N1(I,2) = NA01
N1(I,3) = NB01
N2(I,1) = NA02 + NB02 + NC02
N2(I,2) = NA02
1 N2(I,3) = NB02
DZ=L/FLOAT(LUMPS-1)
RETURN
END

SUBROUTINE BEDS ( P, W, XA, XB, XABR, XBBR, CVA, PS, DTG,
                  CA1, CA2, CB1, CB2, C1, C2 )
C
C*****SOLVES BED EQUATIONS
C
REAL L,N1,N2,MA,MB,MC
DIMENSION CA1(300),CA2(300),CB1(300),CB2(300),N1(300,3),N2(300,3),
1      G1(300,3),G2(300,3),XAT(2),XBT(2),XAO(2),XBO(2)
1      W(3),XA(3),XB(3),CVA(3),PS(2),PN(2),P(2)
COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,MC,RT,AI,AO,AMW01,AMWL1,AMW02
1      AMWL2,WBR
COMMON/VOID/E
COMMON/SYST/DT,PSUP,POUT,TF,WBRL,VS,VO,IIN,TCYC
COMMON/CAP/KA,KB,KC,BB,BC,D
COMMON/TIME/T
C.....
C
C  C = TOTAL GAS STREAM CONCENTRATION ( KGMOLE / M**3 )
C  CA = GAS A STREAM CONCENTRATION ( KGMOLE / M**3 )
C  N = ADSORBED PHASE CONCENTRATION ( KGMOLE / M**3 )
C  G = GAS PHASE CONCENTRATION ( KGMOLE / M**3 )
C  U = STREAM VELOCITY ( M / SEC )
C  Y = MOLECULAR FLUX ( KGMOLE / M**2-SEC )
C  L = BED LENGTH ( M )
C  D = DIFFUSION COEFFICIENT ( 1 / SEC )
C  BETA = D*AREA RATIO ( 1 / SEC )
C  DT = TIME STEP ( SEC )
C  LUMPS = NUMBER OF SPACE LUMPS
C  1,2 = SUBSCRIPTS FOR BED 1 AND BED 2
C  VBR = MIXING VOLUME FOR BREATHING FLOW ( M**3 )
C  BR = SUBSCRIPT INDICATING BREATHING VARIABLE
C  CP = PLENUM MOLE CONCENTRATION ( KGMOLE / M**3 )
C  VP = BREATHING PLENUM VOLUME ( M**3 )
C.....
C
DT=DTG
DO 4 I=1,2
4  PN(I) = PS(I)*6.895E3
   IF(IIN.EQ.1)GO TO 1

```

```

IIN = 1
CALL INITIAL ( C1, C2, N1, N2, XA, XB, P )
VPMUL = 1.
VP = .0018 * VPMUL
PRINT * ' PLENUM VOLUME (M**3) = ', VP
DO I=1,7
PRINT *
END DO
WRITE(6,10) NULL
10 FORMAT ( A, ' PLEASE WAIT.....' )
VBR = .5*VP + .0003
CP = .95*AMAX1( PN(1),PN(2) ) / RT
DO 3 I=1,2
XAT(I) = XABR
3 XBT(I) = XBBR
1 D1 = D
D2 = D
WBR = WBR1*4.72E-5 / 2.205
CALL CONCEN ( C1,C2,CA1,CA2,CB1,CB2,W,G1,G2,XA,XB,N1,N2,PN,CVA)
XABR = XAT(2)
XBBR = XBT(2)
AMWBR = MA*XABR + MB*XBBR + MC*(1.-XABR-XBBR)
DO 5 I=1,2
XAO(I) = XAT(I)
5 XBO(I) = XBT(I)
DO 2 I=1,LUMPS
N1(I,1) = N1(I,1) - D1*(G1(I,1)-C1)*DT
N1(I,2) = N1(I,2) - D1*(G1(I,2)-CA1(I))*DT
N1(I,3) = N1(I,3) - D1*(G1(I,3)-CB1(I))*DT
N2(I,1) = N2(I,1) - D2*(G2(I,1)-C2)*DT
N2(I,2) = N2(I,2) - D2*(G2(I,2)-CA2(I))*DT
N2(I,3) = N2(I,3) - D2*(G2(I,3)-CB2(I))*DT
2 CONTINUE
FAC = CP * VBR * AMWBR
TAUI = WBR / FAC
XAT(1) = XAO(1) + TAUI*(XA(2)-XAO(1))*DT
XBT(1) = XBO(1) + TAUI*(XB(2)-XBO(1))*DT
XAT(2) = XAO(2) + TAUI*(XAO(1)-XAO(2))*DT
XBT(2) = XBO(2) + TAUI*(XBO(1)-XBO(2))*DT
P(1) = C1*RT / 6.895E3
P(2) = C2*RT / 6.895E3
AMW01 = MA*XA(1) + MB*XB(1) + MC*(1.-XA(1)-XB(1))
AMWL1 = MA*XA(2) + MB*XB(2) + MC*(1.-XA(2)-XB(2))
AMW02 = AMWL1
AMWL2 = MA*XA(3) + MB*XB(3) + MC*(1.-XA(3)-XB(3))
RETURN
END

SUBROUTINE CONCEN(CG1,CG2,CA1,CA2,CB1,CB2,W,G1,G2,
1 XA,XB,N1,N2,PN,CVA)
C*****SOLVES FOR CONCENTRATIONS
REAL L,N1,N2,MA,MB,MC
DIMENSION CA1(300),CA2(300),CB1(300),CB2(300),N1(300,3),N2(300,3)
1 G1(300,3),G2(300,3)
DIMENSION GI1(300),GI2(300),PN(2),W(3),XA(3),XB(3),CVA(3)
COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,MC,RT,AI,AO,AMW01,AMWL1,AMW02
1 ,AMWL2,WBR
COMMON/VOID/E

```

RETURN
END

SUBROUTINE CONcab (CG, CA, CB, G, GI, U0G, ULG, XA0, XAL,
1 XB0, XBL, BETAG, ARG)
C*****SOLVES FOR CONCENTRATION OF GAS A AND B
REAL L2,L
DIMENSION CA(300),CB(300),G(300,3),GI(300),XA(2),XB(2),
1 U(300),YA(300),YB(300)
COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,MC,RT,AI,AO,AMWO1,AMWL1,AMWO2
1 ,AMWL2,WBR
COMMON/VOID/E
COMMON/TIME/T
C = CG
BETA = BETAG
UL = ULG
U0 = U0G
XA(1) = XA0
XA(2) = XAL
XB(1) = XB0
XB(2) = XBL
AR = ARG
LUM2 = (LUMPS+1) / 2
GIL2 = GI(LUM2)
L2 = .5*L
IF (U0**2 +UL**2) 22,22,23

C*****ZERO FLOW SOLUTION

22 DO 24 I=1,LUMPS
CA(I) = G(I,2)
24 CB(I) = G(I,3)
RETURN
23 CONTINUE
CA(1) = C*XA(1)
CB(1) = C*XB(1)
U(1) = U0
YA(1) = U0*CA(1)
YB(1) = U0*CB(1)
CA(LUMPS) = C*XA(2)
CB(LUMPS) = C*XB(2)
YA(LUMPS) = UL*CA(LUMPS)
YB(LUMPS) = UL*CB(LUMPS)
Z = 0.0

C*****ICR IS INDEX INDICATING U=0 IN BED

ICR = 1
ICR1 = ICR - 1

C*****SOLVE FOR U IN BED

DO 3 I=2,LUMPS
IM1 = I-1
Z = Z+DZ
IF (IM1-LUM2) 35,34,34
35 U(I) = U(1) +BETA*(GI(I)/C - Z)
GO TO 36

```

34  U(I) = AR*U(LUM2) + BETA*( (GI(I)-GIL2)/C - Z + L2 )
36  CONTINUE

C*****CHECK SIGN CHANGE OF U

      IF ( U(I)*U(IM1) ) 8,9,3
8    ICR = I
      ICR1 = ICR-1

C*****CALCULATE INTERIOR INITIAL FLUX ( U0<0, UL>0 )

      ULAST = U(IM1)
      IF ( IM1 .EQ. LUM2 ) ULAST=AR*ULAST
      DZCR = DZ*U(I) / ( U(I)-ULAST )
      DTC = DZCR - DZ
      R1 = U(ICR) / DZCR
      R3A = ( G(I,2)-G(IM1,2) ) / DZ
      R3B = ( G(I,3)-G(IM1,3) ) / DZ
      R2A = G(IM1,2) - R3A*DTC
      R2B = G(IM1,3) - R3B*DTC
      A1A = BETA*R2A*R1 / ( BETA+R1 )
      A1B = BETA*R2B*R1 / ( BETA+R1 )
      A2A = BETA*R3A*R1 / ( BETA+2.*R1 )
      A2B = BETA*R3B*R1 / ( BETA+2.*R1 )
      YA(ICR) = A1A*DZCR + A2A*DZCR*DZCR
      YB(ICR) = A1B*DZCR + A2B*DZCR*DZCR
      YA(ICR1) = A1A*DTC + A2A*DTC*DTC
      YB(ICR1) = A1B*DTC + A2B*DTC*DTC
      CA(ICR) = YA(ICR) / U(ICR)
      CB(ICR) = YB(ICR) / U(ICR)
      CA(ICR1) = YA(ICR1) / ULAST
      CB(ICR1) = YB(ICR1) / ULAST
      GO TO 3
9    ICR = I
      ICR1 = I
      YA(ICR) = 0.0
      YB(ICR) = 0.0
      IF ( U0 ) 3,25,3
25   ICR = 1
      ICR1 = 1
      YA(1) = 0.0
      YB(1) = 0.0
3    CONTINUE
      UL = U(LUMPS)
C*****LOGIC FOR FORWARD INTEGRATION
      ISIG = 1
      IF ( UL ) 14,14,15
14   IST = 1
      IF ( U0 ) 12,20,21
21   CONTINUE
      IT = ICR-2
      IF ( IT ) 12,12,11
20   ICR = 1
      ICR1 = 1
      GO TO 12
15   CONTINUE
      IST = ICR
      IT = LUMPS-ICR

```

```

      IF ( IT ) 12,12,11
11  CONTINUE

C*****SOLVE FOR CA

      DO 10 J=1,IT
      I = IST + ISIG*J
      IM1 = I-ISIG
      IF ( IM1-LUM2 ) 33,30,33
30  IF ( ISIG ) 31,33,32
31  YA(IM1) = YA(IM1) / AR
      YB(IM1) = YB(IM1) / AR
      U(IM1) = U(IM1) / AR
      GO TO 33
32  U(IM1) = AR*U(IM1)
      YA(IM1) = AR*YA(IM1)
      YB(IM1) = AR*YB(IM1)
33  CONTINUE
      R0 = U(IM1)
      IF ( I .EQ. LUM2 .AND. ISIG .EQ. -1 ) U(I)=AR*U(I)
      R1 = ( U(I)-R0 ) / DZ
      R2A = G(IM1,2)
      R2B = G(IM1,3)
      R3A = ( G(I,2)-R2A ) / DZ
      R3B = ( G(I,3)-R2B ) / DZ
      IF ( R1 ) 6,7,6
7   A0A = ( R2A-R3A*R0/BETA ) * R0
      A0B = ( R2B-R3B*R0/BETA ) * R0
      A1A = R3A*R0
      A1B = R3B*R0
      BR = BETA / R0
      YA(I) = ( YA(IM1)-A0A ) * EXP(-BR*DZ) + A0A + A1A*DZ
      YB(I) = ( YB(IM1)-A0B ) * EXP(-BR*DZ) + A0B + A1B*DZ
      GO TO 10
6   A2A = BETA*R3A*R1 / ( BETA+2.*R1 )
      A2B = BETA*R3B*R1 / ( BETA+2.*R1 )
      A1A = ( BETA*(R3A*R0+R2A*R1) - 2.*A2A*R0 ) / ( BETA+R1 )
      A1B = ( BETA*(R3B*R0+R2B*R1) - 2.*A2B*R0 ) / ( BETA+R1 )
      A0A = ( BETA*R0*R2A-A1A*R0 ) / BETA
      A0B = ( BETA*R0*R2B-A1B*R0 ) / BETA
      IF ( R0 ) 26,27,26
27  YA(I) = A1A*DZ + A2A*DZ*DZ
      YB(I) = A1B*DZ + A2B*DZ*DZ
      GO TO 28
26  CONTINUE
      BR = BETA/R1
      YA(I) = ( YA(IM1)-A0A ) * ( 1.+R1*DZ/R0 )**(-BR) + A0A + A1A*DZ
      + A2A*DZ*DZ
      YB(I) = ( YB(IM1)-A0B ) * ( 1.+R1*DZ/R0 )**(-BR) + A0B + A1B*DZ
      + A2B*DZ*DZ
28  CONTINUE
      IF ( U(I) ) 18,19,18
19  CA(I) = C*G(I,2) / G(I,1)
      CB(I) = C*G(I,3) / G(I,1)
      GO TO 10
18  CA(I) = YA(I) / U(I)
      CB(I) = YB(I) / U(I)
10  CONTINUE

```

```

      IF ( ICR .EQ. 1 ) GO TO 13
      IF ( DZ ) 13,13,12
12  CONTINUE

C*****LOGIC FOR BACKWARD INTEGRATION

      DZ = -DZ
      ISIG = -1
      IF ( UL ) 16,16,17
16  IST = LUMPS
      IT = LUMPS - ICR
      IF ( ICR .EQ. ICR1 ) IT=IT-1
      IF ( IT ) 13,13,11
17  CONTINUE
      IST = ICR1
      IT = ICR1-1
      IF ( IT ) 13,13,11
13  CONTINUE
      XA(1) = CA(1) / C
      XA(2) = CA(LUMPS) / C
      XB(1) = CB(1) / C
      XB(2) = CB(LUMPS) / C
      DZ = FLOAT(ISIG) * DZ
      XA0 = XA(1)
      XAL = XA(2)
      XB0 = XB(1)
      XBL = XB(2)
      RETURN
      END
      SUBROUTINE CAPZ ( N, G, I )
      REAL KA,KB,KC,N,NA,NB,NC
      DIMENSION N(300,3),G(300,3)
      COMMON/CAP/KA,KB,KC,BB,BC,D
      NA = N(I,2)
      NB = N(I,3)
      NC = N(I,1) - NA - NB
      G(I,2) = KA*NA
      G(I,3) = KB*NB / (1.-NB/BB )
      GC = KC*NC / (1.-NC/BC )
      G(I,1) = G(I,2) + G(I,3) + GC
      RETURN
      END
      SUBROUTINE VALVE ( A, PU, PD, W )
C*****CALCULATES VALVE MASS FLOWRATE KG/SEC
      REAL KI
      DATA KI, C1, C2, CK / .714, .00906, .00234, .286 /
      PUL = PU
      PDL = PD
      PR = PDL / PUL
      SIG = 1.
      IF ( 1.-PR ) 4,3,2
3  W = 0.
      RETURN
4  PR = 1. / PR
      PUL = PDL
      SIG = -1.
2  CONTINUE
      IF ( PR .LT. .528 ) GO TO 1

```

```

    PRR = 1. - PR**CK
    W = SIG*A*C1*PUL*(PR**KI)*SQRT(PRR)
    RETURN
1  W = SIG*A*C2*PUL
    RETURN
    END
    SUBROUTINE VAO ( CG1,CG2)
C*****ITERATION AND FALSE POSITION TO SOLVE VALVE/BED EQUATIONS
    EXTERNAL F1,F2
    COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,MC,RT,AI,AO,AMW01,AMWL1,AMW02
    1,AMWL2,WBR
    COMMON/VBEQ/W1,W2,W3,U01,UL1,U02,UL2,PN1,PN2,VA1,VA2,VA3
    1,BGI1L,BGI1L2,BGI2L,BGI2L2,BL21,BL22,P1,P2
    C1 = CG1
    C2 = CG2
    P1 = C1*RT
    PO1 = P1
    J = 1
10  FC2 = F2 ( C2 )
    1  CN2 = C2*( 1.-FC2 )
    IF ( ABS(CN2-C2)/C2 .LE. 1.E-6 ) GO TO 2
    FCN2 = F2 ( CN2 )
    IF ( FC2*FCN2 ) 4,3,3
    3  C2 = CN2
    FC2 = FCN2
    GO TO 1
    4  C2 = FP ( F2, FCN2, CN2, FC2, C2 )
    2  CONTINUE
    FC1 = F1 ( C1 )
    5  CN1 = C1*(1.-FC1)
    IF ( ABS(CN1-C1)/C1 .LE. 1.E-6 ) GO TO 6
    FCN1 = F1 ( CN1 )
    IF ( FC1*FCN1 ) 8,7,7
    7  C1 = CN1
    FC1 = FCN1
    GO TO 5
    8  C1 = FP ( F1, FCN1, CN1, FC1, C1 )
    6  IF ( ABS(P1-PO1)/P1 .LE. 1.E-6 ) GO TO 9
    J = J+1
    PO1 = P1
    IF ( J .LE. 20 ) GO TO 10
    PRINT 100
100  FORMAT ( 1X, "FAILED TO CONVERGE" )
    9  CG1 = C1
    CG2 = C2
    RETURN
    END
    FUNCTION F1 ( CG1 )
C*****BED 1 VALVE/BED EQUATION
    COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,MC,RT,AI,AO,AMW01,AMWL1,AMW02
    1,AMWL2,WBR
    COMMON/VBEQ/W1,W2,W3,U01,UL1,U02,UL2,PN1,PN2,VA1,VA2,VA3
    1,BGI1L,BGI1L2,BGI2L,BGI2L2,BL21,BL22,P1,P2
    C1 = CG1
    P1 = C1*RT
    CALL VALVE ( VA1, PN1, P1, W1 )
    CALL VALVE ( VA2, P1, P2, W2 )
    IF ( P1-P2 ) 1,2,3

```



```

1  WBR1 = 0.
   GO TO 4
2  WBR1 = .5*WBR
   GO TO 4
3  WBR1 = WBR
4  CONTINUE
   U01 = W1 / ( C1*AMW01*AI )
   UL1 = ( W2+WBR1 ) / ( C1*AMWL1*AO )
   AR = AI / AO
   F1 = 1. - ( AR*BGI1L2+BGI1L ) / ( (UL1+BL21-AR*(U01-BL21))*C1 )
   RETURN
   END
   FUNCTION F2 ( CG2 )
C*****BED 2 VALVE/BED EQUATION
   COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,MC,RT,AI,AO,AMW01,AMWL1,AMW02
   1,AMWL2,WBR
   COMMON/VBEQ/W1,W2,W3,U01,UL1,U02,UL2,PN1,PN2,VA1,VA2,VA3
   1,BGI1L,BGI1L2,BGI2L,BGI2L2,BL21,BL22,P1,P2
   C2 = CG2
   P2 = C2*RT
   CALL VALVE ( VA2, P1, P2, W2 )
   CALL VALVE ( VA3, P2, PN2, W3 )
   IF ( P2-P1 ) 1,2,3
1  WBR2 = 0.
   GO TO 4
2  WBR2 = .5*WBR
   GO TO 4
3  WBR2 = WBR
4  CONTINUE
   U02 = ( W2-WBR2 ) / ( C2*AMW02*AO )
   UL2 = W3 / ( C2*AMWL2*AI )
   AR = AO / AI
   F2 = 1. - ( AR*BGI2L2+BGI2L ) / ( (UL2+BL22-AR*(U02-BL22))*C2 )
   RETURN
   END
   FUNCTION FP ( F, FXR, XR, FXL, XL )
C*****USES FALSE POSITION TO FIND ZERO
   DATA EPS / 1.E-6 /
   I = 0
3  X = XL - FXL*(XL-XR) / ( FXL-FXR )
   FX = F ( X )
   I = I+1
   IF ( ABS(FX) .LT. EPS ) GOTO 4
   IF ( I.GT. 50 ) GOTO 4
   IF ( FX*FXL ) 1,1,2
1  XR = X
   FXL = FXL*FXR / ( FXR+FX )
   FXR = FX
   GOTO 3
2  XL = X
   FXR = FXR*FXL / ( FXL+FX )
   FXL = FX
   GOTO 3
4  FP = X
   RETURN
   END

```

subroutine zetaplt(nplot,ncvs,npts,range,x,y,ny,xaxis,nxaxis,

```

        yaxis,nyaxis)
dimension x(300),y(302,11),c(6),range(4),xaxis(10),yaxis(10)
data axlenx,axleny / 7.0, 5.0 /
call plots ( 0, 0, nplot )
call plot ( 0.5, 0.5, -3 )
call factor ( 0.65 )
if ( (range(1).eq.0.) .and. (range(2).eq.0.) ) then
    call scale ( x, axlenx, npts, 1 )
else
    x(npts+1) = range(1)
    x(npts+2) = (range(2)-range(1))/axlenx
end if
if ( (range(3).eq.0.) .and. (range(4).eq.0.) ) then
    call scale ( y, axleny, npts, 1 )
else
    y(npts+1,1) = range(3)
    y(npts+2,1) = (range(4)-range(3))/axleny
end if
if ( ncvs .ge. 2 ) then
    do n = 2,ncvs
        y(npts+1,n) = y(npts+1,1)
        y(npts+2,n) = y(npts+2,1)
    end do
end if
call axis ( 0.0, 0.0, xaxis, -nxaxis, axlenx, 0.0,
            x(npts+1), x(npts+2) )
call axis ( 0.0, 0.0, yaxis, nyaxis, axleny, 90.0,
            y(npts+1,1), y(npts+2,1) )
do m = 1,ncvs
    call line ( x, y(1,m), npts, 1, 0, 2 )
end do
call plot ( 0.0, 0.0, 999 )
return
end

```

```

SUBROUTINE SCALE1 ( Y, YABS )
COMMON/HILOBED/NBED
INTEGER YABS
DY = 1.0 - Y
DPY = DY * 150.0
IDY = IFIX ( DPY )
YABS = IDY + 100
RETURN
END

```

```

SUBROUTINE SCALE2 ( Y, YABS )
COMMON/HILOBED/NBED
INTEGER YABS
DY = 1.0 - Y
DPY = DY * 150.0
IDY = IFIX ( DPY )
YABS = IDY + 300
RETURN
END

```

```

SUBROUTINE INITIAL1 ( YABS1, YABS2, NBED )
INTEGER YABS1,YABS2
IF ( NBED .EQ. 0 ) THEN
    WRITE(9,100) YABS1
ELSE

```

```

WRITE(9,100) YABS2
END IF
100 FORMAT (3X, ' P[275,'I3']')
RETURN
END
SUBROUTINE INITIAL2 ( YABS1, YABS2, NBED )
INTEGER YABS1,YABS2
IF ( NBED .EQ. 0 ) THEN
WRITE(9,100) YABS2
ELSE
WRITE(9,100) YABS1
END IF
100 FORMAT (3X, ' P[575,'I3']')
RETURN
END
SUBROUTINE DRAWF ( XABS, YABS, NM )
C DRAW VECTORS FROM PREVIOUS CURSOR POSITIONS TO CURRENT POINTS
C FORWARD PLOTTING
INTEGER XABS, YABS
WRITE(9,100) NM, XABS, YABS
100 FORMAT (3X, ' W(R,N'I1')V[V['I3','I3']')
RETURN
END
SUBROUTINE DRAWB ( XABS, YABS, NM )
C BACKWARD PLOTTING
INTEGER XABS, YABS
WRITE(9,100) NM, XABS, YABS
100 FORMAT (3X, ' W(R,N'I1')V[V['I3','I3']')
RETURN
END
SUBROUTINE TXTERASE (NTICKS)
C ERASE THE VT125 TEXT SCREEN
LOGICAL ERASE
DIMENSION ERASE(4)
DATA ERASE / 27, 'I', '2', 'J' /
IF ( NTICKS.NE.0 ) THEN
DO I=1,7
CALL DELAY (NTICKS)
END DO
ELSE
END IF
TYPE 10, ERASE
10 FORMAT ('+', 5A1, '$')
RETURN
END
SUBROUTINE REGISTART
C SETS VT125 INTO GRAPHICS(REGIS) MODE
LOGICAL GRAPHIC
DIMENSION GRAPHIC(3)
DATA GRAPHIC / 27, 'P', 'p' /
TYPE 10, GRAPHIC
10 FORMAT ('+', 5A1, '$')
RETURN
END
SUBROUTINE PLTERASE
C ERASES PLOTS
LOGICAL ERASE
DIMENSION ERASE(4)

```

```

DATA ERASE / 'S', '(', 'E', ')' /
TYPE 10, ERASE
10 FORMAT ('+', 5A1, $)
RETURN
END
SUBROUTINE FRAME1
WRITE(9,110)
110 FORMAT (' P[250,5]T(S2)"OBOGS SIMULATION" )
WRITE(9,112)
112 FORMAT (' P[30,30]T(S1)"TIME = SEC." )
WRITE(9,114)
114 FORMAT (' P[260,95]T"1"P[260,245]T"0" )
WRITE(9,116)
116 FORMAT (' P[260,295]T"1"P[260,445]T"0" )
WRITE(9,120)
120 FORMAT (' P[400,80]T"BED #1" )
WRITE(9,130)
130 FORMAT (' P[400,280]T"BED #2" )
WRITE(9,140)
140 FORMAT (' P[20,255]T"INLET" )
WRITE(9,150)
150 FORMAT (' P[5,275]T"PRESSURE" )
WRITE(9,160)
160 FORMAT (' P[195,112]T"R.V." )
WRITE(9,170)
170 FORMAT (' P[95,430]T"EXHAUST" )
WRITE(9,180)
180 FORMAT (' P[115,450]T"GAS" )
WRITE(9,190)
190 FORMAT (' P[624,268]T"P.O." )
WRITE(9,192)
192 FORMAT (' P[680,112]T"B.P." )
WRITE(9,193)
193 FORMAT (' P[720,412]V00000P[730,403]T"0%" )
WRITE(9,195)
195 FORMAT (' P[720,356]V00000P[730,349]T"20%" )
WRITE(9,196)
196 FORMAT (' P[720,132]V00000P[730,123]T"100%" )
WRITE(9,200)
200 FORMAT (' P[275,100]V[575,100]V[575,250]V[275,250]V[275,100]' )
WRITE(9,210)
210 FORMAT (' P[275,300]V[575,300]V[575,450]V[275,450]V[275,300]' )
WRITE(9,300)
300 FORMAT (' P[575,162]V[670,162]V[670,132]V[720,132]V[720,412]' )
WRITE(9,310)
310 FORMAT (' V[670,412]V[670,382]V[575,382]P[575,182]' )
WRITE(9,400)
400 FORMAT (' V[600,182]V[600,267]V[605,272]V[600,277]V[600,362]' )
WRITE(9,410)
410 FORMAT (' V[575,362]P[620,182]V[670,182]V[670,362]V[620,362]' )
WRITE(9,450)
450 FORMAT (' V[620,277]V[615,272]V[620,267]V[620,182]' )
WRITE(9,460)
460 FORMAT (' P[720,262]V[750,262]V[750,282]V[720,282]' )
WRITE(9,500)
500 FORMAT (' P[275,162]V[240,162]V[240,132]V[180,132]' )
WRITE(9,510)
510 FORMAT (' V[180,162]V[118,162]V[118,262]' )

```

```

WRITE(9,520)
520 FORMAT ( ' P[275,182]V[240,182]P[180,182]V[138,182]V[138,262]
1 ' )
WRITE(9,540)
540 FORMAT ( ' P[118,282]V[118,420]V[138,420]V[138,382]' )
WRITE(9,600)
600 FORMAT ( ' V[180,382]V[180,412]V[240,412]V[240,382]V[275,382]' )
WRITE(9,610)
610 FORMAT ( ' P[180,262]V[80,262]V[80,282]' )
WRITE(9,700)
700 FORMAT ( ' V[180,282]V[180,362]V[138,362]V[138,282]P[275,362]' )
WRITE(9,710)
710 FORMAT ( ' P[275,362]V[240,362]V[240,182]P[180,182]V[180,262]' )
WRITE(9,730)
730 FORMAT ( ' P[630,162]V[630,167]V[640,162]' )
WRITE(9,735)
735 FORMAT ( ' P[630,182]V[630,177]V[640,182]' )
WRITE(9,740)
740 FORMAT ( ' P[630,362]V[630,367]V[640,362]' )
WRITE(9,750)
750 FORMAT ( ' P[630,382]V[630,377]V[640,382]' )
RETURN
END
SUBROUTINE NEWTIME ( TIME,NW )
WRITE(9,100) NW,TIME
100 FORMAT ( ' W(N'I1')P[110,30]T(S1)"F6.2"' )
RETURN
END
SUBROUTINE FRAME2 ( NBED,NW )
IF ( NBED.EQ. 1 ) THEN
WRITE(9,710)NW
710 FORMAT ( ' W(N'I1')P[180,262]V[240,362]' )
ELSE
WRITE(9,720)NW
720 FORMAT ( ' W(N'I1')P[180,262]V[240,162]' )
END IF
WRITE(9,730)NW
730 FORMAT ( ' W(N'I1')P[630,162]V[630,167]V[640,162]' )
WRITE(9,735)
735 FORMAT ( ' P[630,182]V[630,177]V[640,182]' )
WRITE(9,740)NW
740 FORMAT ( ' W(N'I1')P[630,362]V[630,367]V[640,362]' )
WRITE(9,750)
750 FORMAT ( ' P[630,382]V[630,377]V[640,382]' )
IF ( NBED.EQ. 1 ) THEN
WRITE(9,800)NW
800 FORMAT ( ' W(N'I1')P[180,282]V[240,382]' )
WRITE(9,805)
805 FORMAT ( ' P[240,162]V[180,162]P[240,182]V[180,182]' )
WRITE(9,810)NW
810 FORMAT ( ' W(N'I1')P[113,272]V[143,272]' )
WRITE(9,815)
815 FORMAT ( ' P[136,268]V[]V[143,272]V[136,276]' )
WRITE(9,820)NW
820 FORMAT ( ' W(N'I1')P[590,372]V[625,372]' )
WRITE(9,825)
825 FORMAT ( ' P[620,369]V[]V[625,372]V[620,375]' )
WRITE(9,830)NW

```

```

830 FORMAT ( ' W(N'I1')P[610,320]V[610,290]' )
    WRITE(9,835)
835 FORMAT ( ' P[607,295]V[V[610,290]V[613,295]' )
    WRITE(9,840)NW
840 FORMAT ( ' W(N'I1')P[610,172]V[585,172]' )
    WRITE(9,845)
845 FORMAT ( ' P[590,169]V[V[585,172]V[590,175]' )
    WRITE(9,850)NW
850 FORMAT ( ' W(N'I1')P[128,352]V[128,392]' )
    WRITE(9,855)
855 FORMAT ( ' P[124,385]V[V[128,392]V[132,385]' )
    WRITE(9,860)NW
860 FORMAT ( ' W(N'I1')P[638,172]C[632,166]' )
    WRITE(9,865)
865 FORMAT ( ' P[645,172]V11117777111177771111000' )
    WRITE(9,870)NW
870 FORMAT ( ' W(N'I1')P[650,372]C[643,372]' )
    WRITE(9,880)
880 FORMAT ( ' P[657,372]V11117777111100' )
    ELSE
    WRITE(9,900)NW
900 FORMAT ( ' W(N'I1')P[180,282]V[240,182]' )
    WRITE(9,905)
905 FORMAT ( ' P[180,362]V[240,362]P[180,382]V[240,382]' )
    WRITE(9,910)NW
910 FORMAT ( ' W(N'I1')P[113,272]V[143,272]' )
    WRITE(9,915)
915 FORMAT ( ' P[136,268]V[V[143,272]V[136,276]' )
    WRITE(9,920)NW
920 FORMAT ( ' W(N'I1')P[590,172]V[615,172]' )
    WRITE(9,925)
925 FORMAT ( ' P[608,168]V[V[615,172]V[608,176]' )
    WRITE(9,930)NW
930 FORMAT ( ' W(N'I1')P[610,222]V[610,252]' )
    WRITE(9,935)
935 FORMAT ( ' P[606,245]V[V[610,252]V[614,245]' )
    WRITE(9,940)NW
940 FORMAT ( ' W(N'I1')P[610,372]V[580,372]' )
    WRITE(9,945)
945 FORMAT ( ' P[587,368]V[V[580,372]V[587,376]' )
    WRITE(9,950)NW
950 FORMAT ( ' W(N'I1')P[159,372]V[128,372]V[128,390]' )
    WRITE(9,955)
955 FORMAT ( ' P[124,381]V[V[128,390]V[132,381]' )
    WRITE(9,960)NW
960 FORMAT ( ' W(N'I1')P[650,172]C[643,172]' )
    WRITE(9,965)
965 FORMAT ( ' P[657,172]V11117777111100' )
    WRITE(9,970)NW
970 FORMAT ( ' W(N'I1')P[638,372]C[633,366]' )
    WRITE(9,980)
980 FORMAT ( ' P[645,372]V11117777111177770000' )
    END IF
    RETURN
    END
    SUBROUTINE REGISOUT
C SETS VT125 BACK INTO TEXT MODE
    LOGICAL ALPHA

```

```

    DIMENSION ALPHA(2)
    DATA ALPHA / 27, ' /
    TYPE 10, ALPHA
10  FORMAT ( '+', 5A1, $ )
    RETURN
    END
    SUBROUTINE DELAY ( NTICKS )
    TYPE 100, NTICKS
100 FORMAT ( '+S(T<'I3>)', $ )
    RETURN
    END
    SUBROUTINE SCALEM ( Y, IYABS, VV, RANGE )
    DY = (( VV - Y ) * 300.0 ) / RANGE
    IDY = IFIX ( DY )
    IYABS = IDY + 90
    RETURN
    END
    SUBROUTINE STIME ( TIME, IXABS, RANGE, AXLEN, SHIFT )
    INTEGER SHIFT
    DX = ( TIME * AXLEN ) / RANGE
    IDX = IFIX ( DX )
    IXABS = IDX + SHIFT
    RETURN
    END
    SUBROUTINE FRAME
    COMMON/SYST/DT, PSUP, POUT, TF, WBRL, VS, VO, IIN, TCYC
    COMMON/NPRINT/NOUT1
    DIMENSION T(10)
    DELT = TF/10.0
    DO I=1,10
    T(I)=I*DELT
    END DO
    TYPE 10
10  FORMAT ( 'IP[130,390]V[+60,]V[.+5]' , $ )
    DO I=1,9
    TYPE 12
12  FORMAT ( 'IP[.-5]V[+60,]V[.+5]' , $ )
    END DO
    TYPE 20
20  FORMAT ( 'IP[120,398]T"0.0"' , $ )
    TYPE 21 , T(1), T(2)
21  FORMAT ( 'IP[160,398]T"F5.1"P[220,398]T"F5.1"' , $ )
    TYPE 22 , T(3), T(4)
22  FORMAT ( 'IP[280,398]T"F5.1"P[340,398]T"F5.1"' , $ )
    TYPE 23 , T(5), T(6)
23  FORMAT ( 'IP[400,398]T"F5.1"P[460,398]T"F5.1"' , $ )
    TYPE 24 , T(7), T(8)
24  FORMAT ( 'IP[520,398]T"F5.1"P[580,398]T"F5.1"' , $ )
    TYPE 25 , T(9), T(10)
25  FORMAT ( 'IP[640,398]T"F5.1"P[700,398]T"F5.1"' , $ )
    TYPE 30
30  FORMAT ( 'IP[350,430]T"TIME (SEC.)"' , $ )
    TYPE 40
40  FORMAT ( 'IP[130,390]V[.-30]V[-5,]' , $ )
    DO I=1,9
    TYPE 50
50  FORMAT ( 'IP[+5,]V[.-30]V[-5,]' , $ )
    END DO

```

```

TYPE 51
51  FORMAT ( 'IP[70,180]T[+0,+16]"OUTPUT" , $ )
    TYPE 53 , PSUP
53  FORMAT ( 'IP[530,34]T[+9,+0]"PSUP = "T"F5.2"T" PSIA", $ )
    TYPE 54 , POUT
54  FORMAT ( 'IP[530,52]T"POUT = "T"F5.2"T" PSIA", $ )
    TYPE 55 , WBRL
55  FORMAT ( 'IP[530,70]T"WBRL = "T"F5.2"T" STD LIT/MIN", $ )
    IF(NOUT1.EQ.1) THEN
57  TYPE 60
60  FORMAT ( 'IP[290,5]T" OXYGEN MOLE FRACTION "" , $ )
    TYPE 62
62  FORMAT ( 'IP[95,383]T"0.0"P[95,353]T"0.1"P[95,323]T"0.2"$ )
    TYPE 64
64  FORMAT ( 'IP[95,293]T"0.3"P[95,263]T"0.4"P[95,233]T"0.5"$ )
    TYPE 66
66  FORMAT ( 'IP[95,203]T"0.6"P[95,173]T"0.7"P[95,143]T"0.8"$ )
    TYPE 68
68  FORMAT ( 'IP[95,113]T"0.9"P[95,83]T"1.0"$ )
    ELSE IF(NOUT1.EQ.2) THEN
    TYPE 80
80  FORMAT ( 'IP[290,5]T" INLET MASS FLOWRATE (KG/SEC) "" , $ )
    TYPE 82
82  FORMAT ( 'IP[85,383]T"-.02"P[85,353]T"-.01"P[95,323]T"0.0"$ )
    TYPE 84
84  FORMAT ( 'IP[95,293]T".01"P[95,263]T".02"P[95,233]T".03"$ )
    TYPE 86
86  FORMAT ( 'IP[95,203]T".04"P[95,173]T".05"P[95,143]T".06"$ )
    TYPE 88
88  FORMAT ( 'IP[95,113]T".07"P[95,83]T".08"$ )
    ELSE IF(NOUT1.EQ.3) THEN
    TYPE 90
90  FORMAT ( 'IP[290,5]T" OUTLET MASS FLOWRATE (KG/SEC) "" , $ )
    TYPE 82
    TYPE 84
    TYPE 86
    TYPE 88
    ELSE
    GO TO 57
    END IF
    RETURN
    END

```


APPENDIX C: ISOBANK Code and Manual

ISOBANK

(Databank for Adsorption Data)

In - Won Kim

Department of Chemical Engineering

University of Texas at Austin

Austin, TX 78712

I. ISOBANK.TXT

ISOBANK.TXT explains what is ISOBANK.

C
C
C *****

C
C Document
C
C Adsorption Data Bank (ISOARMY.FOR)
C
C 20 October 1987 In-Won Kim
C
C Revised on 29 January 1988

C
C *****

C
C
C Purpose

C This code supplies adsorption data of pure and
C multi component mixtures. This code can be used
C to design and study adsorption separation processes.

C
C
C Usage

C RUNISOARMY

C
C Applications

- C 1. To obtain parameter values of F-H VSM
C at given temperatures.
C
C 2. To obtain temperature independent parameter
C values of F-H VSM.
C
C 3. To calculate amount adsorbed of pure-component
C adsorbates on specified adsorbents using F-H VSM.
C
C 4. To calculate amount adsorbed of pure-component
C adsorbates on specified adsorbents using
C Langmuir isotherm model.

5. To calculate pure-component isotherms using F-H VSM.
6. To calculate pure-component isotherms using Langmuir isotherm model.
7. To calculate phase diagrams of binary component mixtures at given temperature and pressure using F-H VSM.
8. To calculate phase diagrams of ternary component mixtures at given temperature and pressure using F-H VSM (three dimensional plots)

Possible Adsorbent-Adsorbate Pairs (As of 20 October 1987)

Adsorbent	Adsorbate	Code	Ref.	
-----	-----	-----	-----	-----
RS-10	Oxygen		1 - 1	1
	Nitrogen	1 - 2	1	
Zeolite 5A	Oxygen	2 - 1	2	
	Nitrogen	2 - 2	2	
Activated Carbon	Ethane	3 - 3	3	
	Propane	3 - 4	3	
Nuxit AL Act. Car.	Methane	4 - 5	4	
	Carbon Dioxide	4 - 6	4	
	Acetylene	4 - 7		4
	Ethene	4 - 8	4	
	Ethane	4 - 3	4	
	Propene	4 - 9	4	
	Propane	4 - 4	4	
Silica	n-Butane	4 - 10		4
	Ethene	5 - 8	4	
	Propene	5 - 9	4	
	Propane	5 - 4	4	

C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C

Zeolite 10X	Oxygen	6 - 1	4
	Nitrogen	6 - 2	4
	Carbon Monoxide	6 - 11	4
Zeolite 13X	Ethene	7 - 8	4
	Ethane	7 - 3	4
	i-Butane	7 - 12	4
	Carbon Dioxide	7 - 6	4

References

1. Kim, I., McDonnell Douglas Report, July 15, 1987.
2. Miller
3. Huang, C. C., PhD Dissertation, UT, 1987.
4. Cochran, T. W., R. L. Kabel and R. P. Danner, AIChE J., Vol. 31, 268, 1985.

C
C Precision

C Double precision/Vax

C
C Comments

C If you have any questions about this code,
C fee! free to contact the author.

C Address : In-Won Kim
C Department of Chemical Engineering
C University of Texas at Austin
C Austin, TX 78712

C
C Library Required

C IMSL Library
C CONSYD Library

C
C Remarks

- C 1. The adsorption data are calculated based on Flory-Huggins
C Vacancy Solution Model (see Reference 4)
C 2. The parameter values are collected from the literatures
C (see References)
C 3. Some options are not available now. Do not choose these
C options. Program will say that sorry!!! not available now.
C - Binary component isotherm plots
C - Ternary component isotherm plots
C 4. Do not choose these adsorbent-adsorbate pairs:
C Code # 2 - 1, Zeolite 5A - Oxygen
C Code # 2 - 2, Zeolite 5A - Nitrogen
C These parameter values will be added in the near future.
C 5. In order to add your own parameter values, edit block data:
C Block data ANAME : Adsorbent Name
C Block data BNAME : Adsorbate Name
C Block Data ADATA : F-H VSM partameters
C 6. For the comparison of your own experiment data with

C the estimated data from F-H VSM, please refer to a modified
C code "VSMPARA.FOR"
C
C *****
C *****
C

II. Example run of ISOBANK.EXE

Example run shows how to run menu-driven code.

III. Example output of Pure component data

Langmuir isotherm model

- 1. Adsorbent and adsorbate name**
- 2. Langmuir parameter values at given temperatures**
- 3. Table of pure component isotherms**
- 4. Figure of isotherms at different temperatures**

Vacancy solution model

- 1. Adsorbent and adsorbate name**
- 2. F-H VSM parameter values at given temperatures**
- 3. Table of pure component isotherms**
- 4. Figure of isotherms at different temperatures**

IV. Example output of binary component data

- 1. Adsorbent and adsorbate names**
- 2. F-H VSM parameter values at given temperatures**
- 3. Table of composition data**
- 4. Figure of phase diagrams**
- 5. Figure of total amount adsorbed of mixture**

V. Example output of ternary component data

- 1. Adsorbent and adsorbate names**
- 2. F-H VSM parameter values at given temperatures**
- 3. Table of phase compositions**
- 4. Figure of phase diagrams, $x_1 - x_2 - y_1$**
- 5. Figure of phase diagrams, $x_1 - x_2 - y_2$**
- 6. Figure of total amount adsorbed w.r.t. x_1 and x_2**

VI. List of ISOARMY.FOR

```

C
C
C      PROGRAMISOARMY
C
C *****
C *****
C
C
C      IN-WON KIM
C      20 OCTOBER 1987
C      REVISED ON 7 FEBUARY 1988
C
C *****
C *****
C
C      THIS PROGRAM SUPPLIES ADSORPTION DATA TO THE USERS
C      TO DESIGN AND STUDY ADSORPTION SEPARATION PROCESSES.
C      IN THIS PROGRAM PLOTTING ROUTINES ARE DELETED. PLEASE
C      REFERTO ORIGINAL PROGRAM "ISOBANK.FOR"
C
C
C      DETAILS ARE IN FILE " ISOBANK.TXT "
C
C      OPTION      DESCRIPTION
C      -----
C
C      1      PURE-COMPONENT SYSTEM
C
C      2      BINARY COMPONENT SYSTEM
C
C      3      TERNARY COMPONENT SYSTEM
C
C      COMMONBLOCKS :NONE
C
C      REQUIRED ROUTINES: PURESYS, BINASYS, TERNSYS
C
C *****
C *****
C
C      CHARACTER*1 YES, NO
C      INTEGER NOPT
C      DATA YES, NO /'Y','N'/

```

```

C
C
OPEN (UNIT=6, FILE='ISOARMY.OUT', STATUS='UNKNOWN')
C
CALL HEADER (6, 'ISOBANK.OUT')
C
WRITE(*,100)
READ(*,101) NO
IF (NO.EQ.'N') STOP
1  CONTINUE
WRITE(*,102)
C
READ(*,*) NOPT
GO TO ( 2, 3, 4 ) NOPT
2  CALLPURESYS
GOTO5
C
3  CALL BINASYS
GOTO5
C
4  CALLTERNSYS
C
5  CONTINUE
WRITE(*,103)
READ(*,101) YES
IF ( YES .EQ. 'Y' ) GO TO 1
C
CLOSE ( UNIT=6 )
C
STOP
C
100  FORMAT(/////T5,'WELCOME TO'///
$      T5,' ** *** ***** ** ***** ***** ** * **/'
$      T5,'* * * * * * * * * * * * * * * * */'
$      T5,'* * * * * * * * * * * * * * * * */'
$      T5,'***** * * * * * * * * * * * * * * */'
$      T5,'* * *** ***** ** * * * * * * * * * * */'
$      ///
$      T5,'**** ** ***** ** *** ** * * * * */'
$      T5,'* * * * * * * * * * * * * * * * */'
$      T5,'* * * * * * * * * * * * * * * * */'
$      T5,'* * * * * * * * * * * * * * * * */'
$      T5,'*** * * * * * * * * * * * * * * */'
$      ///

```

```

$      T5,'DO YOU WANT TO CONTINUE? [ Y OR N ]')
101    FORMAT(A1)
102    FORMAT(////T5,'OPTIONS :')
$      T5,'1. PURE-COMPONENT SYSTEM'///
$      T5,'2. BINARY COMPONENT SYSTEM'///
$      T5,'3. TERNARY COMPONENT SYSTEM'///
$      T5,'ENTER YOUR CHOICE [ 1, 2 OR 3 ]')
103    FORMAT(////T5,'DO YOU WANT TO START AGAIN? [ Y OR N ]')
C
      END
C
C
C
      SUBROUTINEPURESYS
C
C *****
C *****
C
C      SUBROUTINE PURESYS   19 OCTOBER 1987   IN-WON KIM
C
C      THIS SUBROUTINE IS TO WRITE OPTIONS ON THE SCREEN FOR
C      PURE-COMPONENTSYTEM
C
C      OPTION      DESCRIPTION
C      -----
C
C      1      DATA FOR AMOUNT OF ADSORPTION AT
C              GIVEN TEMPERATURE AND PRESSURE
C
C      2      PURE-COMPONENT ISOTHERM PLOTS
C
C      COMMONBLOCKS :NONE
C
C      REQUIRIEDROUTINES:PUREDAT,PUTMPLT
C
C *****
C *****
C
C      INTEGERNOPT
C
C      WRITE(*,100)
C      READ(*,*) NOPT
C

```

```

      GO TO ( 1, 2 ) NOPT
1     CALL PUREDAT
      RETURN
C
2     CALL PUTMPLT
      RETURN
C
100    FORMAT(////T5,'OPTIONS :'////
$      T5,'1. DATA FOR AMOUNT OF ADSORPTION AT/'
$      T5,' GIVEN TEMPERATURE AND PRESSURE'///
$      T5,'2. PURE-COMPONENT ISOTHERM PLOT'////
$      T5,'ENTER YOUR CHOICE [ 1 OR 2 ]')
C
      END
C
C
C
C
      SUBROUTINEPUREDAT
C
C *****
C *****
C
C SUBROUTINE PUREDAT  4 FEBUARY 1988  IN-WON KIM
C
C THIS SUBROUTINE IS TO CALCULATE PURE-COMPONENT
C ADSORPTION DATA WITH 50 MAXIMUM DATA POINTS
C
C PARAMETER DESCRIPTION:
C
C NBENT  : ADSORBENT NO.
C NBATE  : ADSORBATE NO.
C TEMP   : TEMPERATURE [K]
C PRESS  : PRESSURE [KPA]
C ADSAMT : AMOUNT ADSORBED [KMOL/KG]
C NDATA  : NO. OF DATA POINT
C IOPT   : IF IOPT=1, USING VACANCY SOLUTION MODEL
C         IF IOPT=2, USING LANGMUIR MODEL
C
C COMMONBLOCKS      :NONE
C
C REQUIRIEDROUTINES: PURPAIR, PURECAL
C

```

```

C .....
C .....
C
  DOUBLE PRECISION TEMP(50), PRESS(50), ADSAMT(50),
  $      DUM1, DUM2
  INTEGER NBENT, NBATE, NDATA, IOPT
C
C.....TOCHOOSEISOTHERMMODEL
C
  WRITE(*,102)
  READ(*,*) IOPT
C
  CALL PURPAIR ( NBENT, NBATE )
C
  WRITE(*,*) 'HOW MANY POINTS DO YOU WANT TO CALCULATE?'
  WRITE(*,*) '  MAXIMUM 50 ADSORPTION DATA'
  READ(*,*) NDATA
C
  DO 10 I=1,NDATA
    WRITE(*, '/')
    WRITE(*,*) 'ENTER PRESSURE-TEMPERATURE PAIR',I
    WRITE(*,*) '          [ KPA, K ]'
    READ(*,*) PRESS(I), TEMP(I)
  10  CONTINUE
C
C.....TOGENERATEADSORPTIONAMOUNTDATA
C
  DO 20 I=1,NDATA
    DUM1 = TEMP(I)
    DUM2 = PRESS(I)
    CALL PURECAL (NBENT, NBATE, DUM1, DUM2, ADSAMT, IOPT)
  20  CONTINUE
C
  WRITE(*,100)
  DO 30 I=1,NDATA
    WRITE(*,101) PRESS(I),TEMP(I), ADSAMT(I)
  30  CONTINUE
C
  RETURN
C
100  FORMAT(///)
101  FORMAT(///
  $      T5,'TEMPERATURE', T20, 'PRESSURE', T35,

```



```

$      'AMOUNT ADSORBED'/
$      T5,' [ K ]', T20, ' [ KPA ]', T35, ' [KMOL/KG]'/
$      T5, 11(1H-), T20, 11(1H-), T35, 13(1H-)/
$      T5, F10.3, T20, F10.3, T35, E10.3/
102    FORMAT(///T5,'OPTIONS:'''
$      T5,'1. USING VACANCY SOLUTION MODEL'''
$      T5,'2. USING LANGMUIR MODEL'''
$      T5,' ENTER YOUR CHOICE [ 1 OR 2 ]''')
      END

```

C
C
C

SUBROUTINEPUTMPLT

C
C

C
C
C

SUBROUTINE PUTMPLT 19 OCTOBER 1987 IN-WON KIM

C
C
C

THIS SUBROUTINE IS TO PLOT PURE-COMPONENT ISOTHERMS
TO PLOT MAXIMUM 5 PLOTS SIMULTANEOUSLY.

C
C
C

PARAMETER DESCRIPTION:

C
C
C

NBENT : ADSORBENT NO.
NBATE : ADSORBATE NO.
TEMP : TEMPERATURE [K]
PRESS : PRESSURE [KPA]
ADSAMT : AMOUNT ADSORBED [KMOL/KG]
PMAX : MAXIMUM PRESSURE [KPA]
PMIN : MINIMUM PRESSURE [KPA]
NPOINT : NO. OF DATA POINT
IOPT : IF IOPT=1, USING VACANCY SOLUTION MODEL
IF IOPT=2, USING LANGMUIR MODEL.

C
C
C

COMMONBLOCKS :NONE

C
C
C

REQUIRIED ROUTINES: PURPAIR, PURETHM, MAPPLT, PURETM.PDL

C
C

C

DOUBLE PRECISION TEMP(5), PRESS(51), ADSAMT(5,51),

```

$                                PMIN, PMAX, DUM2(51)
    INTEGER NBENT, NBATE, NPOINT, IOPT
C
C.....TOCHOOSEISOTHERMMODEL
C
    WRITE(*,103)
    READ(*,*) IOPT
C
C.....TO CHOOSE ADSORBENT-ADSORBATE PAIR FROM THE DATA BANK
C
    CALL PURPAIR ( NBENT, NBATE )
C
    WRITE(*,*) 'HOW MANY ISOTHERMS DO YOU WANT TO PLOT?'
    WRITE(*,*) ' MAXIMUM 5 ISOTHERMS IN ONE PLOT'
    READ(*,*) NTEMP
C
    DO 10 I=1,NTEMP
        WRITE(*, '/')
        WRITE(*,*) 'TEMPERATURE IN DEGREE K, '
        READ(*,*) TEMP(I)
10    CONTINUE
    WRITE(*, '/')
    WRITE(*,*) 'PRESSURE RANGE IN KPA?'
    WRITE(*,*) ' MIN TO MAX  e.g.[ 0, 100 ]'
    READ(*,*) PMIN,PMAX
C
    WRITE(*, '/')
    WRITE(*,*) 'HOW MANY DATA POINTS DO YOU WANT TO PLOT?'
    WRITE(*,*) ' MAXIMUM 50 POINTS [ 50 ]'
    READ(*,*) NPOINT
C
C.....TOGENERATEPRESSURE-ADSORPTIONAMOUNTDATA
C
    DO 30 I=1,NTEMP
        WRITE(6,100) I, TEMP(I)
        DUM1 = TEMP(I)
        CALL PURETHM (NBENT, NBATE, DUM1, PMIN, PMAX,
$            NPOINT, PRESS, DUM2, IOPT)
        DO 20 J=1,NPOINT+1
            ADSAMT(I,J) = DUM2(J)*1000.
20        CONTINUE
30    CONTINUE
C

```

```

C.....TOUSECONSYD PLOTROUTINES
C
C      WRITE(6,101)
C
C      CALL ASSIGN(1,'PURETM.DAT')
C      DO 40 I=1,NPOINT+1
C          WRITE(1) PRESS(I),(ADSAMT(J,I),J=1,NTEMP)
C          WRITE(6,102) PRESS(I),(ADSAMT(J,I)/1000.,J=1,5)
C      40 CONTINUE
C      CALL CLOSE(1)
C
C      GO TO ( 50, 60, 70, 80, 90) NTEMP
C
C      50      CALL MAPPLT ('PURETM1')
C      RETURN
C
C      60      CALL MAPPLT ('PURETM2')
C      RETURN
C
C      70      CALL MAPPLT ('PURETM3')
C      RETURN
C
C      80      CALL MAPPLT ('PURETM4')
C      RETURN
C
C      90      CALL MAPPLT ('PURETM5')
C
C      RETURN
C
C      100      FORMAT(
C          $          T5,'TEMPERATURE',I3,2X,'IS',F10.3,'[ K ]'
C          $          )
C      101      FORMAT(
C          $          T5,'PURE-COMPONENT ISOTHERM DATA'//
C          $          T5,'PRESS. VS. AMT. ADSORBED'//
C          $          T5,'PRESS. IN KPA, AMOUNT ADSORBED IN KMOL/KG'//
C          $          T5,'PRESSURE',T15,' TEMP. 1',T25,' TEMP. 2',
C          $          T35,' TEMP. 3',T45,' TEMP. 4',T55,' TEMP. 5'//
C          $          )
C      102      FORMAT(T5,6E10.3/)
C      103      FORMAT(///T5,'OPTIONS:'///
C          $          T5,'1. USING VACANCY SOLUTION MODEL'///
C          $          T5,'2. USING LANGMUIR MODEL'///

```

```

$      T5,' ENTER YOUR CHOICE [ 1 OR 2 ]'//)
C
C      END
C
C
C
C      SUBROUTINE PURPAIR (NBENT,NBATE)
C      INTEGER NBENT,NBATE
C
C      .....
C      .....
C
C      SUBROUTINE PURPAIR 19 OCTOBER 1987 IN-WON KIM
C
C      THIS SUBROUTINE IS TO CHOOSE ADSORBENT-ADSORBATE PAIR
C      FROM THE DATA BANK.
C
C      PARAMETER DESCRIPTION:
C
C      NBENT : ADSORBENT NO. (SEE BLOCK DATA ANAME)
C      NBATE : ADSORBATE NO. (SEE BLOCK DATA BNAME)
C
C      COMMONBLOCKS :BATNAME,BETNAME
C
C      ROUTINEREQUIRED:NAME
C
C      .....
C      .....
C
C
C      CHARACTER*25 SORBENT(20), SORBATE(20)
C      COMMON/BATNAME/SORBATE
C      COMMON/BETNAME/SORBENT
C
C.....TO DISPLAY ADSORBENT-ADSORBATE PAIR ON THE SCREEN
C
C      CALL NAME
C
C      WRITE(*,*) 'ENTER ADSORBENT-ADSORBATE PAIR'
C      WRITE(*,*) ' e.g. RS-10 AND OXYGEN [ 1, 1 ]'
C      READ(*,*) NBENT, NBATE
C
C      WRITE(*,100) SORBENT(NBENT), SORBATE(NBATE)

```

```

C
  RETURN
C
100  FORMAT(///
      $    T5,'YOUR CHOICES ARE'///
      $    T8,'ADSORBENT : ',A25//
      $    T8,'ADSORBATE : ',A25//
      $    )
C
  END
C
C
C
  SUBROUTINENAME
C
C *****
C *****
C
C  SUBROUTINE NAME 19 OCTOBER 1987 IN-WON KIM
C
C  THIS SUBROUTINE IS TO WRITE A POSSIBLE PAIR OD
ADSORBENT-
C  ADSORBATEONTHESCREEN.
C
C  PARAMETERDESCRIPTION:
C
C  SORBENT : NAME OF ADSORBENT
C  SORBATE : NAME OF ADSORBATE
C
C  COMMONBLOCKS      :BATNAME,BETNAME
C
C  REQUIRIEDROUTINES:NONE
C
C *****
C *****
C
C  CHARACTER*25 SORBENT(20), SORBATE(20)
C  CHARACTER*1 NO
C  COMMON/BETNAME/SORBENT
C  COMMON/BATNAME/SORBATE
C  DATA NO/'N'/
C
C  WRITE(*,100)

```

```

WRITE(*,101) SORBENT(1), SORBATE(1), SORBATE(2)
WRITE(*,101) SORBENT(2), SORBATE(1), SORBATE(2)
WRITE(*,101) SORBENT(3), SORBATE(3), SORBATE(4)
WRITE(*,101) SORBENT(4), SORBATE(5), SORBATE(6),
SORBATE(7),
$           SORBATE(8), SORBATE(3), SORBATE(9),
$           SORBATE(4), SORBATE(10)
WRITE(*,*) 'DO YOU WANT TO CONTINUE ? [ Y OR N ]'
READ(*, '(A1)') NO
IF(NO.EQ.'N')RETURN
WRITE(*,101) SORBENT(5), SORBATE(8), SORBATE(9),
SORBATE(4)
WRITE(*,101) SORBENT(6), SORBATE(1), SORBATE(2),
SORBATE(11)
WRITE(*,101) SORBENT(7), SORBATE(8), SORBATE(3),
SORBATE(12),
$           SORBATE(6)
C
C   RETURN
C
C   100   FORMAT(///T11,'ADSORBENT', T36,'ADSORBATE'//
$         T5,20(1H-), T30, 20(1H-))
C   101   FORMAT(//T5, A25, T30, A25/, (T30, A25))
C
C   END
C
C
C
C   SUBROUTINE PURETHM (NBENT, NBATE, TEMP, PMIN, PMAX,
$                     NPOINT, PRES, AMOUNT, IOPT)
C   DOUBLE PRECISION TEMP, PMIN, PMAX, PRES(51), AMOUNT(51)
C   INTEGER           NBENT, NBATE, NPOINT, IOPT
C
C *****
C *****
C
C   SUBROUTINE PURETHM 19 OCTOBER 1987 IN-WON KIM
C
C   THIS SUBROUTINE IS TO CALCULATE PURE-COMPONENT
C   ISOTHERM.
C
C   PARAMETER DESCRIPTION :
C

```

```

C      NBENT : ADSORBENT NO.
C      NBATE : ADSORBATE NO.
C      TEMP  : TEMPERATURE [K]
C      PMIN  : MINIMUM PRESSURE [KPA]
C      PMAX  : MAXIMUM PRESSURE [KPA]
C      PAR(1) : PRESSURE, PRES [KPA]
C      PAR(2) : HENRY'S LAW CONSTANT, B [KMOL/KG/KPA]
C      PAR(3) : LIMITING AMOUNT ADSORBED, NI [KMOL/KG]
C      PAR(4) : NONIDEALITY PARAMETER, ALPHA [-]
C      X(1)   : N1, MOLES OF COMPONENT 1 ADSORBED
C              [KMOL/KG]
C      IOPT   : IF IOPT=1, USING VACANCY SOLUTION MODEL
C              IF IOPT=2, USING LANGMUIR MODEL
C
C      COMMONBLOCKS      :NONE
C
C      REQUIRROUTINES:ZSPOW,PUREFCN
C
C      *****
C      *****
C
C      DOUBLE PRECISION X(1), WK(10), FNORM, PAR(4), DELP,
LANGPAR
      INTEGER          N, NSIG, ITMAX, IER
      EXTERNALPUREFCN
C
C.....TO CALCULATE PARAMETER VALUES FOR VSM FOR GIVEN
C.....ADSORBATE AND ADSORBENT
C
      CALL PARACAL (NBENT, NBATE, TEMP, PAR(2), PAR(3), PAR(4))
      IF (IOPT.EQ.2) THEN
        PAR(4) = 0.0
        LANGPAR = PAR(2)/PAR(3)
        WRITE(6,101) LANGPAR
      ENDIF
C
C.....TO CALCULATE PRESSURE-AMOUNT ADSORBED DATA
C.....USING ZSPOW NONLINEAR ALGEBRAIC EQUATION SOLVER
C
      N = 1
      NSIG = 8
      ITMAX = 300
      DELP = (PMAX-PMIN)/DBLE(NPOINT)

```

```

PAR(1) = PMIN
X(1) = 1.0D-4
DO 10 I=1,NPOINT+1
  CALL ZSPOW(PUREFCN, NSIG, N, ITMAX, PAR,
$          X, FNORM, WK, IER)
  WRITE(*,100) I, IER, FNORM
  AMOUNT(I) = X(1)
  PRES(I) = PAR(1)
  PAR(1) = PAR(1) + DELP
10  CONTINUE
C
  RETURN
C
100  FORMAT(
$    (T5,I3,2X,'IER & FNORM',I3,2X,E11.4/)
$    )
101  FORMAT(///
$    T5,' PARAMETER FOR LANGMUIR ISOTHERM, B [ 1/KPA ] ',
$    E11.4///)
C
  END
C
C
C
C
  SUBROUTINE PURECAL (NBENT, NBATE, TEMP, PRES, AMOUNT,
$                    IOPT )
  DOUBLE PRECISION TEMP, PRES, AMOUNT
  INTEGER            NBENT, NBATE, IOPT
C
C *****
C *****
C
C  SUBROUTINE PURETHM 19 OCTOBER 1987 IN-WON KIM
C
C  THIS SUBROUTINE IS TO CALCULATE PURE-COMPONENT
C  ADSORPTION DATA.
C
C  PARAMETER DESCRIPTION :
C
C    NBENT : ADSORBENT NO.
C    NBATE : ADSORBATE NO.
C    TEMP  : TEMPERATURE [K]

```



```

C      PAR(1) : PRESSURE, PRES [KPA]
C      PAR(2) : HENRY'S LAW CONSTANT, B [KMOL/KG/KPA]
C      PAR(3) : LIMITING AMOUNT ADSORBED, NI [KMOL/KG]
C      PAR(4) : NONIDEALITY PARAMETER, ALPHA [-]
C      X(1)   : N1, MOLES OF COMPONENT 1 ADSORBED
C              [KMOL/KG]
C      IOPT   : IF IOPT=1, USING VACANCY SOLUTION MODEL
C              IF IOPT=2, USING LANGMUIR MODEL
C
C      COMMONBLOCKS      :NONE
C
C      REQUIRROUTINES:ZSPOW,PUREFCN
C
C      *****
C      *****
C
C      DOUBLE PRECISION X(1), WK(10), FNORM, PAR(4), LANGPAR
C      INTEGER          N, NSIG, ITMAX, IER
C      EXTERNALPUREFCN
C
C      C.....TOCALCULATE PARAMETER VALUES FOR VSM FOR GIVEN
C      C.....ADSORBATE AND ADSORBENT
C
C      CALL PARACAL (NBENT, NBATE, TEMP, PAR(2), PAR(3), PAR(4))
C      IF (IOPT.EQ.2) THEN
C        PAR(4) = 0.0
C        LANGPAR = PAR(2)/PAR(3)
C        WRITE(6,101) LANGPAR
C      ENDIF
C
C      C.....TO CALCULATE PRESSURE-AMOUNT ADSORBED DATA
C      C.....USING ZSPOW NONLINEAR ALGEBRAIC EQUATION SOLVER
C
C      N = 1
C      NSIG = 8
C      ITMAX = 300
C      PAR(1) = PRES
C      X(1) = 1.0D-4
C
C      CALL ZSPOW ( PUREFCN, NSIG, N, ITMAX, PAR,
C      $          X, FNORM, WK, IER )
C      AMOUNT = X(1)
C

```

```

WRITE(*,100) IER, FNORM
C
RETURN
C
100  FORMAT(//
$    T5,' IER AND FNORM IN PURECAL',I3,2X,E11.4
$    )
101  FORMAT(///
$    T5,' PARAMETER FOR LANGMUIR ISOTHERM, B [ 1/KPA ] ',
$    E11.4///)
C
END

C
C
C
SUBROUTINE PUREFCN ( X, F, N, PAR )
DOUBLE PRECISION X(N), F(N), PAR(4)
INTEGER          N

C
C *****
C *****
C
C   THIS SUBROUTINE IS TO CALCULATE THE AMOUNT OF
C   ADSORPTION
C   FROM GIVEN PRESSURES USING PURE-COMPONENT VSM MODEL.
C
C   PARAMETER DESCRIPOTION :
C
C   N1      : AMOUNT ADSORBED AT GIVEN TEEMPERATURE
C             AND PRESSURE [KMOL/KG]
C   THETA   : SURFACE COVERAGE [-]
C   PAR(1)  : PRESSURE, PRES [KPA]
C   PAR(2)  : HENRY'S LAW CONSTANT, B [KMOL/KG/KPA]
C   PAR(3)  : LIMITING AMOUNT ADSORBED, NI [KMOL/KG]
C   PAR(4)  : NONIDEALITY PARAMETER, ALPHA [-]
C
C   COMMONBLOCKS      :NONE
C
C   REQUIRIEDROUTINES:NONE
C
C *****
C *****

```

```

C
DOUBLE PRECISION N1, THETA, DUM1, DUM2
C
N1 = X(1)
THETA = N1/PAR(3)
DUM1 = PAR(3)/PAR(2)*THETA/(1.-THETA)
DUM2 = EXP(PAR(4)*PAR(4)*THETA/(1.+PAR(4)*THETA))
F(1) = PAR(1) -DUM1*DUM2
C
RETURN
END
C
C
C
SUBROUTINE PARACAL ( NBENT, NBATE, TEMP, B, NI, ALPHA )
DOUBLE PRECISION TEMP, B, NI, ALPHA
INTEGER      NBENT, NBATE
C
C *****
C *****
C
SUBROUTINE PARACAL  19 OCTOBER 1987  IN-WON KIM
C
THIS SUBROUTINE IS TO CALCULATE VSM PARAMETER
VALUES FROM TEMPERATURE INDEPENDENT VALUES.
C
PARAMETER DESCRIPTION :
C
NBATE      : ADSORBATE NO.
NBENT      : ADSORBENT NO.
TEMP       : TEMPERATURE [ K]
B          : HENRY'S LAW CONSTANT [kmol/kg/kpa]
NI         : LIMITING AMOUNT ADSORBED [kmol/kg]
ALPHA      : PARAMETER DESCRIBING NONIDEALITY [-]
R          : GAS CONSTANT [J/KMOL/K]
PARA(I,J,1) : TEMPERATURE INDEPENDENT CONSTANT
              FOR HERY'S LAW CONSTANT,
              Bio          [kmol/kg/kpa]
PARA(I,J,2) : TEMPERATURE INDEPENDENT CONSTANT
              FOR LIMITING AMOUNT ADSORBED,
              n1i         [kmol/kg]
PARA(I,J,3) : TEMPERATURE INDEPENDENT CONSTANT,
              CHARACTERISING EACH ADSORBATE-

```

```

C          ADSORBENT SYSTEM
C          ri,          [ K ]
C      PARA(I,J,4) : ISOSTERIC HEAT OF ADSORPTION,
C          q,          [J/kmol]
C      PARA(I,J,5) : TEMPERATURE INDEPENDENT CONSTANT
C          FOR PROPORTIONALITY,
C          m,          [kg/kmol]
C      I          : ADSORBENT (SEE BLOCK DATA ANAME)
C      J          : ADSORBATE (SEE BLOCK DATA BNAME)
C
C      COMMONBLOCKS      :PARAMET, BATNAME, BETNAME
C
C      REQUIREROUTINES:NONE
C
C      *****
C      *****
C
C      CHARACTER*25 SORBENT(20), SORBATE(20)
C      COMMON/BATNAME/SORBATE
C      COMMON/BETNAME/SORBENT
C      DOUBLE PRECISION PARA(20,20,5), R
C      COMMON/PARAMET/ PARA
C
C      I = NBENT
C      J = NBATE
C      R = 8314.
C      B      = PARA(I,J,1)*EXP(-PARA(I,J,4)/R/TEMP)
C      NI     = PARA(I,J,2)*EXP(PARA(I,J,3)/TEMP)
C      ALPHA  = PARA(I,J,5)*NI-1.
C
C      WRITE(6,100) SORBENT (NBENT), SORBATE (NBATE)
C      WRITE(6,101) TEMP, B, NI, ALPHA
C
C      RETURN
C
C      100      FORMAT(////
C      $          T5,'SORBENT NAME : ',A25//
C      $          T5,'SORBATE NAME : ',A25//)
C      101      FORMAT(
C      $          T5,'TEMPERATURE          [ K ]      ',E10.3//
C      $          T5,'HENRY LAW CONSTANT [KMOL/KG/KPA] ',E10.3//
C      $          T5,'LIMITING AMT. ADSORBED [KMOL/KG] ',E10.3//

```

```

$      T5,'NONIDEALITY PARAMETER [ - ]      :E10.3//
$      )
C
C      END
C
C
C
C      SUBROUTINEBINASYS
C
C      *****
C      *****
C
C      SUBROUTINE BINAYS  19 OCTOBER 1987 IN-WON KIM
C      This subroutine is to write option on the screen
C      for binary component system.
C
C      OPTION      DESCRIPTION
C      -----
C      1          Phase Diagram between gas phase mole
C                  fraction and gas phase mole fraction
C
C      2          Binary component isotherm plots
C
C      COMMONBLOCKS NONE
C
C      REQUROEDROUTINES:PHASDIG,BINAISO
C
C      *****
C      *****
C
C      INTEGERNOPT
C
C      WRITE(*,100)
C      READ(*,*) NOPT
C      GO TO ( 1, 2 ) NOPT
C
C      1  CALL PHASDIG
C         RETURN
C
C      2  CALL BINAISO
C         RETURN
C
C      100  FORMAT (///T5,'OPTIONS :////

```

```

$      T5,'1. PHASE DIAGRAM BETWEEN GAS PHASE MOLE'/
$      T5,' FRACTION AND SOLID PHASE MOLE FRACTION'///
$      T5,'2. BINARY COMPONENT ISOTHERM PLOTS'/
$      T5,'          --NOT AVAILABLE NOW-'///
$      T5,'ENTER YOUR CHOICE [ 1 OR 2 ]')
C
      END
C
C
C
      SUBROUTINE BINAISO
C
C *****
C *****
C
C      THIS SUBROUTINE IS TO CALCULATE BINARY-COMPONENT
C      ISOTHERMS
C
C *****
C *****
C
C      WRITE(*,100)
C
C      RETURN
C
100      FORMAT(///
$          T5,'SORRY!!!! NOT AVAILABLE NOW!!!'
$          )
C
      END
C
C
C
      SUBROUTINE PHASDIG
C
C *****
C *****
C
C      SUBROUTINE PHASDIG 21 OCTOBER 1987 IN-WON KIM
C
C      THIS SUBROUTINE IS TO PLOT PHASEDIGGRAM OF
C      BINARY COMPONENT SYSTEM
C

```

```

C    PARAMETER DESCRIPTION :
C
C    NBENT : ADSORBENT NO.
C    NBATE : ADSORBATE NO.
C    TEMP  : TEMPERATURE [K]
C    PRESS : PRESSURE [KPA]
C
C    COMMONBLOCKS      :NONE
C
C    REQUIRED ROUTINES : BINPAIR, BINACAL
C
C    *****
C    *****
C
C    DOUBLEPRECISIONTEMP,PRESS
C    INTEGERNBENT,NBATE(2)
C
C.....TOCHOOSEADSORBENT-ADSORBATEPAIRS
C
C    CALL BINPAIR ( NBENT, NBATE )
C
C.....TOINPUTPRESSUREANDTEMPERATUREDATA
C
C    WRITE(*,*) 'ENTER PRESSURE-TEMPERATURE PAIR'
C    WRITE(*,*) '          [ KPA, K ]'
C    READ(*,*) PRESS, TEMP
C
C.....TOGENERATESOLIDPHASEMOLEFRACTIONSFROMX1
C
C    CALL BINACAL (NBENT, NBATE, TEMP, PRESS)
C
C    RETURN
C
C    100    FORMAT(///)
C
C    END
C
C
C
C
C    SUBROUTINE BINPAIR ( NBENT, NBATE )
C    INTEGERNBENT,NBATE(2)
C

```

```

C *****
C *****
C
C SUBROUTINE PURPAIR 21 OCTOBER 1987 IN-WON KIM
C
C THIS SUBROUTINE IS TO CHOOSE ADSORBENT - ADSORBATE PAIR
C FROM THE DATA BANK.
C
C PARAMETER DESCRIPTION:
C
C NBENT : ADSORBENT NO. (SEE BLOCK DATA ANAME)
C NBATE : ADSORBATE NO. (SEE BLOCK DATA BNAME)
C
C COMMONBLOCKS : BATNAME, BETNAME
C
C ROUTINEREQUIRED : NAME
C *****
C *****
C
C CHARACTER*25 SORBENT(20), SORBATE(20)
C COMMON/BATNAME/ SORBATE
C COMMON/BETNAME/ SORBENT
C
C ..... TO DISPLAY ADSORBENT-ADSORBATE PAIR ON THE SCREEN
C
C CALL NAME
C
C WRITE(*,*) 'ENTER ADSORBENT NUMBER [ 1 ]'
C READ(*,*) NBENT
C WRITE(*,*) 'ENTER ADSORBATE 1 [ 1 ]'
C READ(*,*) NBATE(1)
C WRITE(*,*) 'ENTER ADSORBATE 2 [ 1 ]'
C READ(*,*) NBATE(2)
C
C WRITE(*,100) SORBENT(NBENT), SORBATE(NBATE(1)),
C $ SORBATE(NBATE(2))
C
C RETURN
C
C 100 FORMAT(///
C $ T5,'YOUR CHOICES ARE'///
C $ T8,'ADSORBENT : ',A25//

```



```

$      T8,'ADSORBATE 1 : ',A25//
$      T8,'ADSORBATE 2 : ',A25//
$      )
C
C      END
C
C
C
C      SUBROUTINE BINACAL ( NBENT, NBATE, TEMP, PRES )
C      DOUBLEPRECISIONTEMP,PRES
C      INTEGER              NBENT, NBATE(2)
C
C      .....
C      .....
C
C      SUBROUTINE BINACAL 21 OCTOBER 1987 IN-WON KIM
C
C      THIS SUBROUTINE IS TO CALCULATE BINARY-COMPONENT
C      ADSORPTION DATA.
C
C      PARAMETER DESCRIPTION :
C
C      NBENT  : ADSORBENT NO.
C      NBATE  : ADSORBATE NO.
C      TEMP   : TEMPERATURE [K]
C      NM     : TOTAL MOLES ADSORBED MIXTURE [KMOL/KG]
C      PAR(1) : X1, MOLE FRACTION OF COMPONENT 1
C              IN ADSORBED MIXTURE
C      PAR(2) : B1, HENRY'S CONSTANT OF COMPONENT 1
C              [KMOL/KG/KPA]
C      PAR(3) : B2, HENRY'S CONSTANT OF COMPONENT 2
C              [KMOL/KG/KPA]
C      PAR(4) : NI1, LIMITING AMOUNT ADSORBED OF PURE
C              COMPONENT 1 [KMOL/KG]
C      PAR(5) : NI2, LIMITING AMOUNT ADSORBED OF PURE
C              COMPONENT 2 [KMOL/KG]
C      PAR(6) : ALPHA1, NONIDEALITY PARAMETER OF COMPONENT
C              1
C              [ - ]
C      PAR(7) : ALPHA2, NONIDEALITY PARAMETER OF COMPONENT
C              2
C              [ - ]
C      PAR(8) : R1S, ACTIVITY COEFFICIENT OF COMPONENT 1

```

```

C          IN SURFACE PHASE
C    PAR(9)  : NMI, LIMITING AMOUNT ADSORBED OF MIXTURE
C              [KMOL/KG]
C    PAR(10) : RVS, ACTIVITY COEFFICIENT OF VACANCY
C              N SURFACE PHASE
C    PAR(11) : XVS, MOLE FRACTION OF VACANCY
C              REPRESENTING ADSORBED PHASE
C    PAR(12) : PRES,EQUILIBRIUM ADSORPTION PRESSURE
C              [ KPA ]
C
C    COMMONBLOCKS      :NONE
C
C    REQUIRIED ROUTINES : ZSPOW, TLBIFCN, MAPPLT, PHADIG.PDL
C
C *****
C *****
C
C    CHARACTER*1 NO
C    DOUBLE PRECISION X(1), WK(10), FNORM, PAR(12), NM(21),
C    $              X1(21), X2(21), Y1(21), Y2(21),
C    $              DUM1, DUM2, DUM3
C    INTEGER      N, NSIG, ITMAX, IER
C    EXTERNAL TLBIFCN
C    DATA NO /'N'/
C
C
C.....TO CALCURATE PARAMETER VALUES FOR VSM FOR GIVEN
C.....ADSORBATE AND ADSORBENT
C
C    CALL PARACAL (NBENT, NBATE(1), TEMP, PAR(2),PAR(4),
C    PAR(6))
C    CALL PARACAL (NBENT, NBATE(2), TEMP, PAR(3),PAR(5),
C    PAR(7))
C
C.....TO CALCULATE MOLE FRACTION OF SOLID PHASE
C.....USING ZSPOW NONLINEAR ALGEBRAIC EQUATION SOLVER
C
C    N = 1
C    NSIG = 8
C    ITMAX = 300
C    PAR(12) = PRES
C    PAR(1) = 0.0
C    X(1) = 3.0D-3
C

```

```

DO 10 I=1,21
C
  CALL ZSPOW ( TLBIFCN, NSIG, N, ITMAX, PAR,
$             X, FNORM, WK, IER )
  WRITE(*,100) I, IER, FNORM
C
C.....TO CALCULATE VAPOR PHASE MOLE FRACTION
C
  NM(I) = X(1)
  DUM1 = NM(I)
  CALL YCOMPBI (
$             PAR(1), DUM1, PAR(6), PAR(2),
$             PAR(4), PAR(8), PAR(9), PAR(10),
$             PAR(11), PRES, DUM2, DUM3
$             )
  Y1(I) = DUM2
  Y2(I) = DUM3
  X1(I) = PAR(1)
  X2(I) = 1. - X1(I)
  PAR(1) = PAR(1) + 0.05
10 CONTINUE
C
  WRITE(6,102) (X1(I), Y1(I), X2(I), Y2(I), NM(I), I=1,21)
C
C.....TO PLOT PHASE DIAGRAM WITH CONSYD
C
C  WRITE(*,*) 'DO YOU WANT TO SEE PHASE DIAGRAMS? [Y OR N]'
C  READ(*, '(A1)') NO
C  IF(NO.EQ.'N') RETURN
C
C  CALL ASSIGN(1, 'PHADIG.DAT')
C  DO 20 I=1,21
C    WRITE(1) X1(I), Y1(I), X2(I), Y2(I), NM(I)
C 20 CONTINUE
C  CLOSE(1)
C
C
C 80 CONTINUE
C  WRITE(*,101)
C  READ(*,*) NOPT
C
C  GO TO ( 30, 40, 50, 60 ) NOPT

```

```

C 30 CALL MAPPLT ( 'PHADIG1' )
C    GO TO 70
C
C 40 CALL MAPPLT ( 'PHADIG2' )
C    GO TO 70
C
C 50 CALL MAPPLT ( 'PHADIG3' )
C    GO TO 70
C
C 60    CALL MAPPLT ( 'PHADIG4' )
C
C 70 WRITE(*,*) 'DO YOU WANT TO SEE ANOTHER PLOT? [ Y OR N ]'
C    READ(*, '(A1)') NO
C    IF( NO.EQ.'N') GO TO 90
C    GO TO 80
C
C 90 RETURN
C
100  FORMAT(
$    (T5,I3,2X,'IER & FNORM',I3,2X,E11.4/)
$    )
101  FORMAT(///
$    T5,'1. PHASE DIAGRAM X1 - Y1'///
$    T5,'2. PHASE DIAGRAM X2 - Y2'///
$    T5,'3. PHASE DIAGRAM X1 - Y1 - Y2'///
$    T5,'4. X1 VS. AMT. ADSORBED OF MIX.'//
$    )
102  FORMAT(///
$    T5,'COMPOSITION DATA'//T3,18(1H-)//
$    T5,' X1',T15,' Y1',T25,' X2',T35,' Y2',
$    T45,'AMT. ADS. OF MIX.[KMOL/KG]'//
$    (T5,4F10.3,E10.3/)
C
C    END
C
C
C
C    SUBROUTINE TLBIFCN ( X, F, N, PAR )
C    DOUBLE PRECISION X(*), F(*), PAR(*)
C    INTEGER          N
C
C *****
C *****

```

```

C
C SUBROUTINE TLBIFCN 19 OCTOBER 1987 IN-WON KIM
C
C SUBROUTINE IS TO CALCULATE TOTAL MOLES ADSORBED
C OFBINARY MIXTURE
C
C PARAMETER LIST :
C
C NM : TOTAL MOLES ADSORBED MIXTURE [KMOL/KG]
C PAR(1) : X1, MOLE FRACTION OF COMPONENT 1
C IN ADSORBED MIXTURE
C PAR(2) : B1, HENRY'S CONSTANT OF COMPONENT 1
C [KMOL/KG/KPA]
C PAR(3) : B2, HENRY'S CONSTANT OF COMPONENT 2
C [KMOL/KG/KPA]
C PAR(4) : NI1, LIMITING AMOUNT ADSORBED OF PURE
C COMPONENT 1 [KMOL/KG]
C PAR(5) : NI2, LIMITING AMOUNT ADSORBED OF PURE
C COMPONENT 2 [KMOL/KG]
C PAR(6) : ALPHA1, NONIDEALITY PARAMETER OF COMPONENT 1
C [ - ]
C PAR(7) : ALPHA2, NONIDEALITY PARAMETER OF COMPONENT 2
C [ - ]
C PAR(8) : R1S, ACTIVITY COEFFICIENT OF COMPONENT 1
C IN SURFACE PHASE
C PAR(9) : NMI, LIMITING AMOUNT ADSORBED OF MIXTURE
C [KMOL/KG]
C PAR(10): RVS, ACTIVITY COEFFICIENT OF VACANCY
C IN SURFACE PHASE
C PAR(11): XVS, MOLE FRACTION OF VACANCY
C REPRESENTING ADSORBED PHASE
C PAR(12): PRES,EQUILIBRIUM ADSORPTION PRESSURE
C [ KPA ]
C
C COMMONBLOCKS :NONE
C
C REQUIREDROUTINES:NONE
C
C *****
C *****
C
C DOUBLE PRECISION X1, X2, NM, NMI, X1S, X2S, XVS,
$ A12, A21, A1V, A2V, ALNR1S, ALNR2S, ALNRVS,

```

```

$      R1S, R2S, RVS, F1, F2, PRES, B1, B2,
$      N1I, N2I
C
  X1 = PAR(1)
  B1 = PAR(2)
  B2 = PAR(3)
  N1I = PAR(4)
  N2I = PAR(5)
  A1V = PAR(6)
  A2V = PAR(7)
  PRES = PAR(12)
C
  X2 = 1. - X1
  NM = X(1)
C
C.....TO MAKE SURE NM IS LESS THAN NMI
C
  NMI=X1*N1I+X2*N2I
  IF(NM.GT.NMI) THEN
    NM=NMI*.99999
  END IF
C
  X1S=NM*X1/NMI
  X2S=NM*X2/NMI
  XVS=1.-NM/NMI
  A12=(A1V+1.)/(A2V+1.)-1.
  A21=(A2V+1.)/(A1V+1.)-1.
C
C
  ALNR1S=-LOG(X1S+X2S/(1.+A12)+XVS/(1.+A1V))
$      +1.-((X1S+X2S/(1.+A12)+XVS/(1.+A1V))**(-1)
  ALNR2S=-LOG((1.+A12)*X1S+X2S+XVS/(1.+A2V))
$      +1.-((1.+A12)*X1S+X2S+XVS/(1.+A2V))**(-1)
  ALNRVS=-LOG((1.+A1V)*X1S+(1.+A2V)*X2S+XVS)
$      +1.-((1.+A1V)*X1S+(1.+A2V)*X2S+XVS)**(-1)
C
C.....TO CALCULATE ACTIVITY COEFFICIENTS
C
  R1S = EXP(ALNR1S)
  R2S = EXP(ALNR2S)
  RVS = EXP(ALNRVS)
C
C

```

```

      F1 = R1S*X1*NM/NMI*N1I/B1*EXP(A1V)/(1.+A1V)
$      *EXP(((N1I-NMI)/NM-1.)*LOG(RVS*XVS))
      F2 = R2S*X2*NM/NMI*N2I/B2*EXP(A2V)/(1.+A2V)
$      *EXP(((N2I-NMI)/NM-1.)*LOG(RVS*XVS))
      F(1)=PRES-F1-F2
C
C.....PARAMETERCONVERSION
C
      PAR(8) = R1S
      PAR(9) = NMI
      PAR(10) = RVS
      PAR(11) = XVS
C
      RETURN
      END
C
C
C
C
      SUBROUTINE YCOMPBI (X1, NM, A1V, B1, N1I, R1S,
$                        NMI, RVS, XVS, PRES, Y1, Y2)
      DOUBLE PRECISION X1, Y1, Y2, R1S, NM, NMI, N1I,
$                        B1, A1V, RVS, XVS, PRES
C
C*****
C*****
C
C      SUBROUTINE YCOMPBI  19 OCTOBER 1987 IN-WON KIM
C
C      THIS SUBROUTINE IS TO CALCULATE MOLE FRACTION OF GAS
PHASE
C      IN BINARY MIXTURE
C
C      PARAMETER DESCRIPTION:
C
C      NM   : TOTAL MOLES ADSORBED MIXTURE [KMOL/KG]
C      X1   : MOLE FRACTION OF COMPONENT 1
C             IN ADSORBED MIXTURE
C      B1   : HENRY'S CONSTANT OF COMPONENT 1
C             [KMOL/KG/KPA]
C      N1I  : LIMITING AMOUNT ADSORBED OF PURE
C             COMPONENT 1 [KMOL/KG]
C      ALPHA1 : NONIDEALITY PARAMETER OF COMPONENT 1
C             [ - ]

```

```

C      R1S : ACTIVITY COEFFICIENT OF COMPONENT 1
C          IN SURFACE PHASE
C      NMI : LIMITING AMOUNT ADSORBED OF MIXTURE
C          KMOL/KG]
C      RVS : ACTIVITY COEFFICIENT OF VACANCY
C          IN SURFACE PHASE
C      XVS : MOLE FRACTION OF VACANCY
C          REPRESENTING ADSORBED PHASE
C      PRES : EQUILIBRIUM ADSORPTION PRESSURE
C          [ KPA ]
C      Y1 : GAS PHASE MOLE FRACTION OF COMPONENT 1
C          [ - ]
C      Y2 : GAS PHASE MOLE FRACTION OF COMPONENT 2
C          [ - ]
C
C      COMMONBLOCKS :NONE
C
C      REQUIRIEDROUTINES:NONE
C
C *****
C *****
C
C      Y1 = R1S*X1*NM/NMI*N1I/B1*EXP(A1V)/(1.+A1V)
C      $ *EXP(((N1I-NMI)/NM-1.)*LOG(RVS*XVS))/PRES
C      Y2 = 1.-Y1
C
C      RETURN
C      END
C
C      SUBROUTINETERNYSYS
C
C *****
C *****
C
C      SUBROUTINE TERNYS 26 OCTOBER 1987 IN-WON KIM
C      This subroutine is to write option on the screen
C      for binary component system.
C
C      OPTION      DESCRIPTION
C      -----
C      1          Phase Diagram between gas phase mole

```



```

C          fraction and gas phase mole fraction
C
C          2          TERNARY component isotherm plots
C
C      COMMONBLOCKS NONE
C
C      REQUROEDROUTINES:PHDITEN,TERNISO
C
C      *****
C      *****
C
C      INTEGERNOPT
C
C      WRITE(*,100)
C      READ(*,*) NOPT
C      GO TO ( 1, 2) NOPT
C
C      1  CALL PHDITEN
C      RETURN
C
C      2  CALL TERNISO
C      RETURN
C
C      100  FORMAT (///T5,'OPTIONS :'////
C      $    T5,'1. PHASE DIAGRAM BETWEEN GAS PHASE MOLE/'
C      $    T5,' FRACTION AND SOLID PHASE MOLE FRACTION'///
C      $    T5,'2. TERNARY COMPONENT ISOTHERM PLOTS/'
C      $    T5,' -NOT AVAILABLE NOW'///
C      $    T5,'ENTER YOUR CHOICE [ 1 OR 2 ]')
C
C      END
C
C
C
C      SUBROUTINETERNISO
C
C      *****
C      *****
C
C      THIS SUBROUTINE IS TO CALCULATE TERNARY COMPONENT
C      ISOTHERM DATA
C

```

```

C *****
C *****
C
C   WRITE(*,100)
C
C   RETURN
C
C 100   FORMAT(///
C      $      T5,'SORRY!!!! NOT AVAILABLE NOW!!!'
C      $      )
C
C   END
C
C
C
C   SUBROUTINEPHDITEN
C
C *****
C *****
C
C   SUBROUTINE PHASDIG 21 OCTOBER 1987 IN-WON KIM
C
C   THIS SUBROUTINE IS TO PLOT PHASEDIGGRAM OF
C   BINARYCOMPONENTSYSTEM
C
C   PARAMETERDESCRIPTION:
C
C   NBENT : ADSORBENT NO.
C   NBATE : ADSORBATE NO.
C   TEMP  : TEMPERATURE [K]
C   PRESS : PRESSURE [KPA]
C
C   COMMONBLOCKS      :NONE
C
C   REQUIRIEDROUTINES:TERNCAL
C
C *****
C *****
C
C   DOUBLEPRECISIONTEMP,PRESS
C   INTEGERNBENT,NBATE(3)
C
C.....TOCHOOSEADSORBENT-ADSORBATEPAIRS

```

```

C
  CALL TENPAIR ( NBENT, NBATE )
C
C.....TO INPUT PRESSURE AND TEMPERATURE DATA
C
  WRITE(*,*) 'ENTER PRESSURE-TEMPERATURE PAIR'
  WRITE(*,*) '          [ KPA, K ]'
  READ(*,*) PRESS, TEMP
C
C.....TO GENERATE SOLID PHASE MOLE FRACTIONS
C
  CALL TERNCAL (NBENT, NBATE, TEMP, PRESS)
C
  RETURN
C
100  FORMAT(///)
C
  END
C
C
C
C
  SUBROUTINE TENPAIR ( NBENT, NBATE )
  INTEGER NBENT, NBATE(3)
C
C *****
C ***** C
C   SUBROUTINE PURPAIR 26 OCTOBER 1987 IN-WON KIM
C
C   THIS SUBROUTINE IS TO CHOOSE ADSORBENT-ADSORBATE PAIR
C   FROM THE DATA BANK.
C
C   PARAMETER DESCRIPTION:
C
C   NBENT : ADSORBENT NO. (SEE BLOCK DATA ANAME)
C   NBATE : ADSORBATE NO. (SEE BLOCK DATA BNAME)
C
C   COMMON BLOCKS : BATNAME, BETNAME
C
C   ROUTINE REQUIRED: NAME
C
C *****
C *****
C

```

```

      CHARACTER*25 SORBENT(20), SORBATE(20)
      COMMON/BATNAME/ SORBATE
      COMMON/BETNAME/ SORBENT
C
C.....TO DISPLAY ADSORBENT-ADSORBATE PAIR ON THE SCREEN
C
      CALL NAME
C
      WRITE(*,*) 'ENTER ADSORBENT NUMBER [ 1 ]'
      READ(*,*) NBENT
      WRITE(*,*) 'ENTER ADSORBATE 1 [ 1 ]'
      READ(*,*) NBATE(1)
      WRITE(*,*) 'ENTER ADSORBATE 2 [ 1 ]'
      READ(*,*) NBATE(2)
      WRITE(*,*) 'ENTER ADSORBATE 3 [ 1 ]'
      READ(*,*) NBATE(3)
C
      WRITE(*,100) SORBENT(NBENT), SORBATE(NBATE(1)),
$              SORBATE(NBATE(2)), SORBATE(NBATE(3))
C
      RETURN
C
100  FORMAT(///
$      T5,'YOUR CHOICES ARE'///
$      T8,'ADSORBENT   : ',A25//
$      T8,'ADSORBATE 1 : ',A25//
$      T8,'ADSORBATE 2 : ',A25//
$      T8,'ADSORBATE 3 : ',A25//
$      )
C
      END
C
C
C
C
      SUBROUTINE TERNCAL ( NBENT, NBATE, TEMP, PRES )
      DOUBLEPRECISIONTEMP, PRES
      INTEGER NBENT, NBATE(3)
C
C *****
C *****
C
C      SUBROUTINE TERNCAL 26 OCTOBER 1987 IN-WON KIM
C

```

C THIS SUBROUTINE IS TO CALCULATE PURE-COMPONENT
 C ADSORPTION DATA.
 C
 C PARAMETER DESCRIPTION :
 C
 C NBENT : ADSORBENT NO.
 C NBATE : ADSORBATE NO.
 C TEMP : TEMPERATURE [K]
 C NM : TOTAL MOLES ADSORBED MIXTURE [KMOL/KG]
 C PAR(1) : X1, MOLE FRACTION OF COMPONENT 1
 C IN ADSORBED MIXTURE
 C PAR(2) : X2, MOLE FRACTION OF COMPONENT 2
 C IN ADSORBED MIXTURE
 C PAR(3) : B1, HENRY'S CONSTANT OF COMPONENT 1
 C [KMOL/KG/KPA]
 C PAR(4) : B2, HENRY'S CONSTANT OF COMPONENT 2
 C [KMOL/KG/KPA]
 C PAR(5) : B3, HENRY'S CONSTANT OF COMPONENT 3
 C [KMOL/KG/KPA]
 C PAR(6) : N1I, LIMITING AMOUNT ADSORBED OF PURE
 C COMPONENT 1 [KMOL/KG]
 C PAR(7) : N2I, LIMITING AMOUNT ADSORBED OF PURE
 C COMPONENT 2 [KMOL/KG]
 C PAR(8) : N3I, LIMITING AMOUNT ADSORBED OF PURE
 C COMPONENT 3 [KMOL/KG]
 C PAR(9) : A1V, NONIDEALITY PARAMETER OF COMPONENT 1
 C [-]
 C PAR(10) : A2V, NONIDEALITY PARAMETER OF COMPONENT 2
 C [-]
 C PAR(11) : A3V, NONIDEALITY PARAMETER OF COMPONENT 3
 C [-]
 C PAR(12) : R1S, ACTIVITY COEFFICIENT OF COMPONENT 1
 C IN SURFACE PHASE
 C PAR(13) : R2S, ACTIVITY COEFFICIENT OF COMPONENT 2
 C IN SURFACE PHASE
 C PAR(14) : NMI, LIMITING AMOUNT ADSORBED OF MIXTURE
 C [KMOL/KG]
 C PAR(15) : RVS, ACTIVITY COEFFICIENT OF VACANCY
 C IN SURFACE PHASE
 C PAR(16) : XVS, MOLE FRACTION OF VACANCY
 C REPRESENTING ADSORBED PHASE
 C PAR(17) : PRES, EQUILIBRIUM ADSORPTION PRESSURE
 C [KPA]

```

C
C   COMMONBLOCKS      :NONE
C
C   REQUIRED ROUTINES : PARACAL, ZSPOW, TLTRFCN, MAPPLT,
C                       PHDITE.PDL
C
C *****
C *****
C
C   CHARACTER*1 NO
C   DOUBLE PRECISION X(1), WK(10), FNORM,
C   $               PAR(17), NM(21,21),
C   $               X1(21), X2(21), Y1(21,21), Y2(21,21),
C   $               X3(21), Y3(21,21),
C   $               DUM1, DUM2, DUM3, DUM4, MINNM
C   INTEGER         N, NSIG, ITMAX, IER, KK
C   EXTERNAL TLTRFCN
C   DATA NO /'N'/
C   DATA X1, X2 /21*0.D0, 21*0.D0/
C   DATA Y1, Y2, NM /441*0.D0, 441*0.D0, 441*0.D0 /
C
C.....TO CALCULATE PARAMETER VALUES FOR VSM FOR GIVEN
C.....ADSORBATE AND ADSORBENT
C
C   CALL PARACAL (NBENT, NBATE(1), TEMP, PAR(3), PAR(6),
C   PAR(9))
C   CALL PARACAL (NBENT, NBATE(2), TEMP, PAR(4), PAR(7),
C   PAR(10))
C   CALL PARACAL (NBENT, NBATE(3), TEMP, PAR(5), PAR(8),
C   PAR(11))
C
C.....TO CALCULATE MOLE FRACTION OF SOLID PHASE
C.....USING ZSPOW NONLINEAR ALGEBRAIC EQUATION SOLVER
C
C   N = 1
C   NSIG = 9
C   ITMAX = 400
C   PAR(17) = PRES
C   PAR(1) = 0.0
C   MINNM = 1.E30
C
C   DO 20 I=1,21
C       PAR(2) = 0.0

```

```

        X1(I) = PAR(1)
        X(1) = 5.0D-3
        KK = 21 - (I-1)
DO 10 J=1, KK
C
        CALL ZSPOW ( TLTRFCN, NSIG, N, ITMAX, PAR,
$                   X, FNORM, WK, IER )
        WRITE(*,100) I, J, IER, FNORM
C
        NM(I,J) = X(1)
        MINNM = MIN (MINNM, NM(I,J) )
        DUM1 = NM(I,J)
C
C.....TO CALCULATE VAPOR PHASE MOLE FRACTION
C
        CALL YCOMPTE (
$                   PAR(1), PAR(2), DUM1, PAR(9), PAR(10),
$                   PAR(3), PAR(4), PAR(6), PAR(7),
$                   PAR(12), PAR(13), PAR(14), PAR(15),
$                   PAR(16), PRES, DUM2, DUM3, DUM4
$                   )
        Y1(I,J) = DUM2
        Y2(I,J) = DUM3
        Y3(I,J) = DUM4
        X2(J) = PAR(2)
        X3(I) = 1. - X1(I) - X2(J)
        PAR(2) = PAR(2) + 0.05
10    CONTINUE
C
        PAR(1) = PAR(1) + 0.05
20    CONTINUE
C
C.....TO PRINT OUT THE DATA
C
        WRITE (6,101) ( X2(I), I=1,11 )
        WRITE (6,102) ( X1(I), ( Y1(I,J), J=1,11 ), I=1,21 )
        WRITE (6,103) ( X2(I), I=12,21 )
        WRITE (6,104) ( X1(I), ( Y1(I,J), J=12,21 ), I=1,21 )
C
        WRITE (6,105) ( X2(I), I=1,11 )
        WRITE (6,102) ( X1(I), ( Y2(I,J), J=1,11 ), I=1,21 )
        WRITE (6,103) ( X2(I), I=12,21 )

```

```

        WRITE (6,104) ( X1(I), ( Y2(I,J), J=12,21 ), I=1,21 )
C
        WRITE (6,106) ( X2(I), I=1,11 )
        WRITE (6,102) ( X1(I), ( NM(I,J)*1000.,J=1,11),I=1,21)
        WRITE (6,103) ( X2(I), I=12,21 )
        WRITE (6,104) ( X1(I), ( NM(I,J)*1000.,J=12,21),I=1,21)

C
C.....TOPLOT PHASE DIAGRAM WITH CONSYD
C
C      WRITE(*,*) 'DO YOU WANT TO SEE THREE-DIMENSIONAL'
C      WRITE(*,*) '      PHASE DIAGRAMS? [Y OR N]'
C      READ(*, '(A1)') NO
C      IF(NO.EQ.'N') GO TO 80
C
C      DO 91 I = 2, 21
C          DO 90 J = 23-I, 21
C              NM(I,J) = MINNM
C 90      CONTINUE
C 91 CONTINUE
C
C 30 CONTINUE
C      WRITE(*,107)
C      READ(*,*) NOPT
C
C      GO TO ( 40, 50, 60 ) NOPT
C
C 40 CALL TRDPLOT ( NOPT, 21, 21, Y1 )
C      GO TO 70
C
C 50 CALL TRDPLOT ( NOPT, 21, 21, Y2 )
C      GO TO 70
C
C 60      CALL TRDPLOT ( NOPT, 21, 21, NM )
C
C 70 CONTINUE
C      WRITE(*,*) ' DO YOU WANT TO SEE ANOTHER PLOT? [Y OR N]'
C      READ(*, '(A1)') NO
C      IF (NO.EQ.'N') GO TO 80
C      GO TO 30
C
C 80 RETURN
C

```



```

100  FORMAT(
      $      (T5,I3,2X,I3,2X,'IER & FNORM',I3,2X,E11.4/)
      $      )
101  FORMAT(///
      $      T5,'COMPOSITION DATA, X1 - X2 - Y1'//T3,35(1H-)//
      $      T1,' X1\X2',11F6.3//)
102  FORMAT(T1,12F6.3)
103  FORMAT(///
      $      T1,' X1\X2',10F6.3//)
104  FORMAT(T1,11F6.3)
105  FORMAT(///
      $      T5,'COMPOSITION DATA, X1 - X2 - Y2'//T3,35(1H-)//
      $      T1,' X1\X2',11F6.3//)
106  FORMAT(///
      $      T5,'AMT. ADSORBED OF MIXTURE, X1000[KMOL/KG]'/
      $      T3,45(1H-)//
      $      T1,' X1\X2',11F6.3//)
107  FORMAT(///
      $      T5,'1. PHASE DIAGRAM X1 - X2 - Y1'///
      $      T5,'2. PHASE DIAGRAM X1 - X2 - Y2'///
      $      T5,'3. X1 AND X2 VS. AMT. ADSORBED OF MIX.'//
      $      )
C
C      END
C
C
C
C      SUBROUTINE TLTRFCN ( X, F, N, PAR )
C      DOUBLE PRECISION X(*), F(*), PAR(*)
C      INTEGER          N
C
C      *****
C
C
C      SUBROUTINE TLTRFCN    19 OCTOBER 1987  IN-WON KIM
C
C      SUBROUTINE IS TO CALCULATE TOTAL MOLES ADSORBED
C      OF TERNARY MIXTURE
C
C      PARAMETER DESCRIPTION :
C
C      NM      : TOTAL MOLES ADSORBED MIXTURE [KMOL/KG]
C      PAR(1)  : X1, MOLE FRACTION OF COMPONENT 1

```

```

C          IN ADSORBED MIXTURE
C  PAR(2) : X2, MOLE FRACTION OF COMPONENT 2
C          IN ADSORBED MIXTURE
C  PAR(3) : B1, HENRY'S CONSTANT OF COMPONENT 1
C          [KMOL/KG/KPA]
C  PAR(4) : B2, HENRY'S CONSTANT OF COMPONENT 2
C          [KMOL/KG/KPA]
C  PAR(5) : B3, HENRY'S CONSTANT OF COMPONENT 3
C          [KMOL/KG/KPA]
C  PAR(6) : N1I, LIMITING AMOUNT ADSORBED OF PURE
C          COMPONENT 1 [KMOL/KG]
C  PAR(7) : N2I, LIMITING AMOUNT ADSORBED OF PURE
C          COMPONENT 2 [KMOL/KG]
C  PAR(8) : N3I, LIMITING AMOUNT ADSORBED OF PURE
C          COMPONENT 3 [KMOL/KG]
C  PAR(9) : A1V, NONIDEALITY PARAMETER OF COMPONENT 1
C          [ - ]
C  PAR(10): A2V, NONIDEALITY PARAMETER OF COMPONENT 2
C          [ - ]
C  PAR(11): A3V, NONIDEALITY PARAMETER OF COMPONENT 3
C          [ - ]
C  PAR(12): R1S, ACTIVITY COEFFICIENT OF COMPONENT 1
C          IN SURFACE PHASE
C  PAR(13): R2S, ACTIVITY COEFFICIENT OF COMPONENT 2
C          IN SURFACE PHASE
C  PAR(14): NMI, LIMITING AMOUNT ADSORBED OF MIXTURE
C          [KMOL/KG]
C  PAR(15): RVS, ACTIVITY COEFFICIENT OF VACANCY
C          IN SURFACE PHASE
C  PAR(16): XVS, MOLE FRACTION OF VACANCY
C          REPRESENTING ADSORBED PHASE
C  PAR(17): PRES, EQUILIBRIUM ADSORPTION PRESSURE
C          [KPA]
C
C  COMMONBLOCKS      :NONE
C
C  REQUIREROUTINES:NONE
C
C *****
C *****
C
C  DOUBLE PRECISION X1, X2, X3, NM, NMI, X1S, X2S, X3S,
$      XVS, A12, A21, A1V, A2V, A13, A31, A23, A32,

```

```

$      A3V, DUM1, DUM2, DUM3, DUM4,
$      ALNR1S, ALNR2S, ALNR3S, ALNRVS,
$      R1S, R2S, R3S, RVS, Y1P, Y2P, Y3P
DOUBLE PRECISION B1, B2, B3, N1I, N2I, N3I,
$      AV1, AV2, AV3, PRES

```

C

```

X1 = PAR(1)
X2 = PAR(2)
B1 = PAR(3)
B2 = PAR(4)
B3 = PAR(5)
N1I = PAR(6)
N2I = PAR(7)
N3I = PAR(8)
A1V = PAR(9)
A2V = PAR(10)
A3V = PAR(11)
PRES = PAR(17)

```

C

```

X3 = 1. - X1 - X2
NM = X(1)

```

C

C.....TO MAKE SURE NM IS LESS THAN NMI

C

```

NMI = X1*N1I + X2*N2I + X3*N3I
IF(NM.GT.NMI) THEN
  NM=NMI*.99999
END IF
X1S = NM*X1 / NMI
X2S = NM*X2 / NMI
X3S = NM*X3 / NMI
XVS = 1. - NM / NMI

```

C

C.....TO CALCULATE NONIDEALITY PARAMETER OF MIXTURE

C

```

A12 = (A1V + 1.)/(A2V+1) - 1.
A21 = 1./(A12+1) -1.
A13 = (A1V + 1.)/(A3V+1) - 1.
A31 = 1./(A13+1) -1.
A23 = (A2V + 1.)/(A3V+1) - 1.
A32 = 1./(A23+1) -1.
AV1 = 1./(A1V+1) -1.
AV2 = 1./(A2V+1) -1.

```

```

      AV3 = 1./(A3V+1) -1.
C
C.....TO CALCULATE ACTIVITY COEFFICIENTS
C
      DUM1 = X1S + X2S/(A12+1.) + X3S/(A13+1.) +XVS/(A1V+1.)
      ALNR1S = -LOG(DUM1) + 1. - 1./DUM1
      R1S = EXP (ALNR1S)
C
      DUM2 = X1S/(A21+1.) + X2S + X3S/(A23+1.) +XVS/(A2V+1.)
      ALNR2S = -LOG(DUM2) + 1. - 1./DUM2
      R2S = EXP (ALNR2S)
C
      DUM3 = X1S/(A31+1.) + X2S/(A32+1.) + X3S +XVS/(A3V+1.)
      ALNR3S = -LOG(DUM3) + 1. - 1./DUM3
      R3S = EXP (ALNR3S)
C
      DUM4 = X1S/(AV1+1.) + X2S/(AV2+1.) + X3S/(AV3+1.) + XVS
      ALNRVS = -LOG(DUM4) + 1. - 1./DUM4
      RVS = EXP (ALNRVS)
C
C
      Y1P = R1S*X1*NM/NMI*N1I/B1*EXP(A1V)/(1.+A1V)
$      *EXP(((N1I-NMI)/NM-1.)*LOG(RVS*XVS))
      Y2P = R2S*X2*NM/NMI*N2I/B2*EXP(A2V)/(1.+A2V)
$      *EXP(((N2I-NMI)/NM-1.)*LOG(RVS*XVS))
      Y3P = R3S*X3*NM/NMI*N3I/B3*EXP(A3V)/(1.+A3V)
$      *EXP(((N3I-NMI)/NM-1.)*LOG(RVS*XVS))
      F(1) = PRES - Y1P -Y2P -Y3P
C
C.....PARAMETER CONVERSION
C
      PAR(12) = R1S
      PAR(13) = R2S
      PAR(14) = NMI
      PAR(15) = RVS
      PAR(16) = XVS
C
      RETURN
      END
C
C
C
SUBROUTINE YCOMPTE (X1, X2, NM, A1V, A2V, B1, B2,

```

```

$          N1I, N2I, R1S, R2S, NMI, RVS,
$          XVS, PRES, Y1, Y2, Y3)
DOUBLE PRECISION X1, X2, Y1, Y2, Y3, R1S, R2S,
$          NM, NMI, N1I, N2I, B2, A2V,
$          B1, A1V, RVS, XVS, PRES
C
C *****
C *****
C
C SUBROUTINE YCOMPTE 26 OCTOBER 1987 IN-WON KIM
C
C THIS SUBROUTINE IS TO CALCULATE MOLE FRACTION OF GAS
PHASE
C IN BINARY MIXTURE
C
C PARAMETER DESCRIPTION:
C
C NM : TOTAL MOLES ADSORBED MIXTURE [KMOL/KG]
C X1 : MOLE FRACTION OF COMPONENT 1
C      IN ADSORBED MIXTURE
C X2 : MOLE FRACTION OF COMPONENT 2
C      IN ADSORBED MIXTURE
C B1 : HENRY'S CONSTANT OF COMPONENT 1
C      [KMOL/KG/KPA]
C B2 : HENRY'S CONSTANT OF COMPONENT 2
C      [KMOL/KG/KPA]
C N1I : LIMITING AMOUNT ADSORBED OF PURE
C      COMPONENT 1 [KMOL/KG]
C N12 : LIMITING AMOUNT ADSORBED OF PURE
C      COMPONENT 2 [KMOL/KG]
C A1V : NONIDEALITY PARAMETER OF COMPONENT 1
C      [ - ]
C A2V : NONIDEALITY PARAMETER OF COMPONENT 2
C      [ - ]
C R1S : ACTIVITY COEFFICIENT OF COMPONENT 1
C      IN SURFACE PHASE
C R2S : ACTIVITY COEFFICIENT OF COMPONENT 2
C      IN SURFACE PHASE
C NMI : LIMITING AMOUNT ADSORBED OF MIXTURE
C      [KMOL/KG]
C RVS : ACTIVITY COEFFICIENT OF VACANCY
C      IN SURFACE PHASE
C XVS : MOLE FRACTION OF VACANCY

```

```

C          REPRESENTING ADSORBED PHASE
C  PRES : EQUILIBRIUM ADSORPTION PRESSURE
C          [ KPA ]
C  Y1  : GAS PHASE MOLE FRACTION OF COMPONENT 1
C          [ - ]
C  Y2  : GAS PHASE MOLE FRACTION OF COMPONENT 2
C          [ - ]
C  Y3  : GAS PHASE MOLE FRACTION OF COMPONENT 3
C          [ - ]
C
C  COMMONBLOCKS      :NONE
C
C  REQUIREDROUTINES:NONE
C
C *****
C *****
C
C
C  Y1 = R1S*X1*NM/NMI*N1I/B1*EXP(A1V)/(1.+A1V)
$      *EXP(((N1I-NMI)/NM-1.)*LOG(RVS*XVS))/PRES
C  Y2 = R2S*X2*NM/NMI*N2I/B2*EXP(A2V)/(1.+A2V)
$      *EXP(((N2I-NMI)/NM-1.)*LOG(RVS*XVS))/PRES
C  Y3 = 1.-Y1 -Y2
C
C  RETURN
C  END
C
C
C
C  SUBROUTINE TRDPLOT ( NOPT, NX, NY, RAWDATA )
C *****
C *****
C
C  SUBROUTINE TRDPLOT 26 OCTOBER 1987 IN-WON KIM
C
C  THIS SUBROUTINE IS TO PLOT THREE DIMENSIONAL PLOT
C
C  PARAMETERDESCRIPTION:
C
C  NOPT : OPTION NO.
C  NX   : NO. OF DATA POINTS ON X-AXIS

```

```

C      NY      : NO. OF DATA POINTS ON Y-AXIS
C      RAWDATA : DATA SET TO BE PLOTTED
C      XMIN    : MINIMUM VALUE OF X-AXIS
C      XMAX    : MAXIMUM VALUE OF X-AXIS
C      YMIN    : MINIMUM VALUE OF Y-AXIS
C      YMAX    : MAXIMUM VALUE OF Y-AXIS

```

```

C      COMMONBLOCKS :NONE

```

```

C      REQUIREROUTINES:NONE

```

```

C      *****
C      *****

```

```

C      DOUBLE PRECISION RAWDATA ( 21, 21 )
C      REAL DUMRAY(21), XMIN, XMAX, YMIN, YMAX
C      INTEGER*2 NX, NY

```

```

C      GO TO ( 10, 20, 30 ) NOPT
10  CALL ASSIGN(1,'THRDAT1.DAT')
    GO TO 40
20  CALL ASSIGN(1,'THRDAT2.DAT')
    GO TO 40
30  CALL ASSIGN(1,'THRDAT3.DAT')
40  CONTINUE
    REWIND 1

```

```

C      XMIN = 0.0
C      XMAX = 1.0
C      YMIN = 0.0
C      YMAX = 1.0
C      WRITE(1) NX, NY, XMIN, XMAX, YMIN, YMAX

```

```

C      DO 60 I = 1, NX
C          DO 50 J = 1, NY
C              DUMRAY(J) = SNGL ( RAWDATA(I,J) )
50  CONTINUE
    WRITE(1) (DUMRAY(K), K=1,NY)
60  CONTINUE
    CLOSE(1)

```

```

C      WRITE(*,100)

```

```

C
      RETURN
C
100      FORMAT(//
$        T5,'TO SEE THE THREE-DIMENSIONAL PLOT, '//
$        T5,'PLEASE TYPE LIKE THAT '//
$        T5,'VMS> PL3 '//
$        T5,'PL3> THRDAT' '//
$        T5,'THRDAT1.DAT : X1 - X2 - Y1 PLOT' '//
$        T5,'THRDAT2.DAT : X1 - X2 - Y2 PLOT' '//
$        T5,'THRDAT3.DAT : X1 - X2 - NM PLOT' '//
$      )
C
      END
C
C
C
      BLOCK DATA ADATA
C
C *****
C *****
C
C      BLOCK DATA ANAME 19 OCTOBER 1987  IN-WON KIM
C
C      THIS SUBROUTINE SULLPLIES THE PARAMETER VALUES OF
C      F-H VACANCY SOLUTION MODEL.
C
C      PARAMETER DESCRIPTION :
C
C      PARA(I,J,1) = TEMPERATURE INDEPENDENT CONSTANT
C                     FOR HERY'S LAW CONSTANT,
C                     Bio      [kmol/kg/kpa]
C      PARA(I,J,2) = TEMPERATURE INDEPENDENT CONSTANT
C                     FOR LIMITING AMOUNT ADSORBED,
C                     n1i      [kmol/kg]
C      PARA(I,J,3) = TEMPERATURE INDEPENDENT CONSTANT,
C                     CHARACTERISING EACH ADSORBATE-
C                     ADSORBENT SYSTEM
C                     ri,      [ K ]
C      PARA(I,J,4) = ISOSTERIC HEAT OF ADSORPTION,
C                     q,      [J/kmol]
C      PARA(I,J,5) = TEMPERATURE INDEPENDENT CONSTANT
C                     FOR PROPORTIONALITY,

```



```

C          m,          [kg/kmol]
C      I      = ADSORBENT (SEE BLOCK DATA ANAME)
C
C      J      = ADSORBATE (SEE BLOCK DATA BNAME)
C
C      DOUBLE PRECISION DATA
C
C      COMMONBLOCKS      : PARAMET
C
C      REQUIRIED ROUTINES : NONE
C
C *****
C *****
C
C      DOUBLE PRECISION PARA(20,20,5)
C      COMMON /PARAMET/ PARA
C
C      DATA (PARA(1,1,I),I=1,5)/
C      $      11.4 D-09, 1.80 D-03, 0.   D 00, -1.057D07, 17.68D02 /
C
C      DATA (PARA(1,2,I),I=1,5)/
C      $      1.03 D-09, 5.99 D-03, 0.   D 00, -1.970D07, 5.32 D02 /
C
C      DATA (PARA(3,3,I),I=1,5)/
C      $      1.011D-09, 2.35 D-03, 2.55 D 02, -3.344D07, 7.19 D02 /
C
C      DATA (PARA(3,4,I),I=1,5)/
C      $      5.385D-09, 1.355D-03, 3.927D 02, -4.081D07, 8.986D02 /
C
C      DATA (PARA(4,5,I),I=1,5)/
C      $      5.94 D-09, 10.5 D-03, 0.   D 00, -1.886D07, 3.52 D02 /
C
C      DATA (PARA(4,6,I),I=1,5)/
C      $      2.49 D-09, 12.4 D-03, 0.   D 00, -2.426D07, 2.76 D02 /
C
C      DATA (PARA(4,7,I),I=1,5)/
C      $      2.35 D-09, 11.0 D-03, 0.   D 00, -2.762D07, 3.75 D02 /
C
C      DATA (PARA(4,8,I),I=1,5)/
C      $      4.78 D-09, 5.91 D-03, 0.   D 00, -2.646D07, 6.04 D02 /
C
C      DATA (PARA(4,3,I),I=1,5)/

```

C \$ 5.60 D-09, 5.71 D-03, 0. D 00, -2.707D07, 6.30 D02 /
 DATA (PARA(4,9,I),I=1,5)/
 C \$ 2.85 D-09, 2.14 D-03, 270. D 00, -3.759D07, 10.79D02 /
 DATA (PARA(4,4,I),I=1,5)/
 C \$ 7.87 D-09, 2.56 D-03, 208. D 00, -3.577D07, 11.91D02 /
 DATA (PARA(4,10,I),I=1,5)/
 C \$ 33.30D-09, 1.93 D-03, 217. D 00, -3.993D07, 18.86D02 /
 DATA (PARA(5,8,I),I=1,5)/
 C \$ 0.561D-09, 4.58 D-03, 0. D 00, -2.644D07, 7.03 D02 /
 DATA (PARA(5,9,I),I=1,5)/
 C \$ 2.16 D-09, 4.61 D-03, 0. D 00, -2.806D07, 7.68 D02 /
 DATA (PARA(5,4,I),I=1,5)/
 C \$ 0.917D-09, 8.98 D-03, 0. D 00, -2.681D07, 4.35 D02 /
 DATA (PARA(6,1,I),I=1,5)/
 C \$ 5.89 D-09, 7.01 D-03, 0. D 00, -1.265D07, 2.54 D02 /
 DATA (PARA(6,2,I),I=1,5)/
 C \$ 11.30D-09, 5.32 D-03, 0. D 00, -1.575D07, 8.05 D02 /
 DATA (PARA(6,11,I),I=1,5)/
 C \$ 136.0D-09, 6.03 D-03, 0. D 00, -1.566D07, 9.87 D02 /
 DATA (PARA(7,8,I),I=1,5)/
 C \$ 1.25 D-09, 2.91 D-03, 0. D 00, -3.532D07, 10.93D02 /
 DATA (PARA(7,3,I),I=1,5)/
 C \$ 1.36 D-09, 2.88 D-03, 0. D 00, -2.775D07, 4.38 D02 /
 DATA (PARA(7,12,I),I=1,5)/
 C \$ 74.30D-09, 0.408D-03, 457. D 00, -2.971D07, 21.71D02 /
 DATA (PARA(7,6,I),I=1,5)/
 C \$ 0.913D-09, 4.35 D-03, 0. D 00, -3.753D07, 9.29 D02 /
 C END

```

C
C
C      BLOCK DATA ANAME
C
C      *****
C      *****
C
C      BLOCK DATA ANAME  19 OCTOBER 1987  IN-WON KIM
C
C      THIS SUBROUTINE IS FOR THE NAME OF ADSORBENT
C
C      COMMONBLOCKS      :BETNAME
C
C      REQUIREROUTINES:NONE
C
C      *****
C      *****
C
C      CHARACTER*25 SORBENT(20)
C      COMMON/BETNAME/SORBENT
C
C      DATA SORBENT(1) / '(1) RS10'/
C      DATA SORBENT(2) / '(2) ZEOLITE 5A'/
C      DATA SORBENT(3) / '(3) ACTIVATED CARBON'/
C      DATA SORBENT(4) / '(4) NUXIT AL ACT. CAR.'/
C      DATA SORBENT(5) / '(5) SILICA'/
C      DATA SORBENT(6) / '(6) ZEOLITE 10X'/
C      DATA SORBENT(7) / '(7) ZEOLITE 13X'/
C
C      END
C
C
C
C      BLOCK DATA BNAME
C
C      *****
C      *****
C
C      BLCK DATA BNAME  19 OCTOBER 1987  IN-WON KIM
C
C      THIS SUBROUTINE IS FOR THE NAME OF ADSORBATE
C
C      COMMONBLOCKS      :BATNAME

```

```

C
C   REQUIREDROUTINES:NONE
C
C *****
C *****
C
C   CHARACTER*25 SORBATE(20)
C   COMMON/BATNAME/SORBATE
C
C   DATA SORBATE(1) / '(1) OXYGEN'/
C   DATA SORBATE(2) / '(2) NITROGEN'/
C   DATA SORBATE(3) / '(3) ETHANE'/
C   DATA SORBATE(4) / '(4) PROPANE'/
C   DATA SORBATE(5) / '(5) METHANE'/
C   DATA SORBATE(6) / '(6) CARBON DIOXIDE'/
C   DATA SORBATE(7) / '(7) ACETHYLENE'/
C   DATA SORBATE(8) / '(8) ETHENE'/
C   DATA SORBATE(9) / '(9) PROPENE'/
C   DATA SORBATE(10) / '(10) N-BUTANE'/
C   DATA SORBATE(11) / '(11) CARBON MONOXIDE'/
C   DATA SORBATE(12) / '(12) I-BUTANE'/
C
C   END

```

APPENDIX D: On Board Oxygen Generation System (OBOGS) Simulator Manual

On Board Oxygen Generation System (OBOGS)

Simulator Manual

By

Glenn Munkvold

April, 1988

Summary

This manual describes the operation of a computer code that solves the adsorption/flow equations in a two bed PSA (pressure swing adsorption) system as described in Beaman, Healey, and Werlin (1983), and Beaman (1985). The model for the beds assumes isothermal and isobaric columns, negligible radial gradients, negligible axial dispersion, and that convective terms dominate temporal terms in the gas phase equations. Modified Langmuir isotherms are used, and can be coupled using the Ideal Adsorbed Solution Theory (IAST) of Myers and Prausnitz (1965). Because the model was developed for oxygen enrichment of air, a common mass transfer coefficient was used for oxygen and nitrogen. Argon and trace gases in air have been neglected.

As reported by Beaman (1985), the model has been successful in matching the behavior of a small two bed OBOGS.

Table of Contents

Summary	2
Table of Contents	3
Introduction	4
Program Structure	5
Data Editing.....	5
Main Processor.....	9
Utility/Post Processing.....	9
PARAMETERs in the Simulator.....	10
COMMON Blocks	10
Usage Notes.....	11
Examples.....	12
Example Run	12
Run Time Messages	19
Input Files	19
Output Files.....	21
Subroutine Descriptions.....	26
Index	43
References	45
Variables Used in Equations.....	46
Source Code Listing.....	48
BINKLEY.FOR	48
COUPISO.FOR.....	116
ISOETHM.FOR	117
FALSEP.FOR.....	118
Example Post Processor--PLOTPSA.FOR.....	119

Introduction

Beaman, et al. (1983), developed isotherms and bed equations for oxygen enrichment of air using zeolite 5A. Beaman (1985) solved these equations for a two bed OBOGS and compared the simulations to experiments, with good results. Unfortunately, extensions and modifications to the original code resulted in an unreliable and inconvenient program. The original program employed hard wired parameters with limited abilities for users to change parameters interactively. Recompile of the code was often necessary.

Therefore, a new program was written to implement Beaman's model. This code uses data files to describe both bed characteristics and run time parameters. Parameters can be changed interactively in the program, or they can be changed in a standard file editor. The code allows for multiple bed zones and step changes in boundary conditions. Bed velocity and gas composition profiles can be output at any time. Also, runs can be restarted after normal termination (or can be restarted from an intermediate velocity and composition profile) in order to extend the run if the initial simulation time was not long enough.

The program was written in Vax FORTRAN and runs under VMS. Graphics output was not included in the code (intentionally) due to the numerous differences that exist in computing packages, graphics packages, and terminals, even at a single site. However, the program can be SPAWNed in a DCL file on the Vax, so that user written output conversion codes can be run to generate graphics output immediately following execution of the OBOGS model. These operations can be performed in a DCL command (.COM) file.

This manual contains details of program structure, the main sections of code, the COMMON blocks used, and an explanation of the PARAMETER statements used to dimension the problem. Usage notes, an example run, and detailed descriptions of each subroutine are also included.

Program Structure

Figure 1 is a simplified flow chart of program BINKLEY, the OBOGS simulator. Lines on the chart indicate calls; returns are always back to the calling routine and no subroutine in the chart calls any routine on the same level or above. The dotted line from ISOTHM represents calls made only in the case of coupled isotherms. Reading left to right in a level generally corresponds to the order the subroutines are called. Details of the routines are presented later.

From the flowchart, three main sections can be found: the data editing section, the main processor, and the utility routine section. Subroutines YES, EXTADD, FILCHK, DATE, TIME, and XGETPAM are each used by several higher level routines and are therefore designated utility routines. Only XGETPAM is specific to BINKLEY; the others perform generic functions. The utility routines appear in the bottom two tiers of the flowchart.

The data editing section is all those routines from STARTUP down through KTCORR. These routines control all the data manipulation and set up the run. The user can quit the simulator after editing the data without running the actual simulation.

The main processor section consists of all the routines from PROC down through VALVE. These routines actually solve the flow and adsorption equations, as well as print results. PROC loops until the simulation ends, and the final bed profiles are output by BINKLEY, through SNAPOUT, to the warm start file (START.WRM).

Data Editing

Data are modified through STARTUP and subsequent routines. For a normal (not a warm start) run, STARTUP calls BEDDAT to prompt the user for a .MSV file that describes bed parameters. This input file is opened as UNIT 8. After getting a file name, the routine calls PREAD to read all the bed parameters and then calls the routines that modify and display the information in the .MSV file. CCNG is the routine that changes the comment section of the data set. Five comment lines can be used to describe the data set. VPCNG changes the valve parameters (orifice diameters and discharge coefficients). Valve numbers are shown in Figure 2, a schematic of the OBOGS being simulated. Valve #1 is the supply gas valve; valve #2 is the exhaust valve; and valve #3 is the bypass orifice. ZCNG changes the zone specific parameters of the beds. Each bed has been allotted a total of NZ zones. NZ is a PARAMETER set equal to five in the original code. Each zone can have a different diameter, void fraction, diffusion coefficient, and/or isotherm coefficients. Zone temperatures are shown on zone menus, but they are only used in temperature correlations for the isotherm coefficients (KTCORR) because

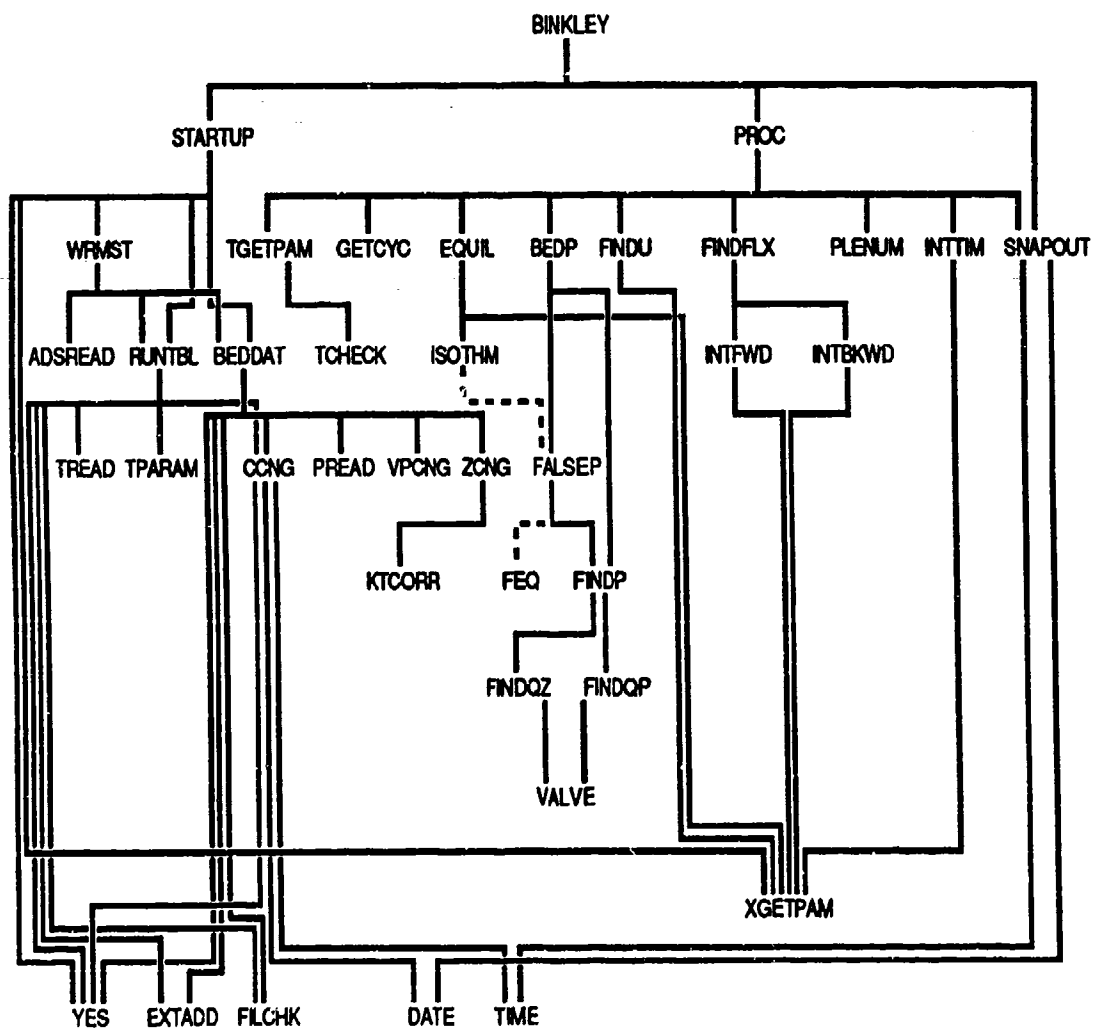


Figure 1. Simplified Simulator Flowchart

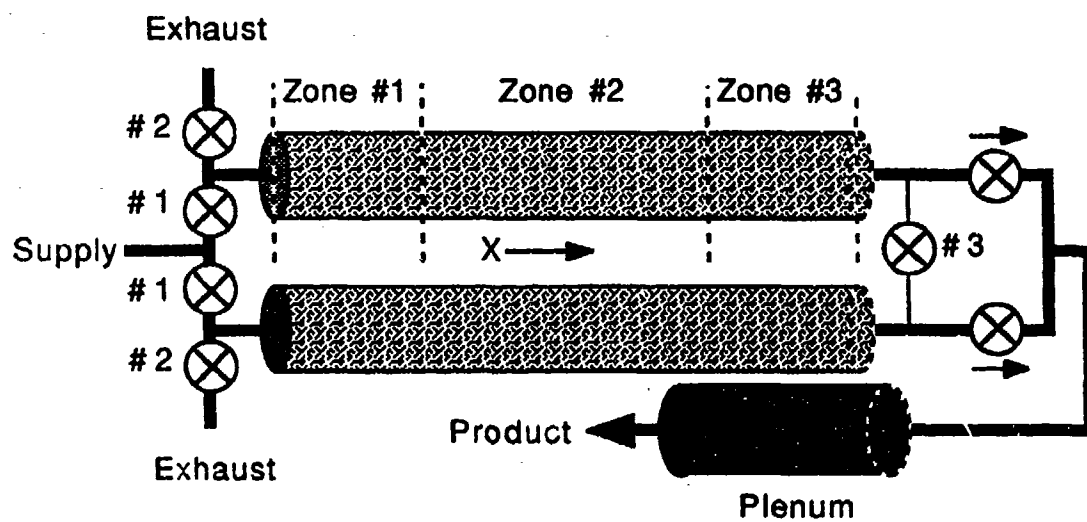


Figure 2. Bed Schematic Showing Valve Numbers, Zones, and Sign Convention for Axial Distance

of the isothermal bed assumption. The temperature of the first zone is used as the temperature in all flow calculations in all zones.

After changing (or viewing) data, the user must save the data. Saving under the same name causes the old version to be DELETED; saving under a new name creates a new data set while retaining the old. UNIT 8 is closed.

Upon return from BEDDAT, STARTUP calls RUNTBL, which prompts for a filename again. This file is a .TBL file, which controls run time parameters such as space step size, time step size, "snap shot" times (when bed profile data are output), and step changes in inlet pressure, exhaust pressure, cycle time, product flowrate, and inlet oxygen concentration. TPARAM changes the parameter values. CCNG modifies file comments, as above. Files must be saved on exit as in BEDDAT, above. The .TBL file is opened as UNIT 8 and closed on exit.

After calling BEDDAT and RUNTBL, the user can quit without running the simulation. If the user continues, STARTUP performs some unit conversions, adjusts the space step size to yield an equally spaced mesh along the bed length, and initializes the adsorbed gas arrays before return to BINKLEY.

If the run is a warm start, STARTUP calls WRMST to read parameters from the highest numbered version of START.WRM. ADSREAD actually reads the file. Again, the input file is UNIT 8. The user is told the last simulation time from the previous run and prompted for a new final value. WRMST uses the filenames read by ADSREAD to obtain bed and run time parameters through RUNTBL and BEDDAT (where interactive requests are suppressed). The warm start capability has been included mainly to allow runs to obtain steady state; therefore, the user cannot modify data files in a warm start. However, parameters in the data files could be modified with a standard file editor before the warm start. The warm start feature can also be used to start runs from intermediate bed profile output (snap shots). This will be detailed below.

Both the .MSV and .TBL files are formatted FORTRAN output files. While changing the values in a free format data file is easier, knowing which values to change is more difficult. A directory full of data files without explanatory information is a directory full of garbage. Thus, the formatted file style is used. This is an important point to consider when modifying files with a standard file editor.

Finally, it is important to note that 'Y' or 'y' (or any string beginning with these letters) are considered affirmative responses to yes/no questions. All other responses are assumed to be negative responses. Also, all interactive numerical response requests are free format reads and the entire program will crash if non-numerical data are input. UNIT 1 is opened in STARTUP as PSA.DAT, the main output file (product oxygen

concentration, total inlet mass rate, and product oxygen mass rate as functions of time). UNIT 2 is opened as START.WRM, which will contain the final bed conditions (for each bed) as a function of position.

Main Processor

The main processor performs the bed equation calculations. PROC coordinates the activity of this section of code. The time varying input parameters are obtained through TGETPAM. GETCYC determines the inlet pressure for each bed, depending on the cycle time and the elapsed simulation time. EQUIL calculates the interfacial concentrations (CINTO2 and CINTOT) from the adsorbed phase concentrations. It also calculates CINTINT and BCINT, which are integrals used in BEDP and FINDU. BEDP solves the equations that couple the pressures in the beds in order to find those pressures. FINDU determines the velocity profile in the bed by integrating equation (5) in Beaman, et al. (1983); FINDFLX calculates the total flux and the oxygen flux through the bed from equations (11) in that paper. PLENUM models the OBOGS plenum as a two stage mixer with time constants determined from half the total plenum volume. INTTIM integrates the adsorbed phase concentrations in time using a forward difference approximation for the time derivatives. Finally, SNAPOUT outputs the bed velocity, adsorbed phase concentration, and gas phase oxygen mole fraction profiles at the times set in RUNTBL for snap shot output.

The details of the PROC section of code are not as important as the details of the data editing section in operating the model. Therefore, these details are deferred to the subroutine descriptions below.

Utility/Post Processing

Six of the routines in the codes are considered utility routines: YES, EXTADD, FILCHK, DATE, TIME, and XGETPAM. Of these, only four are included in the source listing because DATE and TIME are Vax FORTRAN subroutines for the system date and time. YES classifies a response as affirmative ('y' or 'Y') or negative (all other characters) and returns a logical variable that is .TRUE. for an affirmative response. EXTADD adds an extension to a file name. FILCHK checks to see if a file already exists in a directory.

XGETPAM is the only utility routine specific to the simulator. This routine is called to find bed parameters as a function of total bed distance or zone number.

SNAPOUT is the only post processing routine. It is called by BINKLEY to output bed profiles after normal execution is ended in PROC. An example post processor (run as a separate program after the simulator) is included after the listing of the simulator.

PARAMETERS in the Simulator

The following parameters control the dimensions of the problem:

- MAXLUMP-----The maximum number of lumps allowed in a bed plus 1
MAXSTPS-----The maximum number of step changes in the time varying
input variables
NZ-----The maximum number of zones in a bed
NZP1 -----The maximum number of zones in a bed plus 1

COMMON Blocks

The following COMMON blocks occur in the code:

/BDAT/ BEDD(NZ), ZONEL(NZ), DIFF(NZ), VOIDF(NZ), TZONE(NZ),
ZONEKA(NZ), ZONEKB(NZ), ZONEB(NZ), NZONE,FAREA(NZ),
BBETA(NZ)
/BDAT2/ VALVB, VALVO, VALVS, VOLPLEN
/EQFCN/ ALPHA, FRACN2
/FPRESS/ PBEDI, CONV, TEMPP, RMW(2,2), FACIN, NNBD, QZ(2),
QL(2), WBRMOL, PINJ, CCINT(2), CBCINT(2)
/GASES/ O2MW, RN2MW, RGAS1
/INIT/ PATM, TEMPI, O2BLKD
/PLEN/ YO2,YO2M1
/RDAT/ RBFLOW(MAXSTPS,2), RO2IN(MAXSTPS,2),
RPSUP(MAXSTPS,2), RPOUT(MAXSTPS,2), RTCYC(MAXSTPS,2),
SNAPTIM(MAXSTPS,2)
/RDAT2/ DELTAX, DELTAT, SIMTIM, LUMPS, LUMPS1, NTIM
/VLVCON/ CONST1, CONST2, CVCP, CRITRAT

Usage Notes

This section mentions some of the miscellany to consider when running the model.

Initial conditions for the model are air saturated beds (equilibrium) at atmospheric (PATM) pressure. Feed is binary (oxygen and nitrogen). Supply pressure starts on bed number one. The beds are identical.

OBOGS2.EXE is the model with uncoupled isotherms. OBOGSC.EXE is the model with coupled isotherms. The source code itself is in four files: BINKLEY.FOR contains most of the subroutines; COUPISO.FOR contains the Beaman defined isotherm in ISOTHM as well as FEQ1, the function defined in equation (22) of Beaman, et al. (1983); ISOTHM.FOR contains the uncoupled Beaman isotherm; FALSEP.FOR contains the false position solver. The files are supplied this way because the Vax linker is a single pass linker. If the coupled isotherm case is chosen, FALSEP must appear after ISOTHM in the link list. LINK2.COM will link the uncoupled isotherm routines; LINKC.COM will link the coupled isotherm routines.

Other files included with the simulator are example data sets, output, and a post processor. The data sets are EX1.MSV, EX1.TBL, and EX2.TBL. These sets should be used to create new data sets.

The output files are PSA.DAT, START.WRM, and a snap shot file, SNAP04.DAT. Snap shot files are SNAPXX.DAT, where XX is the number selected for the snap shot unit in RUNTBL.

The post processor PLOTPSA.FOR has been included as a guide to reading the output files. It is a plotting routine and uses library software at the University of Texas at Austin.

Also note that the screen is UNIT 6; therefore, no SNAPOUT output should ever be sent to this unit.

Examples

Example Run

The following is an example input/output session that includes some data editing in starting a simulation:

\$ RUN OBOGS2

Welcome to the OBOGS simulator
This version completed in April, 1988

Is this a warm start?

N

This is the program that reads and modifies data files for
the OBOGS simulator (APR 1988)

Please enter the name of the file to be read

Modified files can be saved under a different name later

The filename should be seven or less characters and the
extension

should be MSV

(do NOT include the extension here)

BEA1985

Do you want to modify BEA1985.MSV?

Y

Modify comment lines

OBOGS Simulation 29-MAR-88 08:12:32 Data From BEA1985.MSV

This is a data file for the F-16 OBOGS by Bendix.

It is derived from the hard-wired defaults in the
simulator by Brian Yang. This set was input by
Glenn Munkvold, 4/19/87. This run is an attempt
to duplicate Beaman's 1985 results.

Do you want to modify the data comments?

Y

Comment changes: After each line of comments you have four choices:

1. Simply type a new comment to supercede old comment
2. Type \ to erase a comment line
3. Type + immediately followed by text to append to old comment
4. Type @ to leave comment unchanged

This is a data file for the F-16 OBOGS by Bendix.

@

It is derived from the hard-wired defaults in the

```

@
simulator by Brian Yang. This set was input by
@
Glenn Munkvold, 4/19/87. This run is an attempt
@
to duplicate Beaman's 1985 results.
+Two bed zoned
This is a data file for the F-16 OBOGS by Bendix.
It is derived from the hard-wired defaults in the
simulator by Brian Yang. This set was input by
Glenn Munkvold, 4/19/87. This run is an attempt
to duplicate Beaman's 1985 results. Two bed zones
Do you want to modify the data comments further?
Y
Comment changes: After each line of comments you have four choices:
1. Simply type a new comment to supercede old comment
2. Type \ to erase a comment line
3. Type + immediately followed by text to append to old comment
4. Type @ to leave comment unchanged
This is a data file for the F-16 OBOGS by Bendix.
@
It is derived from the hard-wired defaults in the
@
simulator by Brian Yang. This set was input by
@
Glenn Munkvold, 4/19/87. This run is an attempt
@
to duplicate Beaman's 1985 results. Two bed zones
+are used.
This is a data file for the F-16 OBOGS by Bendix.
It is derived from the hard-wired defaults in the
simulator by Brian Yang. This set was input by
Glenn Munkvold, 4/19/87. This run is an attempt
to duplicate Beaman's 1985 results. Two bed zones are used.
Do you want to modify the data comments further?
n

```

Modify valve paramters

Valve Parameter Changes

```

Enter number to change parameter
Enter zero to exit

```

Valve	Diameter(cm)	Discharge Coeff
Bypass	1. 0.1905	2. 0.6000
Supply	3. 0.7777	4. 0.7000
Outlet	5. 1.1143	6. 0.7000

7.Plenum Volume= 0.180E-02 (cubic meters)
8.Number of Bed Zones= 2 (maximum is 5)
Enter number (zero to exit):
0

Modify zone specific parameters
Enter zone # (zero to exit)

1

Zone Parameter Changes
Enter number to change parameter
Enter zero to exit

1.Zone # 1
2.Bed Diameter= 13.2000 (cm)
3.Zone Length= 39.4000 (cm)
4.Diffusion Coefficient= 200.0000 (1/sec)
5.Void Fraction=0.3700
6.Temperature= 25.00 (degrees C)
7.KA= 0.1928 (kgmol gas/kgmol ads, O2)
8.KB= 0.0687 (kgmol gas/kgmol ads, N2)
9.B= 3.059 (kgmol N2 ads/cubic meter)
KA, KB <0 are calculated from temperature
Enter number (zero to exit):

7
7.KA= 0.1928 (kgmol gas/kgmol ads, O2)
Enter new value

-1

1.Zone # 1
2.Bed Diameter= 13.2000 (cm)
3.Zone Length= 39.4000 (cm)
4.Diffusion Coefficient= 200.0000 (1/sec)
5.Void Fraction=0.3700
6.Temperature= 25.00 (degrees C)
7.KA= -1.0000 (kgmol gas/kgmol ads, O2)
8.KB= 0.0687 (kgmol gas/kgmol ads, N2)
9.B= 3.059 (kgmol N2 ads/cubic meter)
KA, KB <0 are calculated from temperature
Enter number (zero to exit):

8
8.KB= 0.0687 (kgmol gas/kgmol ads, N2)
Enter new value

-1

1.Zone # 1
2.Bed Diameter= 13.2000 (cm)
3.Zone Length= 39.4000 (cm)
4.Diffusion Coefficient= 200.0000 (1/sec)
5.Void Fraction=0.3700
6.Temperature= 25.00 (degrees C)
7.KA= -1.0000 (kgmol gas/kgmol ads, O2)
8.KB= -1.0000 (kgmol gas/kgmol ads, N2)
9.B= 3.059 (kgmol N2 ads/cubic meter)
KA, KB <0 are calculated from temperature
Enter number (zero to exit):

1

1.Zone # 1

Enter new value

2

Zone Parameter Changes

Enter number to change parameter

Enter zero to exit

1.Zone # 2

2.Bed Diameter= 6.8000 (cm)

3.Zone Length= 39.4000 (cm)

4.Diffusion Coefficient= 200.0000 (1/sec)

5.Void Fraction=0.3700

6.Temperature= 25.00 (degrees C)

7.KA= 0.1928 (kgmol gas/kgmol ads, O2)

8.KB= 0.0687 (kgmol gas/kgmol ads, N2)

9.B= 3.059 (kgmol N2 ads/cubic meter)

KA, KB <0 are calculated from temperature

Enter number (zero to exit):

7

7.KA= 0.1928 (kgmol gas/kgmol ads, O2)

Enter new value

-1

1.Zone # 2

2.Bed Diameter= 6.8000 (cm)

3.Zone Length= 39.4000 (cm)

4.Diffusion Coefficient= 200.0000 (1/sec)

5.Void Fraction=0.3700

6.Temperature= 25.00 (degrees C)

7.KA= -1.0000 (kgmol gas/kgmol ads, O2)

8.KB= 0.0687 (kgmol gas/kgmol ads, N2)

9.B= 3.059 (kgmol N2 ads/cubic meter)

KA, KB <0 are calculated from temperature

Enter number (zero to exit):

8

8.KB= 0.0687 (kgmol gas/kgmol ads, N2)

Enter new value

-1

1.Zone # 2

2.Bed Diameter= 6.8000 (cm)

3.Zone Length= 39.4000 (cm)

4.Diffusion Coefficient= 200.0000 (1/sec)

5.Void Fraction=0.3700

6.Temperature= 25.00 (degrees C)

7.KA= -1.0000 (kgmol gas/kgmol ads, O2)

8.KB= -1.0000 (kgmol gas/kgmol ads, N2)

9.B= 3.059 (kgmol N2 ads/cubic meter)

KA, KB <0 are calculated from temperature

Enter number (zero to exit):

1

1.Zone # 2

Enter new value

1

Zone Parameter Changes

Enter number to change parameter

Enter zero to exit

1.Zone # 1
2.Bed Diameter= 13.2000 (cm)
3.Zone Length= 39.4000 (cm)
4.Diffusion Coefficient= 200.0000 (1/sec)
5.Void Fraction=0.3700
6.Temperature= 25.00 (degrees C)
7.KA= 0.1928 (kgmol gas/kgmol ads, O2)
8.KB= 0.0687 (kgmol gas/kgmol ads, N2)
9.B= 3.059 (kgmol N2 ads/cubic meter)
KA, KB <0 are calculated from temperature
Enter number (zero to exit):

1

1.Zone # 1
Enter new value

2

Zone Parameter Changes

Enter number to change parameter

Enter zero to exit

1.Zone # 2
2.Bed Diameter= 6.8000 (cm)
3.Zone Length= 39.4000 (cm)
4.Diffusion Coefficient= 200.0000 (1/sec)
5.Void Fraction=0.3700
6.Temperature= 25.00 (degrees C)
7.KA= 0.1928 (kgmol gas/kgmol ads, O2)
8.KB= 0.0687 (kgmol gas/kgmol ads, N2)
9.B= 3.059 (kgmol N2 ads/cubic meter)
KA, KB <0 are calculated from temperature
Enter number (zero to exit):

0

Do you want to go back for more changes?

N

Do you want to save the data as BEA1985.MSV?

N

Enter the new seven letter(max) filename without any extension

EX1

The file EX1 .MSV
does not exist

Therefore, the data can be stored on EX1 .MSV

This is RUNTBL, which reads and modifies data files for
the OBOGS simulator (APR 1988)

Please enter the name of the file to be read
Modified files can be saved under a different name later
The filename should be seven or less characters and the
extension
should be TBL
(do NOT include the extension here)
BEA1985

Do you want to modify BEA1985.TBL?

Y

Run Table

Modify comment lines
OBOGS Simulation 29-MAR-88 08:17:27 Data From BEA1985.TBL
This is a simple run for 30 SLPM breathing rate, 30 sec. run.
Created by Glenn Munkvold, 14 Jan 1988

Do you want to modify the data comments?

Y

Comment changes: After each line of comments you have four choices:

1. Simply type a new comment to supercede old comment
2. Type \ to erase a comment line
3. Type + immediately followed by text to append to old comment
4. Type @ to leave comment unchanged

This is a simple run for 30 SLPM breathing rate, 30 sec. run.
This is a simple run for 30 L SLPM breathing rate, 35 sec. run.
Created by Glenn Munkvold, 14 Jan 1988

@

@

@

@

This is a simple run for 30 SLPM breathing rate, 35 sec. run.
Created by Glenn Munkvold, 14 Jan 1988

Do you want to modify the data comments further?

n

- 1.Space Step Size= 1.500000 (cm)
- 2.Time Step= 0.050000 (sec)
- 3.Simulation Duration= 30.0000 (seconds)
- 4.Number of Output Points= 100.
- 5.Product Flowrate

```

        (Initially 30.00 STD lpm)
6.Inlet Oxygen Concentration
  (Initially 0.2100 mole fraction)
7.Supply Pressure
  (Initially 0.308200 MPa, absolute)
8.Outlet Pressure
  (Initially 0.101300 MPa, absolute)
9.Cycle Time
  (Initially 10.7000 seconds)
10.Snap Shots
  (Initially at 0.1000E+07 seconds)
    Enter number (zero to exit):
3
  Simulation Time (seconds)
    Current value= 30.0000
    Enter new value
35

1.Space Step Size= 1.500000 (cm)
2.Time Step= 0.050000 (sec)
3.Simulation Duration= 35.0000 (seconds)
4.Number of Output Points= 100.
5.Product Flowrate
  (Initially 30.00 STD lpm)
6.Inlet Oxygen Concentration
  (Initially 0.2100 mole fraction)
7.Supply Pressure
  (Initially 0.308200 MPa, absolute)
8.Outlet Pressure
  (Initially 0.101300 MPa, absolute)
9.Cycle Time
  (Initially 10.7000 seconds)
10.Snap Shots
  (Initially at 0.1000E+07 seconds)
    Enter number (zero to exit):
10
  Snap Shot Unit (>3)          Time (sec)
1. 0.0000E+00          0.1000E+07
2. 0.0000E+00          0.1000E+07
3. 0.0000E+00          0.1000E+07
4. 0.0000E+00          0.1000E+07
5. 0.0000E+00          0.1000E+07
    Enter number to change parameter
    Enter zero to exit

1
  Enter new parameter value
4
  Enter corresponding start time (sec)
15
1. 0.4000E+01          0.1500E+02
2. 0.0000E+00          0.1000E+07
3. 0.0000E+00          0.1000E+07
4. 0.0000E+00          0.1000E+07
5. 0.0000E+00          0.1000E+07
    Enter number to change parameter
    Enter zero to exit

```

0

```
1.Space Step Size= 1.500000 (cm)
2.Time Step= 0.050000 (sec)
3.Simulation Duration= 35.0000 (seconds)
4.Number of Output Points= 100.
5.Product Flowrate
  (Initially 30.00 STD lpm)
6.Inlet Oxygen Concentration
  (Initially 0.2100 mole fraction)
7.Supply Pressure
  (Initially 0.308200 MPa, absolute)
8.Outlet Pressure
  (Initially 0.101300 MPa, absolute)
9.Cycle Time
  (Initially 10.7000 seconds)
10.Snap Shots
  (Initially at 0.1500E+02 seconds)
  Enter number (zero to exit):
```

0

Do you want to save the data as BEA1985.TBL?

N

Enter the new seven letter(max) filename without any extension

EX1

The file EX1 .TBL

does not exist

Therefore, the data can be stored on EX1 .TBL

Data files are edited

Do you want to quit?

N

```
Please wait; TIME = 1.799999
Please wait; TIME = 3.549998
Please wait; TIME = 5.250002
```

.....

```
Please wait; TIME = 31.54980
Please wait; TIME = 33.29978
Please wait; TIME = 35.04975
```

FORTRAN STOP

vms>

Run Time Messages

The only run time messages are the 'Please wait;...' messages above and a message from BEDP when the bed pressures are not found after twenty iterations of the main BEDP loop.

Input Files

EX1.TBL

OBOGS Simulation 29-MAR-88 08:19:00 Data From EX1 .TBL
This is a simple run for 30 SLPM breathing rate, 35 sec. run.
Created by Glenn Munkvold, 14 Jan 1988

Space Step (cm)	
Current value=	1.5000
Time Step (seconds)	
Current value=	0.0500
Simulation Time (seconds)	
Current value=	35.0000
Number of Output Points	
Current value=	100.0000
Breathing Flow (STD lpm)	Time (sec)
0.3000E+02	0.0000E+00
0.0000E+00	0.1000E+06
0.0000E+00	0.1000E+06
0.0000E+00	0.1000E+06
0.0000E+00	0.1000E+06
Inlet O2 (mole fraction)	Time (sec)
0.2100E+00	0.0000E+00
0.2100E+00	0.1000E+06
0.2100E+00	0.1000E+06
0.2100E+00	0.1000E+06
0.2100E+00	0.1000E+06
Supply Pressure (MPa,abs)	Time (sec)
0.3082E+00	0.0000E+00
0.0000E+00	0.1000E+06
0.0000E+00	0.1000E+06
0.0000E+00	0.1000E+06
Exhaust Pressure (MPa,abs)	Time (sec)
0.1013E+00	0.0000E+00
0.0000E+00	0.1000E+06
0.0000E+00	0.1000E+06
0.0000E+00	0.1000E+06
0.0000E+00	0.1000E+06
Cycle Time (seconds)	Time (sec)
0.1070E+02	0.0000E+00
0.0000E+00	0.1000E+06
0.0000E+00	0.1000E+06
0.0000E+00	0.1000E+06
0.0000E+00	0.1000E+06
Snap Shot Unit (>3)	Time (sec)
0.4000E+01	0.1500E+02
0.0000E+00	0.1000E+07
0.0000E+00	0.1000E+07
0.0000E+00	0.1000E+07
0.0000E+00	0.1000E+07

EX1.MSV

OBOGS Simulation 29-MAR-88 08:17:18 Data From EX1 .MSV
 This is a data file for the F-16 OBOGS by Bendix.
 It is derived from the hard-wired defaults in the
 simulator by Brian Yang. This set was input by
 Glenn Munkvold, 4/19/87. This run is an attempt
 to duplicate Beaman's 1985 results. Two bed zones are used.

Valve	Diameter(cm)	Discharge Coeff

Bypass 0.1905 0.6000
Supply 0.7777 0.7000
Outlet 1.1143 0.7000
Plenum Volume= 0.180E-02 (cubic meters)
Number of Bed Zones= 2 (maximum is 5)

Zone # 1
Bed Diameter= 13.2000 (cm)
Zone Length= 39.4000 (cm)
Diffusion Coefficient= 200.0000 (1/sec)
Void Fraction=0.3700
Temperature= 25.00 (degrees C)
KA= 0.1928 (kgmol gas/kgmol ads, O2)
KB= 0.0687 (kgmol gas/kgmol ads, N2)
B= 3.059 (kgmol N2 ads/cubic meter)
KA, KB <0 are calculated from temperature

Zone # 2
Bed Diameter= 6.8000 (cm)
Zone Length= 39.4000 (cm)
Diffusion Coefficient= 200.0000 (1/sec)
Void Fraction=0.3700
Temperature= 25.00 (degrees C)
KA= 0.1928 (kgmol gas/kgmol ads, O2)
KB= 0.0687 (kgmol gas/kgmol ads, N2)
B= 3.059 (kgmol N2 ads/cubic meter)
KA, KB <0 are calculated from temperature

Output Files

PSA.DAT

OBOGS Simulation 29-MAR-88 08:17:17 Data From EX1 .MSV
This is a data file for the F-16 OBOGS by Bendix.
It is derived from the hard-wired defaults in the
simulator by Brian Yang. This set was input by
Glenn Munkvold, 4/19/87. This run is an attempt
to duplicate Beaman's 1985 results. Two bed zones are used.

Valve	Diameter(cm)	Discharge Coeff
Bypass	0.1905	0.6000
Supply	0.7777	0.7000
Outlet	1.1143	0.7000

Plenum Volume= 0.180E-02 (cubic meters)
Number of Bed Zones= 2 (maximum is 5)

Zone # 1
Bed Diameter= 13.2000 (cm)
Zone Length= 39.4000 (cm)
Diffusion Coefficient= 200.0000 (1/sec)
Void Fraction=0.3700
Temperature= 25.00 (degrees C)
KA= 0.1928 (kgmol gas/kgmol ads, O2)
KB= 0.0687 (kgmol gas/kgmol ads, N2)
B= 3.059 (kgmol N2 ads/cubic meter)
KA, KB <0 are calculated from temperature

Zone # 2

Bed Diameter= 6.8000 (cm)
 Zone Length= 39.4000 (cm)
 Diffusion Coefficient= 200.0000 (1/sec)
 Void Fraction=0.3700
 Temperature= 25.00 (degrees C)
 KA= 0.1928 (kgmol gas/kgmol ads, O2)
 KB= 0.0687 (kgmol gas/kgmol ads, N2)
 B= 3.059 (kgmol N2 ads/cubic meter)
 KA, KB <0 are calculated from temperature
 OBOGS Simulation 29-MAR-88 08:19:00 Data From EX1 .TBL
 This is a simple run for 30 SLPM breathing rate, 35 sec. run.
 Created by Glenn Munkvold, 14 Jan 1988

Space Step (cm)	
Current value=	1.5000
Time Step (seconds)	
Current value=	0.0500
Simulation Time (seconds)	
Current value=	35.0000
Number of Output Points	
Current value=	100.0000
Breathing Flow (STD lpm)	Time (sec)
0.3000E+02	0.0000E+00
0.0000E+00	0.1000E+06
0.0000E+00	0.1000E+06
0.0000E+00	0.1000E+06
0.0000E+00	0.1000E+06
Inlet O2 (mole fraction)	Time (sec)
0.2100E+00	0.0000E+00
0.2100E+00	0.1000E+06
0.2100E+00	0.1000E+06
0.2100E+00	0.1000E+06
0.2100E+00	0.1000E+06
Supply Pressure (MPa,abs)	Time (sec)
0.3082E+00	0.0000E+00
0.0000E+00	0.1000E+06
0.0000E+00	0.1000E+06
0.0000E+00	0.1000E+06
0.0000E+00	0.1000E+06
Exhaust Pressure (MPa,abs)	Time (sec)
0.1013E+00	0.0000E+00
0.0000E+00	0.1000E+06
0.0000E+00	0.1000E+06
0.0000E+00	0.1000E+06
0.0000E+00	0.1000E+06
Cycle Time (seconds)	Time (sec)
0.1070E+02	0.0000E+00
0.0000E+00	0.1000E+06
0.0000E+00	0.1000E+06
0.0000E+00	0.1000E+06
0.0000E+00	0.1000E+06
Snap Shot Unit (>3)	Time (sec)
0.4000E+01	0.1500E+02
0.0000E+00	0.1000E+07
0.0000E+00	0.1000E+07
0.0000E+00	0.1000E+07

0.0000E+00		0.1000E+07	
OBOGS Output			
Time	O2 Mole	O2 Mass	Total Mass
Seconds	Fraction	Out g/s	In g/s
0.3500	0.210136	0.1501	23.9619
0.7000	0.210817	0.1506	23.9619
1.0500	0.212112	0.1515	23.9619
.....			
.....			
33.9498	0.439622	0.3140	22.5666
34.2998	0.437603	0.3126	21.1749
34.6498	0.437024	0.3122	19.3181
34.9998	0.437904	0.3128	17.0322

START.WRM

OBOGS Simulation 29-MAR-88 08:17:18 Data From EX1 .MSV
 This is a data file for the F-16 OBOGS by Bendix.
 It is derived from the hard-wired defaults in the
 simulator by Brian Yang. This set was input by
 Glenn Munkvold, 4/19/87. This run is an attempt
 to duplicate Beaman's 1985 results. Two bed zones are used.

Valve	Diameter(cm)	Discharge Coeff
Bypass	0.1905	0.6000
Supply	0.7777	0.7000
Outlet	1.1143	0.7000

Plenum Volume= 0.180E-02 (cubic meters)
 Number of Bed Zones= 2 (maximum is 5)

Zone # 1
 Bed Diameter= 13.2000 (cm)
 Zone Length= 39.4000 (cm)
 Diffusion Coefficient= 200.0000 (1/sec)
 Void Fraction=0.3700
 Temperature= 25.00 (degrees C)
 KA= 0.1928 (kgmol gas/kgmol ads, O2)
 KB= 0.0687 (kgmol gas/kgmol ads, N2)
 B= 3.059 (kgmol N2 ads/cubic meter)
 KA, KB <0 are calculated from temperature

Zone # 2
 Bed Diameter= 6.8000 (cm)
 Zone Length= 39.4000 (cm)
 Diffusion Coefficient= 200.0000 (1/sec)
 Void Fraction=0.3700
 Temperature= 25.00 (degrees C)
 KA= 0.1928 (kgmol gas/kgmol ads, O2)
 KB= 0.0687 (kgmol gas/kgmol ads, N2)
 B= 3.059 (kgmol N2 ads/cubic meter)
 KA, KB <0 are calculated from temperature

OBOGS Simulation 29-MAR-88 08:19:00 Data From EX1 .TBL
 This is a simple run for 30 SLPM breathing rate, 35 sec. run.
 Created by Glenn Munkvold, 14 Jan 1988

```

Space Step (cm)
  Current value=      1.5000
Time Step (seconds)
  Current value=      0.0500
Simulation Time (seconds)
  Current value=     35.0000
Number of Output Points
  Current value=     100.0000
Breathing Flow (STD lpm)      Time (sec)
0.3000E+02      0.0000E+00
0.0000E+00      0.1000E+06
0.0000E+00      0.1000E+06
0.0000E+00      0.1000E+06
0.0000E+00      0.1000E+06
0.0000E+00      0.1000E+06
Inlet O2 (mole fraction)      Time (sec)
0.2100E+00      0.3000E+00
0.2100E+00      0.1000E+06
0.2100E+00      0.1000E+06
0.2100E+00      0.1000E+06
0.2100E+00      0.1000E+06
0.2100E+00      0.1000E+06
Supply Pressure (MPa,abs)      Time (sec)
0.3082E+00      0.0000E+00
0.0000E+00      0.1000E+06
0.0000E+00      0.1000E+06
0.0000E+00      0.1000E+06
0.0000E+00      0.1000E+06
0.0000E+00      0.1000E+06
Exhaust Pressure (MPa,abs)      Time (sec)
0.1013E+00      0.0000E+00
0.0000E+00      0.1000E+06
0.0000E+00      0.1000E+06
0.0000E+00      0.1000E+06
0.0000E+00      0.1000E+06
0.0000E+00      0.1000E+06
Cycle Time (seconds)      Time (sec)
0.1070E+02      0.0000E+00
0.0000E+00      0.1000E+06
0.0000E+00      0.1000E+06
0.0000E+00      0.1000E+06
0.0000E+00      0.1000E+06
0.0000E+00      0.1000E+06
Snap Shot Unit (>3)      Time (sec)
0.4000E+01      0.1500E+02
0.0000E+00      0.1000E+07
0.0000E+00      0.1000E+07
0.0000E+00      0.1000E+07
0.0000E+00      0.1000E+07
0.0000E+00      0.1000E+07
Bed Profiles. Time=      34.9998 sec Time in Cycle=      2.9500 sec
Plenum Vars: YO2= 0.4379      YO2M1= 0.4586
Run on 29-MAR-88      at 08:20:03
Bed# 1 Pressure= 0.2654E+00 MPa
  Distance O2 Mole Velocity O2 Ads      Tot Ads
#      (m)      Frac      (m/s)      (kgmol/m3)      (kgmol/m3)
1 0.0000E+00 0.2100 0.1089E+01 0.1171E+00 0.9961E+00
2 0.1515E-01 0.2115 0.1060E+01 0.1180E+00 0.9957E+00
3 0.3031E-01 0.2132 0.1030E+01 0.1189E+00 0.9953E+00
4 0.4546E-01 0.2150 0.1001E+01 0.1199E+00 0.9949E+00
5 0.6062E-01 0.2170 0.9714E+00 0.1210E+00 0.9944E+00
.....
.....
.....

```

50 0.7425E+00 0.6560 0.4924E+00 0.3618E+00 0.8167E+00
 51 0.7577E+00 0.7363 0.4397E+00 0.4096E+00 0.7708E+00
 52 0.7728E+00 0.7378 0.4107E+00 0.4120E+00 0.7725E+00
 53 0.7880E+00 0.6890 0.3950E+00 0.3859E+00 0.8044E+00

Bed# 2 Pressure= 0.1073E+00 MPa

#	Distance (m)	O2 Mole Frac	Velocity (m/s)	O2 Ads (kgmol/m3)	Tot Ads (kgmol/m3)
1	0.0000E+00	0.1897	-.1285E+01	0.4244E-01	0.4799E+00
2	0.1515E-01	0.1921	-.1250E+01	0.4298E-01	0.4793E+00
3	0.3031E-01	0.1947	-.1215E+01	0.4356E-01	0.4787E+00
4	0.4546E-01	0.1974	-.1180E+01	0.4416E-01	0.4780E+00
5	0.6062E-01	0.2002	-.1145E+01	0.4479E-01	0.4774E+00

.....

 50 0.7425E+00 0.4547 -.7966E+00 0.1031E+00 0.4132E+00
 51 0.7577E+00 0.5292 -.7229E+00 0.1197E+00 0.3915E+00
 52 0.7728E+00 0.6111 -.6529E+00 0.1378E+00 0.3662E+00
 53 0.7880E+00 0.6889 -.5939E+00 0.1549E+00 0.3403E+00

SNAPXX.DAT

Bed Profiles. Time= 15.0000 sec Time in Cycle= 4.3000 sec

Plenum Vars: YO2= 0.3357 YO2M1= 0.4059

Run on 29-MAR-88 at 08:19:25

Bed# 1 Pressure= 0.3046E+00 MPa

#	Distance (m)	O2 Mole Frac	Velocity (m/s)	O2 Ads (kgmol/m3)	Tot Ads (kgmol/m3)
1	0.0000E+00	0.2100	0.2984E+00	0.1337E+00	0.1099E+01
2	0.1515E-01	0.2110	0.2926E+00	0.1344E+00	0.1099E+01
3	0.3031E-01	0.2122	0.2868E+00	0.1352E+00	0.1099E+01
4	0.4546E-01	0.2136	0.2810E+00	0.1360E+00	0.1098E+01
5	0.6062E-01	0.2151	0.2751E+00	0.1370E+00	0.1098E+01

.....

 50 0.7425E+00 0.2775 0.4206E+00 0.1770E+00 0.1084E+01
 51 0.7577E+00 0.2977 0.4125E+00 0.1905E+00 0.1078E+01
 52 0.7728E+00 0.3434 0.4006E+00 0.2207E+00 0.1064E+01
 53 0.7880E+00 0.4163 0.3835E+00 0.2680E+00 0.1038E+01

Bed# 2 Pressure= 0.1015E+00 MPa

#	Distance (m)	O2 Mole Frac	Velocity (m/s)	O2 Ads (kgmol/m3)	Tot Ads (kgmol/m3)
1	0.0000E+00	0.1942	-.2438E+00	0.4123E-01	0.4562E+00
2	0.1515E-01	0.1967	-.2431E+00	0.4177E-01	0.4556E+00
3	0.3031E-01	0.1994	-.2424E+00	0.4233E-01	0.4550E+00
4	0.4546E-01	0.2020	-.2417E+00	0.4289E-01	0.4544E+00
5	0.6062E-01	0.2047	-.2410E+00	0.4346E-01	0.4538E+00

.....

 50 0.7425E+00 0.5255 -.6809E+00 0.1118E+00 0.3698E+00
 51 0.7577E+00 0.5081 -.6917E+00 0.1086E+00 0.3727E+00
 52 0.7728E+00 0.4674 -.7153E+00 0.1002E+00 0.3827E+00
 53 0.7880E+00 0.4163 -.7452E+00 0.8936E-01 0.3966E+00

Subroutine Descriptions

Main Program BINKLEY

COMMON Blocks

/GASES/
/INIT/
/RDAT2/
/VLVCON/

Subroutines Called

PROC
SNAPOUT
STARTUP

Description

This is the main program. It calls each of the three main sections of code: data editing, the main processor, and the post processor (SNAPOUT). PARAMETERS and many of the COMMON variables are defined. STARTUP is called first, then PROC, then SNAPOUT.

ADSREAD

COMMON Blocks

/PLEN/

Subroutines Called

NONE

Description

This subroutine reads in SNAPOUT files, as warm start input files.

BEDDAT

COMMON Blocks

/BDAT/
/BDAT2/

Subroutines Called

CCNG
EXTADD
FILCHK
PREAD
VPCNG
YES
ZCNG

Description

This routine calls PREAD to read the .MSV input files, and provides the mechanism for interactive modification of the input parameters. The COMMON blocks /BDAT/ and /BDAT2/ are loaded here, and the variables in those COMMON blocks are defined in this routine. The subroutine has two modes: normal and warm start. The warm start mode

disables all the interactive parts of BEDDAT and routines below. Normal mode allows the user to change parameters interactively, before beginning the actual simulation. Parameters cannot be changed during the simulation. BEDDAT also calculates the flow area (FAREA, from the bed diameter and void fraction ϵ) and the value of β for each zone. β is $d(1-\epsilon)/\epsilon$.

BEDP

COMMON Blocks

/BDAT/
/FPRESS/
/GASES/

Subroutines Called

FALSEP
FINDP

Description

Subroutine BEDP finds bed pressures using equation (10) from Beaman (1985). This equation is used by first assuming that the pressure in bed #1 (PBED(1)) is correct. The current bed pressure in bed #2 (PBED(2)) is used as a first guess for the evaluation of equation (10). This equation is evaluated as function FINDP.

$$1. - \frac{\int_0^L \beta A C^* dx}{C \left(Q_L - Q_0 + \int_0^L \beta A dx \right)} = 0 \quad (10)$$

$$1. - \frac{RT \int_0^L \beta A C^* dx}{P_{BED} \left(Q_L - Q_0 + \int_0^L \beta A dx \right)} = 0$$

The integrals are evaluated in EQUIL:

$$\left(\int_0^L \beta A C^* dx \right)_{BED \#1} = CINTINT(1)$$

$$\left(\int_0^L \beta A C^* dx \right)_{BED \#2} = CINTINT(2)$$

$$\left(\int_0^L \beta A dx \right)_{\text{BED \#1}} = \text{BCINT}(1)$$

$$\left(\int_0^L \beta A dx \right)_{\text{BED \#2}} = \text{BCINT}(2)$$

Q_L and Q_0 are found through FINDP calls to FINDQP and FINDQZ with bed pressures as loaded into COMMON /FPRESS/.

In a typical sequence, BEDP takes the current bed pressures and loads them in /FPRESS/. The pressure in bed #1 is stored as POLD. FINDP is called with bed #2 pressure to find the residual (RESID1) of equation (10) (the present bed pressures will not cause equation (10) to be satisfied because the adsorbed phase concentrations have been integrated in time; i.e., the bed pressures assumed in the call to BEDP are actually old values. We need to find the current values that will satisfy the equation). A new guess at bed #2 pressure is made by multiplying the current guess (PJ1) by (1. - RESID1). From the new guess (PJ2), a new residual (RESID2) is found. If the residuals are of different signs, then the pressure in bed #2 that satisfies equation (10) has been bracketed. Therefore, FALSEP is called to find the correct pressure more exactly using the false position algorithm. If the residuals are not of different sign, RESID1 and PJ1 are discarded and assigned the values of RESID2 and PJ2. A new RESID2 is calculated from a new guess at PJ2 (found, as before, by multiplying PJ1 by (1. - RESID1)). This continues until the proper pressure is bracketed. After bed #2 pressure has been found, the loop is executed again, with bed #2 pressure assumed correct, and RESID1 and RESID2 now being calculated through guesses at bed #1 pressure. When the proper bed pressure has been bracketed, FALSEP is called again to finish finding the value of bed #1 pressure. This current value is compared to POLD, the value of bed #1 pressure when the routine was originally called. If the relative difference in these values is within a preset tolerance, the proper pressures are known. If the relative difference is too great, POLD is assigned the current bed #1 pressure, and we loop on the beds again, as above, assuming bed #1 pressure is correct and guessing at bed #2 pressure.

After the proper pressures are found, inlet mass rates (QZMASS), inlet velocities (UZERO), outlet velocities (UL), and average total bed concentrations (CAVG) are found. Figure 3 is a flowchart of the pressure calculations in BEDP.

If the relative tolerance is not met for the bed pressure, BEDP exits after twenty iterations of the beds loop and writes a failure message to the screen.

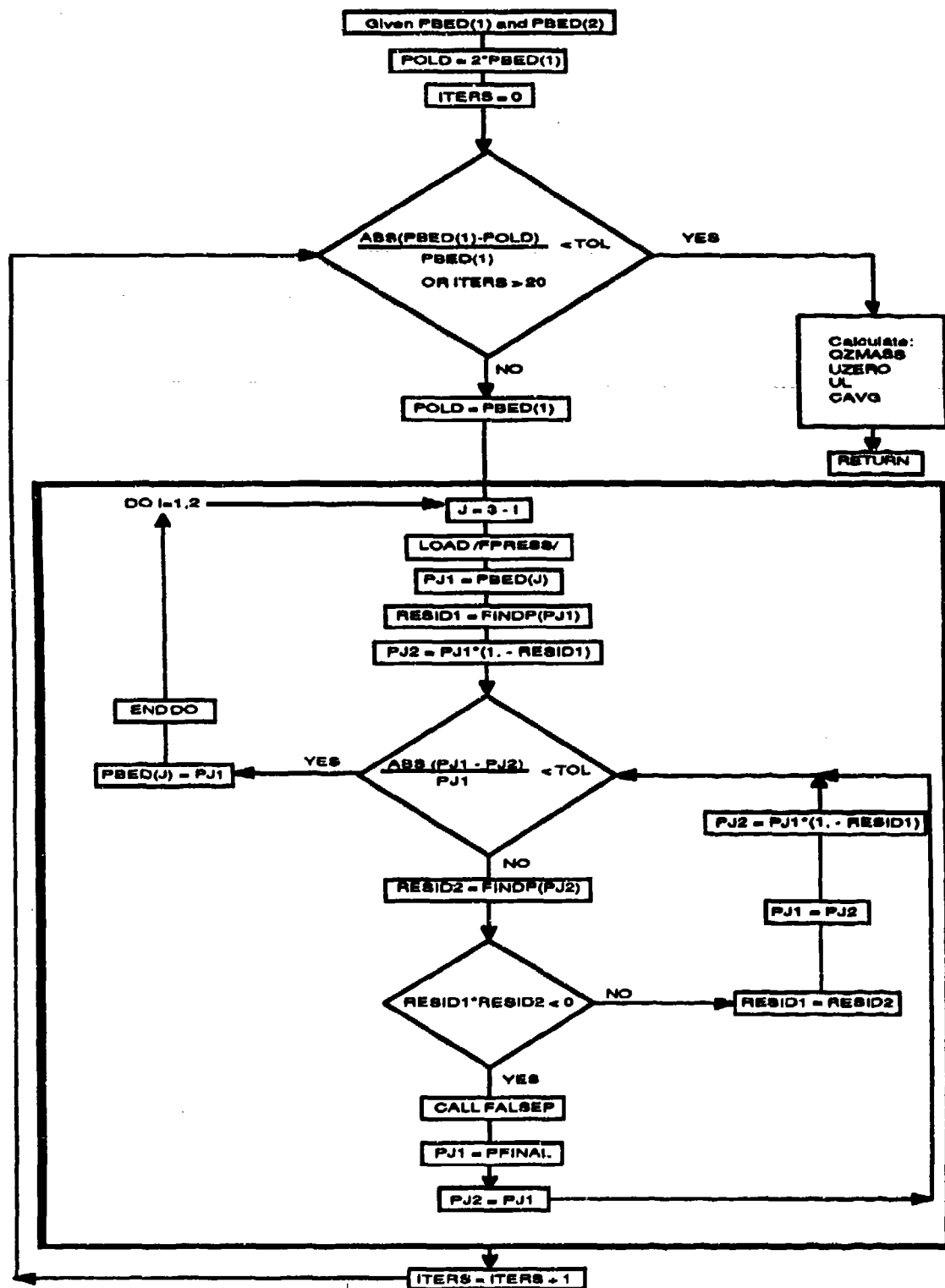


Figure 3. Flowchart of Calculations in Subroutine BEDP

CCNG

COMMON Blocks
NONE

Subroutines Called
DATE
TIME
YES

Description

This subroutine interactively changes and saves the comment lines for both the .MSV and .TBL files. Comment lines can be output without modification in a warm start without interactive instructions. The date and time of a run are also recorded and output. Simple editing functions are included (comment lines can be superceded, appended, removed, or left unchanged using single character format commands).

COUPISO

COMMON Blocks
/EQFCN/

Subroutines Called
FALSEP
FEQ1 (THROUGH FALSEP)

Description

COUPISO is actually the name of the file that contains the version of ISOTHM that solves the Beaman et al. (1983) equation (23) for the coupled isotherm case. Using the lower bound mentioned in the paper and an upper bound of $Z=1$, FALSEP is called with function FEQ1 as argument to solve for the interfacial nitrogen concentration (CINTN2). The interfacial oxygen concentration (CINTO2) and the total interfacial concentration (CINTOT) are calculated from the nitrogen concentration. Note that it may be necessary to run with a shorter time step for the coupled isotherm case. The shorter time step and additional calculations cause the coupled case to take longer to run than the uncoupled case.

DATE

COMMON Blocks
NONE

Subroutines Called
NONE

Description

This is a Vax FORTRAN subroutine used to obtain the system date. Various formats are available, depending on the specific date routine called and the variable type passed.

EQUIL

COMMON Blocks
NONE

Subroutines Called
ISOTHM
XGETPAM

Description

EQUIL takes the current adsorbed phase concentrations and finds the corresponding interfacial concentrations from the equilibrium isotherm. The BCINT and CINTINT integrals (see BEDP description) are calculated. BETACNT is also calculated. BETACNT is used in calculation of velocity profiles in FINDU, as is defined as:

$$\text{BETACNT}(J, \text{IBED}) = \beta (\Delta x) \frac{C_{j-1}^* + C_j^*}{2}$$

where j is a space lump number and IBED is bed number.

The calculations are done through all zones of each bed, with calls to XGETPAM at zone boundaries to get appropriate isotherm, mass transfer, and void fraction values.

EXTADD

COMMON Blocks
NONE

Subroutines Called
NONE

Description

This routine adds an extension to a filename. Essentially, it is simply a character string concatenation, as the filename and extension are both input. The completed filename is returned.

FALSEP

COMMON Blocks
NONE

Subroutines Called
FUNCTION NAME PASSED AS ARGUMENT

Description

Given the name of an EXTERNAL function and two initial guesses for the root of the function (the guesses must bracket the solution), FALSEP uses the false position algorithm to find the root of the equation. The function must be constructed such that its value will be zero for a solution to the equation. FALSEP returns after meeting an absolute error tolerance or after exceeding a loop limit.

FEQ1

COMMON Blocks
/EQFCN/

Subroutines Called
NONE

Description

This is the function that evaluates equation (23) from the Beaman et al. (1983) paper for the coupled isotherm case. It is called by FALSEP. See the COUPISO description.

FILCHK

COMMON Blocks
NONE

Subroutines Called
NONE

Description

Given a filename, FILCHK checks to see if the file already exists in the current directory. A logical variable is returned .TRUE. if the file already exists.

FINDFLX

COMMON Blocks
/RDAT2/

Subroutines Called
INTBKWD
INTFWD

Description

Given the velocity distribution (from FINDU), this routine calculates the oxygen flux through the bed according to equations (11) in Beaman, et al. (1983). FINDFLX classifies the case according to the value of the input parameter NUZERO. NUZERO can be the lump number where the velocity in the bed is zero (positive values) or a flag that indicates the type of flow in the bed (zero or negative values). INTBKWD is called for flow backward through the bed; INTFWD is called for forward flow through the bed. If flow is into or out of both ends of the bed, both INTFWD and INTBKWD are called. See FINDU description for a table of NUZERO values. It is important to note that flux integrations must be in the direction of flow in order to match assumptions made in solving the flux equations.

FINDP

COMMON Blocks
/FPRESS/
/GASES/

Subroutines Called

FINDQP
FINDQZ

Description

This function is used in calculation of the bed pressures. See the BEDP description for an overview of this calculation. FINDP evaluates the residual of equation (10) in Beaman (1985), and is called by FALSEP as well as BEDP.

FINDQP

COMMON Blocks
NONE

Subroutines Called
VALVE

Description

This function returns the volume flowrate through the bypass valve at the current bed pressure. It is called by FINDP in evaluation of bed pressures.

FINDQZ

COMMON Blocks
NONE

Subroutines Called
VALVE

Description

This function returns the volume flowrate through either the supply or exhaust valve at the current bed pressure. The valve selection is made through the argument NVLV. NVLV is one for the supply valve; it is two for the exhaust valve. This distinction is important if the diameters and/or discharge coefficients of these valves differ. FINDQZ is called by FINDP in evaluation of bed pressures.

FINDU

COMMON Blocks
/RDAT2/

Subroutines Called
XGETPAM

Description

This routine spatially integrates equation (1) from Beaman (1985) using the trapezoidal rule. Once the bed pressures are known, the volumetric flowrates at each end of the beds are known. The adsorbed phase concentrations are also known, so it is possible to calculate the velocity profile for a bed as:

$$u_j = u_{j-1} + \beta \left(\frac{C^*_j + C^*_{j-1}}{2C} - 1 \right) \Delta x$$

$$u_j = u_{j-1} + \left(\frac{\text{BETACNT}(J)}{C} \right) - \beta \Delta x$$

BETACNT is calculated in EQUIL, and includes a bed number as one of its indices. It is necessary to modify u_{j-1} by a ratio of zone cross-sectional areas when a zone boundary is crossed. This must be done to ensure volumetric flow continuity across the area change.

FINDU also calculates initial fluxes in the bed, if necessary. No fluxes are calculated for any of the cases below where NUZERO is negative. If NUZERO is non-negative, its value will be the lump number that corresponds most closely to the zero velocity point in the bed. In this case, flow is either into or out of both ends of the bed, and an initial flux is necessary for FINDFLX to properly integrate the flux equations (for the negative NUZERO cases, FINDFLX can use inlet or outlet velocities to determine the initial flux).

<u>UL</u>	<u>UZ</u>	<u>NUZERO</u>
-,0	-	-2
+	+,0	-1
+	-	+
-	+	+

GETCYC

COMMON Blocks
NONE

Subroutines Called
NONE

Description

This subroutine puts supply or exhaust pressure on the beds, depending on the cycle time and elapsed time in the current cycle. Odd half-cycles have supply pressure (PSUP) on bed #1 and exhaust pressure (POUT) on bed #2.

INTBKWD

COMMON Blocks
/RDAT2/

Subroutines Called
XGETPAM

Description

This routine integrates the flux equation in the negative x direction. Arguments include the starting lump number for the backwards integration. From this lump number,

the bed parameters are determined with a call to XGETPAM. For backward axial integrations through the bed, calls to XGETPAM are made using a different condition than in the forward direction to ensure that the proper bed parameters are consistently applied for the case of coincident nodes and zone boundaries. The program uses parameters from the left side of zone boundaries coincident with nodes.

INTFWD

COMMON Blocks
/RDAT2/

Subroutines Called
XGETPAM

Description

This subroutine integrates the flux equation forward through the bed. Arguments include the starting lump number for the integration. Again, this lump number is used with a call to XGETPAM to determine the proper bed parameters. The standard conditions are used for changing parameters at zone boundaries (see the source code). The program uses the parameters from the left side of zone boundaries that are coincident with nodes.

INTTIM

COMMON Blocks
/RDAT2/

Subroutines Called
XGETPAM

Description

This routine integrates the adsorbed phase concentrations in time using a simple forward difference for the approximation of the derivatives with respect to time. XGETPAM is called find the proper mass transfer coefficient values.

ISOTHM

COMMON Blocks
NONE (SEE COUPISO)

Subroutines Called
NONE (SEE COUPISO)

Description

This subroutine calculates the interfacial concentrations given the adsorbed phase concentrations. It may calculate the coupled or uncoupled isotherms, depending on the version of the routine used (see COUPISO description). The isotherm derived by Beaman, et al. (1983) is used. A relatively straightforward analysis reveals the connection between the Beaman isotherm and the classical Langmuir isotherm. Beaman's isotherm is:

$$C^* = \frac{Kn}{\left(1 - \frac{n}{b}\right)}$$

The Langmuir isotherm is:

$$\theta = \frac{q}{q_s} = \frac{kp}{1 + kp}$$

Comparing, we find:

$$K = \frac{1}{kq_sRT\rho}$$

and

$$b = q_s\rho$$

Therefore, if the parameters of the Langmuir isotherm are known for some other gases, the model can be used to model these gases in a PSA unit. This course of action is fraught with risks, however; the model assumes that the mass transfer coefficient is the same for both components. If the difference between mass transfer coefficients is important, the results of using this model to simulate PSA operation with other feed gases may be useless. Furthermore, the model has been shown to work well for the case of high mass transfer coefficient (nearly equilibrium separation). Results are not expected to be as good for a mass transfer dominated system, even if the mass transfer coefficients are nearly equal.

KTCORR

COMMON Blocks
NONE

Subroutines Called
NONE

Description

This routine uses a power law temperature correlation from some unpublished work by David Walshak for the temperature dependence of the oxygen and nitrogen isotherm parameters in the Beaman model. This three parameter correlation is expected to work well for interpolation, and therefore should only be used for temperatures between -50°C and 100°C. The value for b, the limiting amount of nitrogen adsorbed, is not expected to depend on temperature. Note that a two parameter correlation is possible if the typical exponential temperature dependence of the Henry constant is used in the Langmuir comparison above. For this case:

$$K = \frac{1}{k_0 \exp\left(\frac{-\Delta H}{RT}\right) q_s RT \rho}$$

The two parameter fit has been found and proves inferior (according to available data) to the three parameter fit used in KTCORR for interpolation. The divergence of the power law correlation and the two parameter correlation outside the above mentioned temperature range emphasizes that the correlation in the program must only be used within the stated temperature limits.

PLENUM

COMMON Blocks
/PLEN/

Subroutines Called
NONE

Description

This routine models the outlet plenum on the OBOGS unit as two first order mixers in series. Half the plenum volume is used to calculate the time constant for each mixer.

PREAD

COMMON Blocks
NONE

Subroutines Called
NONE

Description

PREAD reads in all the .MSV file parameters.

PROC

COMMON Blocks
/BDAT/
/GASES/
/INIT/
/PLEN/
/RDAT2/

Subroutines Called
BEDP
EQUIL
FINDFLX
FINDU
GETCYC
INTTIM
PLENUM
SNAPOUT
TGETPAM

Description

PROC controls the main processing of the simulator. Table headings, final bed profiles (through SNAPOUT), and concentrator output are written. The amount of output generated is selected by the user in a .TBL file and is implemented using a printing cycle

time. A message is also written to the screen (giving the simulation time elapsed) during execution to tell the user that the program is still running and making progress. PROC calls subroutines in the following order: TGETPAM, GETCYC, EQUIL, BEDP, FINDU, FINDFLX, PLENUM, SNAPOUT, and INTTIM. SNAPOUT is only called once from PROC, at the end of execution. However, the call is before the call to INTTIM, so it appears before INTTIM in the list above.

RUNTBL

COMMON Blocks

/RDAT/
/RDAT2/

Subroutines Called

EXTADD
FILCHK
TPARAM
TREAD
YES

Description

This subroutine reads .TBL files (through TREAD) and modifies the run time parameters for a simulation. COMMON blocks /RDAT/ and /RDAT2/ are loaded and data sets are saved before exit from the routine. As in BEDDAT, a warm start causes the interactive sections of the routine to be disabled.

SNAPOUT

COMMON Blocks

/PLEN/

Subroutines Called

DATE
TIME

Description

SNAPOUT writes the bed profiles at the end of simulation as well as at intermediate times selected in the .TBL file. See the example file in the I/O section above. Elapsed simulation time, time in the current cycle, mole fractions in the mixers, date, and time are output as header information. The gas oxygen mole fraction and gas velocity are output as functions of lump number and total distance in the bed.

STARTUP

COMMON Blocks

/BDAT/
/GASES/
/INIT/
/PLEN/
/RDAT/

/RDAT2/

Subroutines Called

BEDDAT
RUNTBL
WRMST
XGETPAM
YES

Description

This routine initializes the entire simulation. It calls BEDDAT, WRMST, and RUNTBL to read parameters and load COMMON blocks. In a normal start, it also adjusts Δx to get an integer number of lumps in the bed and initializes the adsorbed gas arrays (assuming beds at equilibrium with air at atmospheric pressure).

TCHECK

COMMON Blocks

NONE

Subroutines Called

NONE

Description

TCHECK checks to see if, at the current simulation time, a particular run table parameter has a new value. Values in the time arrays are set equal to -1 after the time for implementation has passed. When no step changes are desired, the implementation times for succeeding available steps should be set to a large time (at a minimum, greater than the simulation time).

TGETPAM

COMMON Blocks

/RDAT/
/RDAT2/

Subroutines Called

TCHECK

Description

This routine calls TCHECK for each different time varying parameter. The values returned by TCHECK are then returned to PROC.

TIME

COMMON Blocks

NONE

Subroutines Called

NONE

Description

This is the Vax FORTRAN routine for finding the system time. The time is used in header information in output files.

TPARAM

COMMON Blocks
NONE

Subroutines Called
NONE

Description

TPARAM is called by RUNTBL to actually change the values of the run table parameters. It is called once for each different parameter and displays the table of values and start times for that parameter. In warm starts, TPARAM dumps all its output to the appropriate file in a single call.

TREAD

COMMON Blocks
NONE

Subroutines Called
NONE

Description

This is the routine called by RUNTBL to read in the .TBL file.

VALVE

COMMON Blocks
/BDAT2/
/LVCON/

Subroutines Called
NONE

Description

VALVE models all the valves in the system as orifices, according to Ezekiel and Shearer (1960). The diameter and discharge coefficients used in the code are determined by the valve numbers given in Figure 2 (1=supply, 2=exhaust, 3=bypass). In this simulation, the ratio of constant volume heat capacity to constant pressure heat capacity is taken to be that of air.

VPCNG

COMMON Blocks
NONE

Subroutines Called
NONE

Description

This routine interactively changes the diameters and discharge coefficients of the supply, exhaust, and bypass valves.

WRMST

COMMON Blocks

/BDAT/
/RDAT2/

Subroutines Called

ADSREAD
BEDDAT
RUNTBL

Description

This routine implements a warm start. Interactive modes are suspended in lower routines, and the adsorbed gas array initialization and Δx adjustment are done here instead of in STARTUP. ADSREAD reads in the START.WRM file information to restart the run. WARMST can also be used to restart a file from snap shot output. See the example files above. Essentially, to restart from a snap shot, a file must be created that has snap shot output appended to the header information from a PSA.DAT or START.WRM file. This file must be named START.WRM. A warm start will then use this file on start up. This capability may be useful in comparing responses to different step changes in input parameters, all starting from the same steady state.

XGETPAM

COMMON Blocks

/BDAT/

Subroutines Called

NONE

Description

This routine returns the proper bed parameters for either the lump number or distance in the bed passed as argument. If the lump number is negative, then the distance is used to find the zone, and the proper lump number is returned along with the bed parameters. Positive lump numbers are used directly to find the parameters. If the lump number passed is too large, the parameters for the last zone are returned and the lump number is returned as -1.

YES

COMMON Blocks

NONE

Subroutines Called

NONE

Description

YES is passed a single character. If the character is 'y' or 'Y', YES returns a logical variable set to .TRUE.. If the character is anything else, YES returns .FALSE..

ZCNG

COMMON Blocks

NONE

Subroutines Called

KTCORR

Description

This routine interactively modifies the zone specific parameters in the beds. As mentioned above, zone temperatures are only used in calculating isotherm parameters (the entire bed is assumed to be at the temperature of the first zone for the flow calculations). Zones can have different diameters, lengths, diffusion coefficients, void fractions, and isotherm coefficients.

Index

ADSREAD 8, 26, 41, 50
BEDDAT 5, 26, 39, 41, 51
BEDP 9, 27, 37, 56
BINKLEY 5, 26, 48
BINKLEY.FOR 11
CCNG 5, 26, 30, 59
COMMON Blocks
 /BDAT/ 10, 26, 27, 37, 38, 41, 51, 56, 97, 109, 111
 /BDAT2/ 10, 26, 40, 51, 86, 105
 /EQFCN/ 10, 30, 32, 116
 /FPRESS/ 10, 27, 32, 56, 67
 /GASES/ 10, 26, 27, 32, 37, 38, 48, 56, 67, 86, 97
 /INIT/ 10, 26, 37, 38, 48, 86, 97
 /PLEN/ 10, 26, 37, 38, 50, 83, 86, 95, 97
 /RDAT/ 10, 38, 39, 89, 97, 101
 /RDAT2/ 10, 26, 32, 33, 34, 35, 37, 38, 39, 41, 48, 65, 70, 74, 77, 80, 86, 89,
 97, 101, 109
 /VLVCON/ 10, 26, 40, 48, 105
COUPISO 30
COUPISO.FOR 11
DATE 5, 9, 30, 38
EQUIL 9, 31, 37, 61
EXTADD 5, 9, 26, 31, 38, 63
FALSEP 27, 30, 31, 118
FALSEP.FOR 11
FEQ1 11, 30, 32, 116
FILCHK 5, 9, 26, 32, 38, 64
FINDFLX 9, 32, 37, 65
FINDP 27, 32, 67
FINDQP 33, 68
FINDQZ 33, 69
FINDU 9, 33, 37, 70
GETCYC 9, 34, 37, 73
INTBKWD 32, 34, 74
INTFWD 32, 35, 77
INTTIM 9, 35, 37, 80
ISOTHM 5, 31, 35, 116, 117
ISOTHM.FOR 11
KTCORR 5, 36, 42, 82
LINK2.COM 11
LINKC.COM 11
PLENUM 9, 37, 83
PREAD 5, 26, 37, 84
PROC 5, 9, 26, 37, 86
RUNTBL 8, 38, 39, 41, 89
SNAPOUT 5, 9, 11, 26, 37, 38, 95
STARTUP 5, 26, 38, 97
TCHECK 39, 100
TGETPAM 9, 37, 39, 101
TIME 5, 9, 30, 38, 39
TPARAM 8, 38, 40, 102

TREAD 38, 40, 104
VALVE 5, 33, 40, 105
VPCNG 5, 26, 40, 106
WRMST 8, 39, 41, 109
XGETPAM 5, 9, 31, 33, 34, 35, 39, 41, 111
YES 5, 9, 26, 30, 38, 39, 41, 112
ZCNG 5, 26, 42, 113

References

- Beaman, J.J., A.J. Healey, and J. Werlin, "A Dynamic Model of a Molecular Sieve Bed with Nonlinear and Coupled Isotherms," I. Dyn. Sys. Meas. and Control, **105** (1983) 265.
- Beaman, J.J., "A Dynamic Model of a Pressure Swing Oxygen Generation System," I. Dyn. Sys. Meas. and Control, **107** (1985) 111.
- Ezekiel, F.D., and J.L. Shearer, "Pressure-Flow Characteristics of Pneumatic Valves," in Fluid Power Control, J.F. Blackburn, G. Reethof, and J.L. Shearer, eds, MIT Press (1960).
- Myers, A.L., and J.M. Prausnitz, "Thermodynamics of Mixed Gas Adsorption," AIChE L., **11** (1965) 121.

Variables Used in Equations

- A ----- Flow area in bed [square meters]
- b ----- Beaman isotherm coefficient--limiting adsorption amount for nitrogen [kgmol adsorbed/cubic meter]
- β ----- $d(1 - \epsilon)/\epsilon$ [1/second]
- C ----- Total gas concentration in bed [kgmol/cubic meter]
- C^* ----- Total interfacial concentration [kgmol/cubic meter]
- d ----- Mass transfer coefficient [1/second]
- ϵ ----- Bed void fraction
- ΔH ----- Isosteric heat of adsorption [kcal/kgmol]
- j ----- Subscript indicating lump number (also j-1)
- K ----- Beaman isotherm coefficient [kgmol gas phase/kgmol adsorbed phase]
- k ----- Langmuir isotherm coefficient (Henry constant) [1/Pa]
- k_0 ----- Pre-exponential factor for Henry constant [1/Pa]
- L ----- Bed length [meters]
- n ----- Amount adsorbed [kgmol/cubic meter]
- p ----- Partial pressure of a component [Pa]
- P_{BED} ----- Bed pressure [Pa]
- q ----- Amount adsorbed (Langmuir) [kgmol/kg adsorbent]
- θ ----- Fractional coverage
- Q_0 ----- Volumetric flowrate at beginning of bed ($x=0$) [cubic meters/second]
- Q_L ----- Volumetric flowrate at end of bed ($x=L$) [cubic meters/second]
- q_s ----- Saturation amount of adsorption [kgmol/kg adsorbent]
- R ----- Universal gas constant
- ρ ----- Bed density [kg adsorbent/cubic meter]
- T ----- Temperature [K]

u ----- Bed velocity [meters/second]

x ----- distance in bed [meters]

Δx ----- Space step size [meters]

Source Code Listing

BINKLEY.FOR

```

PROGRAM BINKLEY
C   This is the main program of the OBOGS simulator.  This simulator
C   is based on two papers which should be consulted before modifying
C   the code:
C   Beaman, J.J., Healey, A.J., and Werlin, J., "A Dynamic Model of a
C   Molecular Sieve Bed with Nonlinear and Coupled Isotherms,"
C   ASME Journal of Dynamic Systems, Measurement and Control,
C   Vol. 105, Dec. 1983, pp.265-271
C   Beaman, J.J., "A Dynamic Model of a Pressure Swing Oxygen Generation
C   System," ASME Journal of Dynamic Systems, Measurement and
C   Control, Vol. 107, June 1985, pp.111-116

C   This simulator done by Glenn Munkvold, 4/88

PARAMETER (NZ=5,MAXSTPS=5,MAXLUMP=101,NZP1=6)
COMMON/INIT/PATM,TEMPI,O2BLKD
COMMON/GASES/O2MW,RN2MW,RGAS1
COMMON/VLVCON/CONST1,CONST2,CVCP,CRITRAT
COMMON/RDAT2/ DELTAX,DELTAT,SIMTIM,LUMPS,LUMPS1,NTIM
DIMENSION ADSO2 (MAXLUMP,2),ADSTOT (MAXLUMP,2),O2MOL (MAXLUMP,2),
+      U (MAXLUMP,2),ENDS (2,2),PBED (2)

DATA PATM,TEMPI,O2BLKD/.1013,25.,.21/
DATA O2MW,RN2MW,RGAS1/32.,28.,8312.3/
DATA CONST1,CONST2,CVCP,CRITRAT/.1563,.0404,.7143,.528/

C   Variable definitions:
C   ADSO2      Array of values of the adsorbed oxygen
C              concentrations, by (LUMP,bed#), kg-mol/cu m
C   ADSTOT     Array of total adsorbed gas concentrations,
C              by (LUMP,bed#), kg-mol/cu m
C   CONST1,CONST2  Constants in the valve mass flow equations
C              used (see Beaman, 1985)
C   CVCP       Ratio of heat capacities for the valve
C              equations (NOTE: This is CV/CP, which is
C              the inverse of the usual CP/CV)
C   CRITRAT    The "choke flow" pressure ratio
C   CTIME      Running total of time in current cycle (sec)
C   ENDS       Array of the oxygen mole fractions at the
C              ends of the bed; read in for a warm start,
C              normally defaulted.  Stored by (end,bed#),
C              where end=1 is the supply/exhaust end of the
C              bed.
C   MAXLUMP    (PARAMETER) Maximum number of lumps (+1) allowed
C   MAXSTPS    (PARAMETER) Maximum number of steps possible for any one
C              of the time changing variables
C   NZ         (PARAMETER) Maximum number of bed zones allowed
C   NZP1       (PARAMETER) NZ + 1
C   O2BLKD     Default bulk oxygen mole fraction
C   O2MOL      Mole fraction of oxygen in the beds
C   O2MW       Molecular weight of oxygen
C   PATM       Atmospheric pressure, (.1013 MPa)
C   RGAS1      Gas constant (Pa-cu m/kg-mol K)

```

```

C      RN2MW      Molecular weight of nitrogen
C      SIMST      Simulation start time (sec). This zero
C                  except for a warm start.
C      TEMPI      An initial default temperature, 25 C.
C      TFINAL      The actual final time simulated (seconds)
C      U          The array of velocities in each bed (m/s)

C      STARTUP reads all the data files and loads COMMON blocks BDAT,BDAT2,
C      RDAT,RDAT2, and PLENUM. It also opens the output files. Main output
C      is to unit 1, warm start output is to unit 2.
C      CALL STARTUP(SIMST,ADSO2,ADSTOT,CTIME,ENDS,PBED)

C      PROC processes all the data. It prints results at appropriate times
C      and returns the final bed profiles for output to the warm start file.
C      CALL PROC(SIMST,ADSO2,ADSTOT,O2MOL,U,CTIME,ENDS,PBED,
+          TFINAL)

C      SNAPOUT outputs the bed adsorbed gases, bulk oxygen mole fraction,
C      and velocity profiles as well as the scalars CTIME,SIMST(which is
C      the final time on return from PROC). Output is to UOUT.
C      NUOUT=2
C      CALL SNAPOUT(NUOUT,TFINAL,CTIME,ADSO2,ADSTOT,
+          O2MOL,U,PBED,LUMPS1,DELTAX)

      STOP

C2345678901234567890123456789012345678901234567890123456789012
      END

```

```

SUBROUTINE ADSREAD (NUMU, TIME, CTIME, ADSO2, ADSTOT, ENDS,
+ PBED, LUMPS1)
C This routine reads SNAPOUT output to get the required
C initialization information for a warm start.

PARAMETER (NZ=5, MAXSTPS=5, MAXLUMP=101, NZP1=6)

COMMON/PLEN/YO2, YO2M1

DIMENSION ADSO2 (MAXLUMP, 2), ADSTOT (MAXLUMP, 2), O2 (MAXLUMP),
+ ENDS (2, 2), PBED (2)

C PARAMETERS are defined in BINKLEY
C PLEN COMMON variables are defined in STARTUP
C Arguments are defined in PROC

C O2 is O2MOL in the other program units. O2 is only
C used here in order to get ENDS, the O2 mole fractions
C at the ends of the beds.

READ (NUMU, 100) TIME, CTIME, YO2, YO2M1

DO I=1, 2
  READ (NUMU, 110) PBED (I)
  DO J=1, LUMPS1
    READ (NUMU, 120) O2 (J), ADSO2 (J, I), ADSTOT (J, I)
  END DO

  ENDS (1, I) = O2 (1)
  ENDS (2, I) = O2 (LUMPS1)

END DO

RETURN
100 FORMAT (T25, F10.4, T55, F8.4/T23, F6.4, T38, F6.4/)
110 FORMAT (T23, E10.4//)
120 FORMAT (T17, F6.4, T35, E10.4, T46, E10.4)

C2345678901234567890123456789012345678901234567890123456789012
END

```

SUBROUTINE BEDDAT(IWRM,FILNM2,COMENTS)

C This routine reads and modifies the data files
C for the OBOGS simulator. The COMMON blocks BDAT and BDAT2 are
C loaded here.

C Original routine by Glenn Munkvold 4/88

CHARACTER*80 COMENTS(5)
CHARACTER*7 FILNAM,VALC*1,EXT*4,FILNM*11,FILNM2*11,VALC1*1,
+ OLDFIL*11

LOGICAL MODIFY,EXST,LOGDUM,OPN

PARAMETER (NZ=5,MAXSTPS=5,MAXLUMP=101,NZP1=6)
COMMON/BDAT/ BEDD(NZ),ZONEL(NZ),DIFF(NZ),VOIDF(NZ),TZONE(NZ),
+ ZONEKA(NZ),ZONEKB(NZ),ZONEB(NZ),NZONE,FAREA(NZ),BBETA(NZ)
COMMON/BDAT2/ VALVB,VALVO,VALVS,VOLPLEN

DATA MODIFY/.FALSE./,EXST/.FALSE./,LOGDUM/.TRUE./
DATA EXT/'.MSV'/

C Variable definitions follow:
C BBETA Array of BETA values, as defined in
C Beaman's paper $DIFF*(1-VOIDF)/VOIDF$ (1/s)
C BEDD Array of bed diameters, one diameter for
C each zone (cm)
C COMENTS Character array of comments describing
C the data set
C DCOEFB Discharge coefficient of the bypass valve
C DCOEFO Discharge coefficient of the outlet valve
C DCOEFS Discharge coefficient of the supply valve
C Positive discharge coefficients passed
C to the routine are used; negative values
C cause the discharge coefficients to be
C calculated
C DIAMB Diameter of the bypass valve (cm)
C DIAMO Diameter of the outlet valve (cm)
C DIAMS Diameter of the supply valve (cm)
C DIFF Array of the mass transfer coefficients
C for each zone (1/second)
C EXST Logical variable used to tell if a file
C exists or not
C EXT Character variable that contains the
C default extension for the data files for
C the simulator
C FAREA Flow area (total area*void fraction, sq m)
C FILNAM Character variable for the file to be
C modified/created (7 characters, no ext.)
C FILNM Character variable for the complete filename
C (FILNAM + EXT)
C FILNM2 Character variable for the filename that
C the file is to be saved under (may or may
C not be the same as FILNM)
C IWRM Flag for warm starts (=0 for normal start,
C =1 for warm start)
C OLDFIL Name of the data file where bed data were

C originally stored (used in WRMST when
 C WRMST calls PREAD).
 C MAXSTPS Parameter defining the maximum number of
 C time steps that can be input for each
 C time varying parameter
 C MODIFY Logical variable used to tell if a file
 C is to be created or modified
 C NZ Parameter defining the maximum number of
 C zones in the bed
 C NZONE The number of different zones in the bed
 C OPN Logical variable used to tell if a file is
 C open or not
 C PI Yours and my favorite constant (3.14159...)
 C TZONE Array of temperatures for each zone,
 C degrees C
 C VALC, VALC1 Character variables used in reading
 C responses to yes/no questions
 C VALVB The product of the bypass valve area and
 C the bypass discharge coefficient (sq meter)
 C VALVO The product of the outlet valve area and
 C the outlet discharge coefficient (sq meter)
 C VALVS The product of the supply valve area and
 C the supply discharge coefficient (sq meter)
 C VOIDF Array of the void fractions for each zone
 C VOLPLEN The volume of the mixing plenum,
 C cubic meters
 C ZONEB Array of the nitrogen equil constants,
 C one for each zone (kgmol ads/cubic meter)
 C ZONEKA Array of the oxygen equilibrium constants,
 C one for each zone (kgmol gas/kgmol ads)
 C ZONEKB Array of the nitrogen equil constants,
 C one for each zone (kgmol gas/kgmol ads)
 C For a more extended explanation of the
 C equilibrium constants, see Beaman's
 C June, 1985, paper (noting that the B
 C defined in that paper has incorrect units)
 C ZONEL Array of the lengths of each bed zone (cm
 C on input; converted to meters in STARTUP)

PI=3.1415926536

IF(IWRM.EQ.0) THEN

C Ask for the filename
 WRITE(6,100)

DO WHILE(.NOT.EXST)
 WRITE(6,110)
 READ(5,310) FILNAM

C FILNAM is only the seven character name. Call EXTADD to add
 C the proper extension and then ask if the file exists already.
 CALL EXTADD(FILNAM,EXT,FILNM)
 CALL FILCHK(FILNM,EXST)

C If the file exists, ask if the user wants to modify
 C it.

IF(EXST) THEN
 WRITE(6,120) FILNM

```

        READ(5,300) VALC1
        CALL YES(VALC1,MODIFY)

        OPEN(UNIT=8,FILE=FILNM,STATUS='OLD')
        CALL PREAD(8,COMENTS,DIAMB,DIAMS,DIAMO,NZONE,BEDD,
+         ZONEL,DIFF,VOIDF,TZONE,ZONEKA,ZONEKB,ZONEB,VOLPLEN,
+         DCOEFB,DCOEFS,DCOEFO,OLDFIL)

C      If the file does not exist, start over.
        ELSE
        WRITE(6,125) FILNM
        END IF

        END DO

C      If the user wants to modify the data, step through the change
C      routines. Also, set IFLG to zero in order to set interactive
C      mode for the routines

        IF(MODIFY) THEN
        IFLG=0
        DO WHILE(LOGDUM)

C      Comment changes first
        WRITE(6,130)
        CALL CCNG(COMENTS,FILNM,IFLG)

C      Valve parameter changes
        WRITE(6,150)
        CALL VPCNG(DIAMB,DIAMS,DIAMO,DCOEFB,DCOEFS,DCOEFO,
+         VOLPLEN,NZONE,IFLG)

C      Zone specific parameter changes, by zone number IZ
        WRITE(6,160)
        READ(5,*) IZ
        DO WHILE(IZ.GT.0)
        IF(IZ.GT.NZONE) IZ=NZONE
        CALL ZCNG(IZ,BEDD(IZ),ZONEL(IZ),DIFF(IZ),VOIDF(IZ),
+         TZONE(IZ),ZONEKA(IZ),ZONEKB(IZ),ZONEB(IZ),IFLG)
        END DO

C      Ask if changes are complete; repeat if they are not
        WRITE(6,170)
        READ(5,300) VALC1
        CALL YES(VALC1,LOGDUM)
        END DO

C      Ask if the file is to be saved as FILNM. If it is not,
C      get the new filename and save. Otherwise, delete the
C      original file and save the modified one under the same
C      name.
        WRITE(6,180) FILNM
        READ(5,300) VALC1
        CALL YES(VALC1,LOGDUM)

C      If you want to save under another name, get the name, see

```

```

C   if it already exists, and save (if it doesn't exist already).
C   If the file already exists, get a new name.
      IF (.NOT.LOGDUM) THEN
        EXST=.TRUE.
        DO WHILE (EXST)
          WRITE(6,190)
          READ(5,310) FILNAM
          CALL EXTADD(FILNAM,EXT,FILNM2)
          CALL FILCHK(FILNM2,EXST)
          IF (EXST) THEN
            WRITE(6,200)
          ELSE
            WRITE(6,210) FILNM2
          END IF
        END DO
      CLOSE(8)

C   If a data set is being modified, delete the old file so
C   that it can be superceded.
      ELSE
        INQUIRE(8,OPENED=OPN)
        FILNM2=FILNM
        IF (OPN) CLOSE(8,STATUS='DELETE')
      END IF

C   Now open the file for storing the new data
      OPEN(UNIT=3,FILE=FILNM2,STATUS='NEW')

      END IF

      END IF

C   Whether or not the user wanted to change the data, the data are
C   saved. Unit 1 is the main output file for the simulator. Unit 2
C   is the "warm start" file (which contains all the input data and the
C   adsorbed phase concentrations, if the program ends normally).
C   Unit 3 is the new (modified) bed data file, and is only saved if
C   the file has been changed.

      OPEN(UNIT=1,FILE='PSA.DAT',STATUS='NEW')
      OPEN(UNIT=2,FILE='START.WRM',STATUS='NEW')

      LIM=2
      IF (MODIFY) THEN
        LIM=3
      ELSE
        FILNM2=FILNM
      END IF

C   Read the input data if this is a warm start
      IF (IWRM.EQ.1) THEN
        CALL PREAD(8,COMENTS,DIAMB,DIAMS,DIAMO,NZONE,BEDD,
+         ZONEL,DIFF,VOIDF,TZONE,ZONEKA,ZONEKB,ZONEB,VOLPLEN,
+         DCOEFB,DCOEFS,DCOEFO,FILNM2)
      END IF

      DO IFLG=1,LIM
C   Comments

```

```

      CALL CCNG(COMENTS,FILNM2,IFLG)

C   Valve parameters
      CALL VPCNG(DIAMB,DIAMS,DIAMO,DCEFB,DCEFS,DCEFO,VOLPLEN,
+       NZONE,IFLG)

C   Zone specific parameters
      DO I=1,NZONE
        CALL ZCNG(I,BEDD(I),ZONEL(I),DIFF(I),VOIDF(I),
+       TZONE(I),ZONEKA(I),ZONEKB(I),ZONEB(I),IFLG)
      END DO

      END DO

      IF(LIM.EQ.3) CLOSE(3)

C   Calculate the VALV variables, the FAREA and the BBETA vars
      VALVB=(DCEFB*PI*DIAMB**2)/(4.*100**2)
      VALVO=(DCEFO*PI*DIAMO**2)/(4.*100**2)
      VALVS=(DCEFS*PI*DIAMS**2)/(4.*100**2)

      DO I=1,NZONE
        FAREA(I)=VOIDF(I)*PI*BEDD(I)**2/(4.*100.**2)
        BBETA(I)=DIFF(I)*(1.-VOIDF(I))/VOIDF(I)
      END DO

      RETURN

100  FORMAT(///' This is the program that reads and modifies data',
+   ' files for'/T10,'the OBOGS simulator (APR 1988)'/)
110  FORMAT(' Please enter the name of the file to be read',
+   '/T10,'Modified files can be saved under a ',
+   'different name later'/T10,'The filename should be seven or',
+   ' less characters and the extension'/T10,
+   'should be MSV'/T10,'(do NOT include the extension',
+   ' here)')
120  FORMAT(/T10,'Do you want to modify',1X,A11,'?')
125  FORMAT(/T10,'File ',A11,' does not exist!')
130  FORMAT(///' Modify comment lines')
150  FORMAT(///' Modify valve paramters')
160  FORMAT(///' Modify zone specific parameters'/
+   T10,'Enter zone # (zero to exit)')
170  FORMAT(///' Do you want to go back for more changes?')
180  FORMAT(///' Do you want to save the data as ',A11,'?')
190  FORMAT(' Enter the new seven letter(max) filename without',
+   ' any extension')
200  FORMAT(' That file already exists. Please choose another.')
210  FORMAT(' Therefore, the data can be stored on ',A11)
300  FORMAT(A1)
310  FORMAT(A7)

C234567890123456789012345678901234567890123456789012345678901234567890123456789012
      END

```

```

SUBROUTINE BEDP(UL,UZERO,BFLOW,O2IN,PIN,PBED,CAVG,CINTINT,
+ O2MOL,LUMPS1,PSUP,POUT,QZMASS,BCINT)
C This subroutine finds the bed pressures from equation 10 of
C Beaman's 1985 paper. The actual equations used here are
C more general, however.

EXTERNAL FINDP

PARAMETER (NZ=5,MAXSTPS=5,MAXLUMP=101,NZP1=6)
COMMON/BDAT/ BEDD(NZ),ZONEL(NZ),DIFF(NZ),VOIDF(NZ),TZONE(NZ),
+ ZONEKA(NZ),ZONEKB(NZ),ZONEB(NZ),NZONE,FAREA(NZ),BBETA(NZ)
COMMON/GASES/O2MW,RN2MW,RGAS1

COMMON/FPRESS/PBEDI,CONV,TEMPP,RMW(2,2),FACIN,NNBD,QZ(2),
+ QL(2),WBRMOL,PINJ,CCINT(2),CBCINT(2)

DIMENSION CINTINT(2),O2MOL(MAXLUMP,2),QZMASS(2),PIN(2),
+ PBED(2),CAVG(2),UL(2),UZERO(2),BCINT(2),QLMASS(2)

C BDAT COMMON variables are defined in BEDDAT
C GASES COMMON variables are defined in BINKLEY
C PARAMETERS are defined in BINKLEY
C Arguments are defined in PROC

C Variable definitions:
C AVGMWIN Average supply gas molecular weight (based
C on O2IN)
C CCINT This is CINTINT translated to common FPRESS.
C It is an integral calculated in EQUIL.
C Stored by (bed#).
C CBCINT This is BCINT translated to common FPRESS.
C It is an integral calculated in EQUIL.
C Stored by (bed#).
C CONV (PV/T) at STP for Ideal Gas Law conversions
C ((cu m) (MPa)/(kg-mol) (K))
C FACIN Converts (kg/s) to (cu m/s) ambient for
C supply gas
C FINDP Function that is used in FALSEP. It
C returns the residual of equation 10
C from Beaman's 1985 paper.
C ITERS Iteration counter for the loop that tries
C to solve for the bed pressures
C NNBD The bed number being solved
C PBEDI This is the assumed pressure in the other
C bed when bed J is being solved. (MPa)
C PFINAL The final bed pressure from the false
C position solver. (MPa)
C PINJ This is the inlet pressure to bed J. (MPa)
C PJ1,PJ2 Intermediate values of the bed pressure in
C bed J. (MPa)
C POLD This is the last guess at the pressure in
C bed 1. It is used to test when the loop
C has converged (MPa).
C QL The volumetric flowrate (cu m/s)
C at the outlet end of each bed (where X=L).
C Stored by (bed#)
C QLMASS The mass flowrate at the outlet end of each
C bed (kg/s). Stored by (bed#). NOTE that

```

```

C          QZMASS that is returned above is in (grams/s),
C          NOT (kg/s)
C      QZ      The volumetric flowrate (cu m/s) at the
C              inlet of each bed (where X=0).  Stored
C              by (bed#)
C      RESID1,RESID2      Residuals of equation 10 when evaluated by
C                          PJ1 and PJ2 respectively.
C      RMW      Array of molecular weights, stored by
C              (end,bed#)
C      TEMPP    Temperature changes by zone are not yet
C              supported, so a single T can be used for
C              the pressure calculations (degrees K)
C      WBRMOL    BFLOW converted from STDlpm to (kg-mol/sec)

C      First, do conversions and initial calculations
      TEMPP=TZONE(1) + 273.16
      WBRMOL=BFLOW/(1000.*22.4*60.)
      AVGMWIN=(1.-O2IN)*RN2MW + O2IN*O2MW
      CONV=22.4*.1013/273.16
      FACIN=CONV*TEMPP/AVGMWIN
      DO I=1,2
        CCINT(I)=CINTINT(I)
        CBCINT(I)=BCINT(I)
        RMW(1,I)=O2MOL(1,I)*O2MW+(1.-O2MOL(1,I))*RN2MW
        RMW(2,I)=O2MOL(LUMPS1,I)*O2MW+(1.-O2MOL(LUMPS1,I))*RN2MW
      END DO
      ITERS=0
C      Make POLD twice PBED(1) just to assure entry into the loop below
      POLD=PBED(1)*2.

C      Start the loop to find the bed pressures.  First assume PBED(1)
C      is correct.  Using this value, calculate the flows in bed #2 and
C      evaluate equation 10 from the paper (calls to FINDP).  Keep calling
C      FINDP (while varying the bed pressure) until the proper pressure is
C      bracketed (when the product of the residuals is negative).  Once the
C      proper pressure is bracketed, call FALSEP to finish solving for the
C      pressure.  Then, using this value for the pressure in bed #2, calculate
C      the flows in bed #1 and evaluate equation 10 from the paper.  Bracket
C      the solution, call FALSEP to finish, and compare the new bed #1
C      pressure with POLD, the originally assumed pressure.  If they match,
C      the pressures are solved.
      DO WHILE (ABS(PBED(1)-POLD)/PBED(1).GT.1.E-6.AND.
+      ITERS.LE.20)
        POLD=PBED(1)

C      Loop on beds
        DO I=1,2
          J=3-I
C      Load FPRESS for the function FINDP.  FINDP is the only place
C      where FPRESS is used.
          NNBD=J
          PBEDI=PBED(I)
          PINJ=PIN(J)
C      Start finding residuals for bed J.
          PJ1=PBED(J)
          RESID1=FINDP(PJ1)
          PJ2=PJ1*(1. - RESID1)
          DO WHILE (ABS(PJ1-PJ2)/PJ1 .GT. 1.E-6)

```

```

        RESID2=FINDP(PJ2)
        IF (RESID1*RESID2 .LT. 0.) THEN
            CALL FALSEP (FINDP,PJ1,PJ2,PFINAL)
C      Because we are so sure that FALSEP will work once the pressure
C      is bracketed, set PJ1=PJ2 to exit the loop.
            PJ1=PFINAL
            PJ2=PJ1
        ELSE
            PJ1=PJ2
            RESID1=RESID2
            PJ2=PJ1*(1. - RESID1)
            END IF
        END DO

        PBED(J)=PJ1

    END DO

    ITERS=ITERS + 1
    END DO

C      Having found the bed pressures, solve for the other variables
C      of interest.
    DO I=1,2
        CAVG(I)=PBED(I)*1.E6/(RGAS1*TEMPP)
        UZERO(I)=QZ(I)/FAREA(1)
        IF (QZ(I).GT.0.) THEN
            QZMASS(I)=QZ(I)*AVGMWIN*1000.*CAVG(I)
        ELSE
            QZMASS(I)=QZ(I)*RMW(1,I)*1000.*CAVG(I)
        END IF
        UL(I)=QL(I)/FAREA(NZONE)
    END DO

    IF (ITERS.GT.20) WRITE(6,*) ' Failed to converge'

    RETURN

C2345678901234567890123456789012345678901234567890123456789012
END

```

```

SUBROUTINE CCNG(COMENTS,FILNM,IFLG)

C   This subroutine allows for interactive changes in the data
C   comments (IFLG=0) or writes the comments to unit number
C   IFLG (for non-zero IFLG)

LOGICAL LOGDUM
CHARACTER VAL*1,COMENTS(*)*(*),NEWCOM*80,FILNM*(*)
CHARACTER*9 DDATE,DTIME*8

C   Variable definitions follow:
C   COMMENTS           Character array of comments describing
C                       the data set
C   DDATE              Character variable for the date.
C   DTIME              Character variable for the time.
C   IFLG               Flag that indicates whether this call is
C                       interactive (IFLG=0) or a call to print
C                       to unit number IFLG.
C   FILNM              File name where data are stored.
C   LOGDUM             Dummy logical variable used in return
C                       from YES routine, which tests for
C                       affirmative interactive responses
C   NEWCOM             Temporary character variable used to
C                       read in the new comment
C   VAL                Character variable used for reading
C                       the response to yes/no questions

C   Interactive mode first
IF(IFLG.EQ.0) THEN
CALL DATE(DDATE)
CALL TIME(DTIME)
WRITE(6,120) DDATE,DTIME,FILNM
DO I=1,5
WRITE(6,110) COMENTS(I)
END DO
WRITE(6,90)
READ(5,300) VAL
CALL YES(VAL,LOGDUM)

C   If the user wants to change the comments, enter the DO WHILE loop
DO WHILE (LOGDUM)
WRITE(6,100)
DO I=1,5
WRITE(6,110) COMENTS(I)
READ(5,310) NEWCOM

C   The first character of NEWCOM determines what action is taken
IF(NEWCOM(1:1).EQ.'\'') NEWCOM=' '
IF(NEWCOM(1:1).EQ.'+') THEN
IBLANK=INDEX(COMENTS(I),' ')
IBLANK=IBLANK + 1
NEWCOM=COMENTS(I) (: IBLANK) //NEWCOM(2:)
END IF
IF(NEWCOM(1:1).EQ.'@') NEWCOM=COMENTS(I)

COMENTS(I)=NEWCOM

END DO

```



```

C      Print out the comments
      DO I=1,5
        WRITE(6,110) COMENTS(I)
      END DO

C      Check if any more changes are desired
      WRITE(6,130)
      READ(5,300) VAL
      CALL YES(VAL,LOGDUM)

      END DO

C      Non-interactive mode
      ELSE
        CALL DATE(DDATE)
        CALL TIME(DTIME)
        WRITE(IFLG,120) DDATE,DTIME,FILNM
        DO I=1,5
          WRITE(IFLG,110) COMENTS(I)
        END DO

        END IF

      RETURN

90      FORMAT(/' Do you want to modify the data comments?')
100     FORMAT(' Comment changes: After each line of comments',
+ ' you have four choices: '/
+ T5,'1. Simply type a new comment to supercede old comment' /
+ T5,'2. Type \ to erase a comment line' /
+ T5,'3. Type + immediately followed by text to append to ',
+ 'old comment' /
+ T5,'4. Type @ to leave comment unchanged')
110     FORMAT(1X,A72)
120     FORMAT(' OBOGS Simulation',T20,A9,T35,A8,T45,'Data From',
+ T60,A11)
130     FORMAT(' Do you want to modify the data comments further?')
300     FORMAT(A1)
310     FORMAT(A72)

C23456789012345678901234567890123456789012345678901234567890123456789012
      END

```

```

SUBROUTINE EQUIL(ADSO2,ADSTOT,LUMPS1,LUMPS,CINTOT,CINTO2,
+ CINTINT,BETACNT,BCINT,DELTAX)
C EQUIL takes the current adsorbed phase concentrations
C and finds the corresponding interfacial concentrations. It also
C integrates the total interfacial concentration and multiplies
C by BETA. These integrals are used in BEDP and FINDU.

PARAMETER (NZ=5,MAXSTPS=5,MAXLUMP=101,NZP1=6)

DIMENSION ADSO2(MAXLUMP,2),ADSTOT(MAXLUMP,2),CINTO2(MAXLUMP,2),
+ CINTOT(MAXLUMP,2),CINTINT(2),BETACNT(MAXLUMP,2),BCINT(2)

C PARAMETERS are defined in BINKLEY
C Arguments are defined in PROC

C ACCUM1 The accumulation variable for the integration
C of CINTINT
C ACCUM2 The accumulation variable for the integration
C of BCINT
C NUMZ The next zone number, used in space
C integrations.
C XTOT The current x coordinate in the bed (m)
C XZONE The total distance to the end of
C the current zone (m)

C Loop on beds
DO IBED=1,2
  XTOT=0.
  NUMZ=1
  ACCUM1=0.
  ACCUM2=0.
  CINTINT(IBED)=0.
  BCINT(IBED)=0.
  BETACNT(1,IBED)=0.
C Call XGETPAM to get parameter values at position XTOT.
  CALL XGETPAM(XTOT,ZBEDD,ZLEN,ZDIFF,ZVOIDF,ZTEMP,
+ ZKA,ZKB,ZB,NUMZ,BETA,AREA)
  NUMZ=NUMZ + 1
  XZONE=XLEN

C Call ISOTHM to get the interfacial concentrations from the
C adsorbed phase concentrations
  CALL ISOTHM(ZKA,ZKB,ZB,ADSO2(1,IBED),ADSTOT(1,IBED),
+ CINTO2(1,IBED),CINTOT(1,IBED))

C Loop on lumps. Only call XGETPAM when a zone boundary has
C been crossed.
DO L=1,LUMPS
  XTOT=XTOT + DELTAX
  IF(XTOT.GT.XZONE) THEN
    CALL XGETPAM(XTOT,ZBEDD,ZLEN,ZDIFF,ZVOIDF,ZTEMP,
+ ZKA,ZKB,ZB,NUMZ,BETA,AREA)
    CINTINT(IBED)=CINTINT(IBED) + ACCUM1
    BCINT(IBED)=BCINT(IBED) + ACCUM2
    ACCUM1=0.
    ACCUM2=0.
    XZONE=XZONE + ZLEN

```

```

      NUMZ=NUMZ + 1
      END IF

      LP1=L + 1
      CALL ISOTHM(ZKA,ZKB,ZB,ADSO2(LP1,IBED),ADSTOT(LP1,IBED),
+          CINTO2(LP1,IBED),CINTOT(LP1,IBED))

C      Do the integrations. CINTINT is used in the calculation of bed
C      pressures. BCINT is also used in the calculation of bed pressures.
C      BETACNT is used in the calculation of the velocity profile
C      in the bed.
      ACCUM1 = ACCUM1 +
+          DELTAX*BETA*AREA*((CINTOT(L,IBED)+CINTOT(LP1,IBED))/2.)
      ACCUM2 = ACCUM2 + DELTAX*BETA*AREA
      BETACNT(LP1,IBED)=BETA*DELTAX*(CINTOT(L,IBED) +
+          CINTOT(LP1,IBED))/2.

      END DO

      CINTINT(IBED)=CINTINT(IBED) + ACCUM1
      BCINT(IBED)=BCINT(IBED) + ACCUM2

      END DO

      RETURN

C23456789012345678901234567890123456789012345678901234567890123456789012
      END

```

```
SUBROUTINE EXTADD(F1,EXT,F2)
```

```
C   This subroutine adds an extension to a filename.  
C   F1 is the input filename, EXT is the extension, and  
C   F2 is the returned filename.
```

```
CHARACTER F1*(*),EXT*(*),F2*(*)
```

```
C   First, find the end of F1
```

```
IBLANK=INDEX(F1,' ')
```

```
C   Then, add the extension where appropriate
```

```
IF (IBLANK.NE.0) THEN
```

```
    F2=F1(:IBLANK)//EXT
```

```
ELSE
```

```
    F2=F1//EXT
```

```
END IF
```

```
RETURN
```

```
C2345678901234567890123456789012345678901234567890123456789012  
END
```

```

SUBROUTINE FILCHK(F1,RES)

C   This subroutine checks to see if file F1 exists or not.
C   F1 is the filename, RES is a logical variable returned.
C   RES is TRUE if the file exists.

CHARACTER F1*(*)
LOGICAL RES

INQUIRE(FILE=F1,EXIST=RES)

IF(.NOT.RES) WRITE(6,100) F1

RETURN

100  FORMAT('/' The file ',A20/T10,'does not exist')

C2345678901234567890123456789012345678901234567890123456789012
END

```

```

SUBROUTINE FINDFLX(NUZERO, YNUZ, YNUZM1, U, O2MOL, CAVG, CINTO2,
+   UL, UZERO, O2IN, CINTOT)
C   This subroutine calculates the flux of oxygen through the bed

PARAMETER (NZ=5, MAXSTPS=5, MAXLUMP=101, NZP1=6)

COMMON/RDAT2/ DELTAX, DELTAT, SIMTIM, LUMPS, LUMPS1, NTIM

DIMENSION O2MOL(MAXLUMP, 2), U(MAXLUMP, 2), CAVG(2), UL(2), UZERO(2),
+   YNUZ(2), YNUZM1(2), NUZERO(2), CINTO2(MAXLUMP, 2), CINTOT(MAXLUMP, 2)

C   RDAT2 COMMON variables are defined in RUNTBL
C   PARAMETERS are defined in BINKLEY
C   Arguments are defined in PROC

C   Loop on beds
DO IBED=1, 2

    IF ((UZERO(IBED)**2+UL(IBED)**2).EQ.0.) THEN
C   Zero flow solution
        DO I=1, LUMPS1
            O2MOL(I, IBED)=CINTO2(I, IBED)/CINTOT(I, IBED)
        END DO
    ELSE

        IF (NUZERO(IBED).EQ.-1) THEN
C   Integrate forward through the whole bed
            LIMIT = 1
            YN = UZERO(IBED)*O2IN*CAVG(IBED)
            CALL INTFWD(LIMIT, LUMPS1, U, CAVG, CINTO2, O2MOL,
+               YN, IBED, CINTOT)
        ELSE IF (NUZERO(IBED).EQ.-2) THEN
C   Integrate backward through the whole bed
            LIMIT = 1
            YN = UL(IBED)*O2MOL(LUMPS1, (3-IBED))*CAVG(IBED)
            CALL INTBKWD(LUMPS1, LIMIT, U, CAVG, CINTO2, O2MOL,
+               YN, IBED, CINTOT)
        ELSE IF (U(NUZERO(IBED)-1, IBED).LT.0.) THEN
C   Then flux must be out both sides of the bed so integrate from the
C   center out.
            CALL INTFWD(NUZERO(IBED), LUMPS1, U, CAVG, CINTO2, O2MOL,
+               YNUZ(IBED), IBED, CINTOT)
            NST=NUZERO(IBED)-1
            LIMIT=1
            CALL INTBKWD(NST, LIMIT, U, CAVG, CINTO2, O2MOL,
+               YNUZM1(IBED), IBED, CINTOT)
        ELSE
C   Flux must be in from both sides, so integrate from the ends in.
            LIMIT=NUZERO(IBED)-1
            YN=U(1, IBED)*O2IN*CAVG(IBED)
            CALL INTFWD(1, LIMIT, U, CAVG, CINTO2, O2MOL,
+               YN, IBED, CINTOT)
            YN=U(LUMPS1, IBED)*O2MOL(LUMPS1, (3-IBED))*CAVG(IBED)
            CALL INTBKWD(LUMPS1, NUZERO(IBED), U, CAVG, CINTO2, O2MOL,
+               YN, IBED, CINTOT)

        END IF
    END IF
END DO

```

END IF

END DO

RETURN

C2345678901234567890123456789012345678901234567890123456789012
END

```

FUNCTION FINDP(PBED)
C   This function finds the residual of equation 10 in Beaman's 1985
C   paper. FINDQZ finds the volumetric flowrate at the outlet end
C   of the bed; FINDQP finds the volumetric flowrate through the
C   bypass valve. All volumetric calculations are done at pressure
C   FBED.

COMMON/FPRESS/PBEDI, CONV, TEMPP, RMW(2,2), FACIN, NNBD, QZ(2),
+  QL(2), WBRMOL, PINJ, CCINT(2), CBCINT(2)
COMMON/GASES/O2MW, RN2MW, RGAS1

C   GASES COMMON variables are defined in BINKLEY
C   FPRESS COMMON variables are defined in BEDP

C   Variable definitions:
C   PBED           The pressure in bed currently being
C                   sought by FALSEP, (MPa)
C   PBEDI          The pressure assumed in the other bed (MPa)
C   QP             The volumetric flowrate through the bypass
C                   valve (cu m/s)
C   TEMPP          The temperature, in K

C   First, call FINDQZ to find the volume rate at the beginning of the bed.
CALL FINDQZ(PINJ, PBED, CONV, TEMPP, RMW, FACIN, NNBD, QZ(NNBD))

C   Then call FINDQP to find the bypass rate.
CALL FINDQP(PBED, PBEDI, CONV, TEMPP, RMW, NNBD, QP)

C   If QP is negative, than product flow is out of the other bed, so
C   QL=QP. If QP is positive, then product flow is out of this bed
C   (QL=QP + product flow). If QP is zero, then half of the product
C   flow comes from this bed (QL=0.5*(product flow)).
IF(QP.GT.0.) THEN
  QL(NNBD)=QP + WBRMOL*RGAS1*TEMPP/(PBED*1.E6)
ELSE IF(QP.LT.0.) THEN
  QL(NNBD)=QP
ELSE
  QL(NNBD)=WBRMOL*RGAS1*TEMPP/(2.*PBED*1.E6)
END IF

C   Now evaluate equation 10
FACTOR=RGAS1*TEMPP*CCINT(NNBD)
FACTOR2=PBED*1.E6*(QL(NNBD) - QZ(NNBD) + CBCINT(NNBD))
FINDP=1. - FACTOR/FACTOR2

RETURN

C23456789012345678901234567890123456789012345678901234567890123456789012
END

```



```

SUBROUTINE FINDQP (PBED,PBEDJ,CONV,TEMPP,RMW,NNBD,QP)
C   This function returns the volume flowrate through the bypass
C   valve at pressure PBED.  Variables are defined in BEDP.
C   Units are (cu m/sec).

DIMENSION RMW(2,2)

C   VALVE returns W, a mass flowrate, from the pressures on each
C   side of the valve.  NVLV is a valve number (NVLV=3 is the
C   bypass valve).
NVLV=3
CALL VALVE (PBED,PBEDJ,W,NVLV,TEMPP)

IF (W.EQ.0.) THEN
    QP=0.
    RETURN
ELSE
    IF (W.LT.0.) THEN
C   Then PBED<PBEDJ, so use the molecular weight from the other
C   bed in conversions.
        QP=W*CONV*TEMPP/(PBED*RMW(2,3-NNBD))
    ELSE
C   Then PBED>PBEDJ, so use the molecular weight from this
C   bed (NNBD) in conversions.
        QP=W*CONV*TEMPP/(PBED*RMW(2,NNBD))

    END IF
END IF

RETURN

C2345678901234567890123456789012345678901234567890123456789012
END

```

```

SUBROUTINE FINDQZ (PIN,PBED,CONV,TEMPP,RMW,FACIN,NNBD,QZ)
C   This function returns the volume flowrate through the supply/
C   exhaust valves at pressure PBED.  Variables are defined in BEDP.
C   Units are (cu m/sec).

DIMENSION RMW(2,2)

C   VALVE returns W, a mass flowrate, from the pressures on each
C   side of the valve.  NVLV is a valve number (NVLV=1 is the
C   supply valve; 2 is the exhaust valve).
NVLV=1
CALL VALVE (PIN,PBED,W,NVLV,TEMPP)

IF (W.EQ.0.) THEN
  QZ=0.
  RETURN
ELSE
  IF (W.LT.0.) THEN
C   Then PIN<PBED, so this is an exhaust sequence.  Note that this
C   flow is negative, as is desired.  Use the molecular weight from this
C   bed (NNBD) in conversions.
    NVLV=2
    CALL VALVE (PIN,PBED,W,NVLV,TEMPP)
    QZ=W*CONV*TEMPP/(PBED*RMW(1,NNBD))
  ELSE
C   Then PIN>PBED, so this is a supply sequence.  Use the molecular
C   weight from the supply gas in conversions.
    QZ=W*FACIN/PBED

    END IF
  END IF

  RETURN
END
C2345678901234567890123456789012345678901234567890123456789012
END

```

```

      SUBROUTINE FINDU (NUZERO, U, BETACNT, CAVG, UL, UZERO, YNUZ, YNUZM1,
+      CINTO2)
C      This subroutine finds the velocity profile in the beds. It also
C      calculates the appropriate initial fluxes. U is found from the
C      trapezoidal rule.
C       $U(I) = U(I-1) + \text{BETA} * ((\text{CINTOT}(I) + \text{CINTOT}(I-1)) / (2 * \text{CAVG}) - 1) * \text{DELTA}X$ 

      PARAMETER (NZ=5, MAXSTPS=5, MAXLUMP=101, NZP1=6)
      COMMON /RDAT2/ DELTAX, DELTAT, SIMTIM, LUMPS, LUMPS1, NTIM

      DIMENSION U (MAXLUMP, 2), CINTO2 (MAXLUMP, 2), BETACNT (MAXLUMP, 2),
+      CAVG (2), UL (2), UZERO (2), YNUZ (2), YNUZM1 (2), NUZERO (2)

C      RDAT2 COMMON variables are defined in RUNTBL
C      Arguments are defined in PROC

C      Variable definitions:
C      ALAST      Integrations done in this routine are
C                  done zone by zone. At each zone change,
C                  the last value of many parameters are
C                  needed as the lower limits of integration.
C                  ALAST is the last area (sq m)
C      BETACNT       $\text{BETACNT}(I) = \text{BETA} * \text{DELTA}X * (\text{CINTOT}(I) +$ 
C                   $\text{CINTOT}(I-1)) / 2.$ 
C      NUMZ      The current zone number plus one. It is
C                  used in calls to XGETPAM.
C      ULAST      The last velocity (m/s)
C      XLAST      The x coordinate of the beginning of the
C                  zone. (m)
C      XTOT      The current x coordinate in the bed (m)
C      XZONE      The total distance to the end of
C                  the current zone (m)

C      Loop on beds
      DO IBED=1,2

C      The zero flow solution
      IF ((UL (IBED) **2 + UZERO (IBED) **2) .LE. 0.) THEN
        DO I=1, LUMPS1
          U (I, IBED) = 0.
        END DO
        NUZERO (IBED) = 0
        YNUZ (IBED) = 0.
        YNUZM1 (IBED) = 0.
      ELSE

C      Initialization
        XTOT = 0.
        XLAST = 0.
        U (1, IBED) = UZERO (IBED)
        NUZERO (IBED) = 0
        NUMZ = 1

C      Call XGETPAM to get parameter values at position XTOT.
        CALL XGETPAM (XTOT, ZBEDD, ZLEN, ZDIFF, ZVOIDF, ZTEMP,
+      ZKA, ZKB, ZB, NUMZ, BETA, AREA)
        NUMZ = NUMZ + 1
        XZONE = ZLEN
        ALAST = AREA

```

```

C   Set NUZERO for the cases of flow in the positive only (-1) or negative
C   only (-2) directions in the beds.
      IF((UL(IBED)*UZERO(IBED)).GE.0.) THEN
C   Then the velocities at the ends of the beds are of the same sign, and
C   therefore, there is no zero point in the bed.
      IF(UL(IBED).LE.0.) NUZERO(IBED) = -2
C   That is, flow is entirely in the negative X direction
      IF(UZERO(IBED).GE.0.) NUZERO(IBED) = -1
C   That is, flow is entirely in the positive X direction
      END IF

C   Loop on lumps. Only call XGETPAM when a zone boundary has
C   been crossed. This loop calculates the velocity profile.
      DO L=1,LUMPS
        ULAST=U(L,IBED)
        XTOT=XTOT + DELTAX
        IF(XTOT.GT.XZONE) THEN
          XLAST=XLAST + ZLEN
          CALL XGETPAM(XTOT,ZBEDD,ZLEN,ZDIFF,ZVOIDF,ZTEMP,
+           ZKA,ZKB,ZB,NUMZ,BETA,AREA)
          XZONE=XZONE + ZLEN
          NUMZ=NUMZ + 1
          ULAST=U(L,IBED)*ALAST/AREA
          ALAST=AREA
          END IF

          LP1=L + 1
C   Do the integration
          U(LP1,IBED)=ULAST + BETACNT(LP1,IBED)/CAVG(IBED)
+          - BETA*DELTAX

C   A zero velocity point will only exist if NUZERO is > or = 0.
          IF(NUZERO(IBED).GE.0) THEN

            IF((U(LP1,IBED)*U(L,IBED)).EQ.0.) THEN
C   Then U(LP1,IBED) or U(L,IBED) is zero. Only record it the first
C   time.
              IF(NUZERO(IBED).EQ.0) NUZERO(IBED)=LP1
              YNUZ(IBED)=0.

              ELSE IF((U(LP1,IBED)*U(L,IBED)).LT.0.) THEN
C   Then the zero velocity point is between L and LP1. However, fluxes
C   should only be calculated here if the flow is out both ends of the
C   bed. The flux will only be out both sides of the bed if a zero
C   velocity point exists (and one will exist if the above IF statement
C   is true) and if the velocity at the end of the bed is positive.

C   First, assign the zero velocity point
              NUZERO(IBED) = LP1

              IF (UL(IBED).GT.0.) THEN
C   Then the conditions are correct for flow out both sides of the
C   bed. What follows is
C   a hybridization of the integration scheme from Beaman, Healey, and
C   Werlin 1983 paper. The idea is to find the X intercept between L
C   and LP1 where the velocity is zero and then to integrate from this
C   point to L and LP1 to get the initial fluxes at L and LP1.

```

```

C      DZCR is the distance from the zero velocity point to X(LP1).
C      DTC is the distance from the zero point to X(L). Note that DZCR
C      is positive, and that DTC is negative.
          DZCR=DELTA*U(LP1,IBED)/(U(LP1,IBED)-ULAST)
          DTC=DZCR-DELTA
C      Evaluate the integration parameters and calculate the initial
C      flux at the LP1 point (this is YNUZ)
          R1=U(LP1,IBED)/DZCR
          R3=(CINTO2(LP1,IBED)-CINTO2(L,IBED))/DELTA
          R2=CINTO2(L,IBED)-R3*DTC
          A1=BETA*R2*R1/(BETA+R1)
          A2=BETA*R3*R1/(BETA+2.*R1)
          YNUZ(IBED)=A1*DZCR + A2*DZCR*DZCR
C      R1 has to be reevaluated for the flux in the other direction.
C      YNUZM1 is the flux at the L point. Note that for integrations
C      in the backward direction, the definition for DELTA changes
C      (normally, DELTA=X(I) - X(I-1); for backward integration,
C      DELTA=X(I-1) - X(I)). This change in definition is necessary
C      because the node numbers I are monotonically increasing with
C      distance. Because of the change in definition of DELTA, the
C      definition for R1 changes to:
C      R1=(U(I-1) - U(I))/(X(I-1) - X(I)). Note further that R3 and R2
C      are unaffected by these concerns here.
          R1=ULAST/DTC
          A1=BETA*R2*R1/(BETA+R1)
          A2=BETA*R3*R1/(BETA+2.*R1)
          YNUZM1(IBED)=A1*DTC + A2*DTC*DTC

          END IF

          END IF

          END IF
        END DO
      END IF
    END DO
  RETURN

C2345678901234567890123456789012345678901234567890123456789012
END

```

```

SUBROUTINE GETCYC(TCYC,CTIME,PIN,PSUP,POUT)
C   This routine matches the proper pressure to the beds based
C   on the cycle time (TCYC) and the time in the current cycle
C   (CTIME).  Odd half cycles put PSUP on bed 1 ;
C   even half cycles put PSUP on bed 2.  All
C   pressures are MPa; all times are seconds.

DIMENSION PIN(2)

HAFCYC=TCYC/2.

C   The following lines are for initialization only.
IF(PIN(1).NE.POUT.AND.PIN(1).NE.PSUP) THEN
    PIN(1)=PSUP
    PIN(2)=POUT
END IF

IF(CTIME.GE.HAFCYC) THEN
    IF((CTIME-HAFCYC).GE.HAFCYC) THEN
C   Reset CTIME and start an odd half cycle
        CTIME=0.
        PIN(1)=PSUP
        PIN(2)=POUT
    ELSE
C   Set the even half cycle
        PIN(1)=POUT
        PIN(2)=PSUP
    END IF
END IF

RETURN

C23456789012345678901234567890123456789012345678901234567890123456789012
END

```

```

SUBROUTINE INTBKWD(LONE,LLAST,U,CAVG,CINTO2,O2MOL,
+           FLUX1,IBED,CINTOT)
C   This routine integrates the flux backward through the bed. See
C   both of the papers noted in BINKLEY for a description of these
C   equations and the solution method. Note that LONE>LLAST, and that
C   DELTAX must be reversed in sign (using DELXNEG).

PARAMETER (NZ=5,MAXSTPS=5,MAXLUMP=101,NZP1=6)

COMMON/RDAT2/ DELTAX,DELTAT,SIMTIM,LUMPS,LUMPS1,NTIM

DIMENSION O2MOL(MAXLUMP,2),U(MAXLUMP,2),CAVG(2),
+         CINTO2(MAXLUMP,2),CINTOT(MAXLUMP,2)

C   RDAT2 COMMON variables are defined in RUNTBL
C   PARAMETERS are defined in BINKLEY
C   Arguments are defined in PROC, EXCEPT as noted below.

C   Variable definitions:
C   ALAST           The flow area of the zone that has just
C                   been integrated (sq.m.)
C   DELXNEG         The negative of the space step, DELTAX
C   FLUXOLD         The flux at the last space step
C                   ((kg-mol)/(sq m)(sec))
C   FLUX1           The flux at the lower integration limit
C                   ((kg-mol)/(sq m)(sec))
C   IBED            Bed number
C   LLAST           The upper integration limit (lump #)
C   LONE            The lower integration limit (lump #)
C   NUMZ            The zone number to be used in the next
C                   call to XGETPAM. If NUMZ is less than 1,
C                   XGETPAM uses XTOT to get the parameters
C                   and returns the proper NUMZ.
C   ULAST           The velocity at lump J, used in the
C                   calculation of the flux at the (J-1) lump.
C                   ULAST has to be adjusted using areas at
C                   zone boundaries in order to enforce
C                   continuity (m/s).
C   XTOT            The current x coordinate in the bed (m)
C   XZONE           The total distance to the end of the
C                   current zone (m), IN THE DIRECTION OF
C                   INTEGRATION! Note that this definition is
C                   different than in the forward integrating
C                   routines, and corresponds to the total
C                   distance from the beginning of the bed
C                   to the beginning of the current zone.

C   First, find the coordinate of the lump LONE
XTOT=DELTAX*FLOAT(LONE-1)
DELXNEG= - DELTAX

C   Now find out where in the current zone XTOT is. This is necessary
C   because we need to know where to switch to the next zone. Increment
C   through the bed to the end of the current zone to find its
C   coordinate. Find out the place in the zone from this coordinate, the
C   length of the zone (ZLEN) and XTOT.
NUMZ=1
XZONE=0.

```

```

DO WHILE (XZONE.LT.XTOT)
  CALL XGETPAM(XZONE,ZBEDD,ZLEN,ZDIFF,ZVOIDF,ZTEMP,
+    ZKA,ZKB,ZB,NUMZ,BETA,AREA)
  XZONE = XZONE + ZLEN
  NUMZ = NUMZ + 1
END DO
XZONE = XZONE - ZLEN

C   Now call XGETPAM with XTOT to get the proper parameters at XTOT
NUMZ = -1
CALL XGETPAM(XTOT,ZBEDD,ZLEN,ZDIFF,ZVOIDF,ZTEMP,
+    ZKA,ZKB,ZB,NUMZ,BETA,AREA)
NUMZ=NUMZ - 1
ALAST=AREA

C   Note the minus sign here because of the backward integration

C   Initialize variables
FLUXOLD=FLUX1
IF (U(LONE,IBED).NE.0.) THEN
  O2MOL(LONE,IBED)=FLUX1/(U(LONE,IBED)*CAVG(IBED))
ELSE
  O2MOL(LONE,IBED)=CINTO2(LONE,IBED)/CINTOT(LONE,IBED)
END IF

C   Start integration loop. Only call XGETPAM when necessary. Note
C   the loop counter is DECREMENTED.
DO J=LONE,LLAST + 1,-1
  JM1=J - 1
  ULAST=U(J,IBED)
  XTOT=XTOT - DELTAX

C   Note that in the forward integrations, parameters from the left
C   side of the bed are used whenever zone boundaries and nodes are
C   coincident. Therefore, use the .LE. condition here to use the
C   same parameters in the reverse integrations.
  IF (XTOT.LE.XZONE) THEN
    CALL XGETPAM(XTOT,ZBEDD,ZLEN,ZDIFF,ZVOIDF,ZTEMP,
+    ZKA,ZKB,ZB,NUMZ,BETA,AREA)
    XZONE=XZONE - ZLEN
    NUMZ=NUMZ - 1
    ULAST=ULAST*ALAST/AREA
    ALAST=AREA
  END IF

C   See the papers for integration details
  R0=ULAST
  R1=(U(JM1,IBED) - R0)/DELXNEG
  R2=CINTO2(J,IBED)
  R3=(CINTO2(JM1,IBED) - R2)/DELXNEG
  A2=BETA*R3*R1/(BETA + 2.*R1)
  A1=(BETA*(R3*R0 + R2*R1) - 2.*A2*R0)/(BETA + R1)
  A0=R0*(BETA*R2 - A1)/BETA
  IF (R1.NE.0..AND.R0.NE.0.) THEN
    PWR=-BETA/R1
    FLUXNEW=(FLUXOLD - A0)*(1. + R1*DELXNEG/R0)**PWR
    FLUXNEW=FLUXNEW + A0 + A1*DELXNEG + A2*DELXNEG**2
  ELSE
    IF (R0.NE.0.) THEN

```



```

      PWR=-BETA*DELXNEG/R0
      FLUXNEW=(FLUXOLD - A0)*EXP(PWR) + A0 + A1*DELXNEG
    ELSE
      FLUXNEW=A1*DELXNEG + A2*DELTAX*DELTAX
    END IF

```

```

  END IF

```

```

  IF (U(JM1,IBED).NE.0.) THEN
    O2MOL(JM1,IBED)=FLUXNEW/(CAVG(IBED)*U(JM1,IBED))
  ELSE
    O2MOL(JM1,IBED)=CINTO2(JM1,IBED)/CINTOT(JM1,IBED)
  END IF
  FLUXOLD=FLUXNEW
END DO

```

```

RETURN

```

```

C2345678901234567890123456789012345678901234567890123456789012
END

```

```

SUBROUTINE INTFWD(LONE,LLAST,U,CAVG,CINTO2,O2MOL,
+             FLUX1,IBED,CINTOT)
C   This routine integrates the flux forward through the bed.  See
C   both of the papers noted in BINKLEY for a description of these
C   equations and the solution method.  Note that LLAST>LONE.

PARAMETER (NZ=5,MAXSTPS=5,MAXLUMP=101,NZP1=6)

COMMON/RDAT2/ DELTAX,DELTAT,SIMTIM,LUMPS,LUMPS1,NTIM

DIMENSION O2MOL(MAXLUMP,2),U(MAXLUMP,2),CAVG(2),
+         CINTO2(MAXLUMP,2),CINTOT(MAXLUMP,2)

C   RDAT2 COMMON variables are defined in RUNTBL
C   PARAMETERS are defined in BINKLEY
C   Arguments are defined in PROC, EXCEPT as noted below.

C   Variable definitions:
C   ALAST           The flow area of the last zone integrated
C                   (sq.m.)
C   FLUXOLD          The flux at the last space step
C                   ((kg-mol)/(sq m)(sec))
C   FLUX1           The flux at the lower integration limit
C                   ((kg-mol)/(sq m)(sec))
C   IBED            Bed number
C   LLAST           The upper integration limit (lump #)
C   LONE            The lower integration limit (lump #)
C   NUMZ            The zone number to be used in the next
C                   call to XGETPAM.  If NUMZ is less than 1,
C                   XGETPAM uses XTOT to get the parameters
C                   and returns the proper NUMZ.
C   ULAST           The velocity at lump J, used in the
C                   calculation of the flux at the (J+1) lump.
C                   ULAST has to be adjusted using areas at
C                   zone boundaries in order to enforce
C                   continuity (m/s).
C   XTOT            The current x coordinate in the bed (m)
C   XZONE           The total distance to the end of
C                   the current zone (m)

C   First, find the coordinate of the lump LONE
XTOT=DELTAX*FLOAT(LONE-1)

C   Now find out where in the current zone XTOT is.  This is necessary
C   because we need to know where to switch to the next zone.  Increment
C   through the bed to the end of the current zone to find its
C   coordinate.  Find out the place in the zone from this coordinate
C   and XTOT.
NUMZ=1
XZONE=0.
DO WHILE(XZONE.LE.XTOT)
  CALL XGETPAM(XZONE,ZBEDD,ZLEN,ZDIFF,ZVOIDF,ZTEMP,
+             ZKA,ZKB,ZB,NUMZ,BETA,AREA)
  XZONE = XZONE + ZLEN
  NUMZ = NUMZ + 1
END DO

NUMZ=-1

```

```

C      Call XGETPAM to get parameter values at position XTOT.
      CALL XGETPAM(XTOT,ZBEDD,ZLEN,ZDIFF,ZVOIDF,ZTEMP,
+       ZKA,ZKB,ZP,NUMZ,BETA,AREA)
      NUMZ=NUMZ + 1
      ALAST=AREA

C      Initialize variables
      FLUXOLD=FLUX1
      IF (U(LONE,IBED).NE.0.) THEN
        O2MOL(LONE,IBED)=FLUX1/(U(LONE,IBED)*CAVG(IBED))
      ELSE
        O2MOL(LONE,IBED)=CINTO2(LONE,IBED)/CINTOT(LONE,IBED)
      END IF

C      Start integration loop. Only call XGETPAM when necessary
      DO J=LONE,LLAST - 1
        JP1=J + 1
        ULAST=U(J,IBED)
        XTOT=XTOT + DELTAX

        IF (XTOT.GT.XZONE) THEN
          CALL XGETPAM(XTOT,ZBEDD,ZLEN,ZDIFF,ZVOIDF,ZTEMP,
+           ZKA,ZKB,ZB,NUMZ,BETA,AREA)
          XZONE=XZONE + ZLEN
          NUMZ=NUMZ + 1
          ULAST=ULAST*ALAST/AREA
          ALAST=AREA
        END IF

C      See the papers for integration details
        R0=ULAST
        R1=(U(JP1,IBED) - R0)/DELTAX
        R2=CINTO2(J,IBED)
        R3=(CINTO2(JP1,IBED) - R2)/DELTAX
        A2=BETA*R3*R1/(BETA + 2.*R1)
        A1=(BETA*(R3*R0 + R2*R1) - 2.*A2*R0)/(BETA + R1)
        A0=R0*(BETA*R2 - A1)/BETA
        IF (R1.NE.0..AND.R0.NE.0.) THEN
          PWR=-BETA/R1
          FLUXNEW=(FLUXOLD - A0)*(1. + R1*DELTAX/R0)**PWR
          FLUXNEW=FLUXNEW + A0 + A1*DELTAX + A2*DELTAX**2
        ELSE
          IF (R0.NE.0.) THEN
            PWR=-BETA*DELTAX/R0
            FLUXNEW=(FLUXOLD - A0)*EXP(PWR) + A0 + A1*DELTAX
C2345678901234567890123456789012345678901234567890123456789012
          ELSE
            FLUXNEW=A1*DELTAX + A2*DELTAX*DELTAX
          END IF
        END IF

        IF (U(JP1,IBED).NE.0.) THEN
          O2MOL(JP1,IBED)=FLUXNEW/(CAVG(IBED)*U(JP1,IBED))
        ELSE
          O2MOL(JP1,IBED)=CINTO2(JP1,IBED)/CINTOT(JP1,IBED)
        END IF
      END DO

```

FLUXOLD=FLUXNEW
END DO

RETURN

C2345678901234567890123456789012345678901234567890123456789012
END

```

SUBROUTINE INTTIM(ADSO2,ADSTOT,CAVG,O2MOL,CINTO2,
+ CINTOT)
C This routine integrates the adsorbed phase gas concentrations
C in time. See the 1985 paper (as noted in BINKLEY) for details
C of the method (a simple Euler method).

PARAMETER (NZ=5,MAXSTPS=5,MAXLUMP=101,NZP1=6)

COMMON/RDAT2/ DELTAX,DELTAT,SIMTIM,LUMPS,LUMPS1,NTIM

DIMENSION ADSO2(MAXLUMP,2),ADSTOT(MAXLUMP,2),O2MOL(MAXLUMP,2),
+ CINTO2(MAXLUMP,2),CINTOT(MAXLUMP,2),CAVG(2)

C RDAT2 COMMON variables are defined in RUNTBL
C PARAMETERS are defined in BINKLEY
C Arguments are defined in PROC

C Variable definitions:
C NUMZ The zone number to be used in the next
C call to XGETPAM. If NUMZ is less than 1,
C XGETPAM uses XTOT to get the parameters
C and returns the proper NUMZ.
C XTOT The current x coordinate in the bed (m)
C XZONE The distance in the current zone (m)

C Loop on beds
DO I=1,2

    NUMZ=1
    XTOT=0.
C Call XGETPAM to get parameter values at position XTOT.
    CALL XGETPAM(XTOT,ZBEDD,ZLEN,ZDIFF,ZVOIDF,ZTEMP,
+ ZKA,ZKB,ZB,NUMZ,BETA,AREA)
    NUMZ=NUMZ + 1
    XZONE=ZLEN

C Loop on lumps; only call XGETPAM when necessary
    DO J=1,LUMPS1
        IF (XTOT.GT.XZONE) THEN
            CALL XGETPAM(XTOT,ZBEDD,ZLEN,ZDIFF,ZVOIDF,ZTEMP,
+ ZKA,ZKB,ZB,NUMZ,BETA,AREA)
            XZONE=XZONE + ZLEN
            NUMZ=NUMZ + 1
            END IF

            ADSTOT(J,I)=ADSTOT(J,I)-ZDIFF*
+ (CINTOT(J,I)-CAVG(I))*DELTAT

            ADSO2(J,I)=ADSO2(J,I)-ZDIFF*
+ (CINTO2(J,I)-CAVG(I))*O2MOL(J,I))*DELTAT

        END DO

        XTOT=XTOT + DELTAX

    END DO

RETURN

```

C2345678901234567890123456789012345678901234567890123456789012
END

```

SUBROUTINE KTCORR(TEMP,ZKA,ZKB)
C   This routine returns the isotherm coefficients (ZKA,ZKB) as
C   functions of temperature (TEMP) IF ZKA and/or ZKB are
C   negative when this routine is called.

C   TEMP                (degrees C)
C   ZKA                  (kg-mol O2 gas/kg-mol O2 ads)
C   ZKB                  (kg-mol N2 gas/kg-mol N2 ads)

C   These data are from Dave Walshak. See Beaman for details.
C   This work was done for a Master's Thesis that Walshak has
C   yet to finish (as of 4/87). Where all the "significant"
C   figures come from is not clear.

C   Also note that Walshak uses ZB (B) as
C   3.0591250816 (kg-mol N2 ads/cu m)

T=TEMP
IF(ZKA.LT.0.) THEN
  ZKA=.1423275+.00183744*T+.0000072443*T*T
  END IF

IF(ZKB.LT.0.) THEN
  ZKB=.03654573+.00095775*T+.00001321*T*T
  END IF

RETURN

C2345678901234567890123456789012345678901234567890123456789012
END

```

```

SUBROUTINE PLENUM(BFLOW,VOLPLEN,CAVG,O2MOL,LUMPS1,
+   DELTAT,FO2MOL)
C   This routine models the plenum as a two stage (second order)
C   mixer. Each stage is half the total plenum volume (VOLPLEN).
C   The time constant for each stage is the total molar plenum
C   concentration (taken to be CAVG if we assume no pressure
C   drop across the bed) times the stage volume over the molar
C   product rate. FO2MOL is the final O2 mole fraction output.

PARAMETER (NZ=5,MAXSTPS=5,MAXLUMP=101,NZP1=6)
DIMENSION CAVG(2),O2MOL(MAXLUMP,2)
COMMON/PLEN/YO2,YO2M1

C   The variables are defined in BINKLEY, BEDDAT, and RUNTBL
C   PLEN COMMON variables are defined in STARTUP
C   PARAMETERS are defined in BINKLEY

WBRMOL=BFLOW/(22.4*60.*1000.)

C   If the bed pressures are equal, average the compositions
IF(CAVG(1).EQ.CAVG(2)) THEN
    TAU=CAVG(1)
    XO2=(O2MOL(LUMPS1,1)+O2MOL(LUMPS1,2))/2.
ELSE
    IF(CAVG(1).GT.CAVG(2)) THEN
        TAU=CAVG(1)
        XO2=O2MOL(LUMPS1,1)
    ELSE
        TAU=CAVG(2)
        XO2=O2MOL(LUMPS1,2)
    END IF
END IF

TAU=TAU*VOLPLEN/(2.*WBRMOL)

C   Now, integrate the mixer in time. XINT is the intermediate
C   composition.
XINT=YO2M1+(1./TAU)*(XO2-YO2M1)*DELTAT
FO2MOL=YO2+(1./TAU)*(YO2M1-YO2)*DELTAT

YO2=FO2MOL
YO2M1=XINT

RETURN

C234567890123456789012345678901234567890123456789012345678901234567890123456789012
END

```



```

SUBROUTINE PREAD(NUMU,COMENTS,DIAMB,DIAMS,DIAMO,NZONE,BEDD,
+ ZONEL,DIFF,VOIDF,TZONE,ZONEKA,ZONEKB,ZONEB,VOLPLEN,
+ DCOEFB,DCOEFS,DCOEFO,FILNM)

```

C This subroutine reads in all the parameters, from unit NUMU.

```

CHARACTER COMENTS(*)*(*),FILNM*(*)
DIMENSION BEDD(*),ZONEL(*),DIFF(*),VOIDF(*),TZONE(*),
+ ZONEKA(*),ZONEKB(*),ZONEB(*)

```

C Variable definitions follow:

C	BEDD	Array of bed diameters, one diameter for
C		each zone (cm)
C	COMENTS	Character array of comments describing
C		the data set
C	DCOEFB	Discharge coefficient of the bypass valve
C	DCOEFO	Discharge coefficient of the outlet valve
C	DCOEFS	Discharge coefficient of the supply valve
C		Positive discharge coefficients passed
C		to the routine are used; negative values
C		cause the discharge coefficients to be
C		calculated
C	DIAMB	Diameter of the bypass valve (cm)
C	DIAMO	Diameter of the outlet valve (cm)
C	DIAMS	Diameter of the supply valve (cm)
C	DIFF	Array of the mass transfer coefficients
C		for each zone (1/second)
C	NZONE	The number of different zones in the bed
C	TZONE	Array of temperatures for each zone,
C		degrees C
C	VOIDF	Array of the void fractions for each zone
C	VOLPLEN	The volume of the mixing plenum,
C		cubic meters
C	ZONEB	Array of the nitrogen equil constants,
C		one for each zone (kgmol ads/cubic meter)
C	ZONEKA	Array of the oxygen equilibrium constants,
C		one for each zone (kgmol gas/kgmol ads)
C	ZONEKB	Array of the nitrogen equil constants,
C		one for each zone (kgmol gas/kgmol ads)
C		For a more extended explanation of the
C		equilibrium constants, see Beaman's
C		June, 1985, paper (noting that the B
C		defined in that paper has incorrect units)
C	ZONEL	Array of the lengths of each bed zone (cm)

REWIND NUMU

C First, read the general system parameters

```

READ(NUMU,100) FILNM,(COMENTS(I),I=1,5),
+ DIAMB,DCOEFB,DIAMS,DCOEFS,DIAMO,DCOEFO,VOLPLEN,NZONE

```

C Then, read zone specific parameters

```

DO J=1,NZONE
  READ(NUMU,110) I
  READ(NUMU,120) BEDD(I),ZONEL(I),DIFF(I),VOIDF(I),TZONE(I),
+ ZONEKA(I),ZONEKB(I),ZONEB(I)
END DO

```

RETURN

```
100  FORMAT (T60,A11/5(1X,A72/))//T17,F10.4,T36,F8.5/T17,F10.4,  
+   T36,F8.5/T17,F10.4,T36,F8.5/T19,E10.3/T24,I2)  
110  FORMAT (/T11,I2)  
120  FORMAT (T17,F10.4/T16,F10.4/T26,F8.2/T18,F6.4/T16,F6.2/  
+   T7,F10.4/T7,F10.4/T6,F8.3/)
```

C2345678901234567890123456789012345678901234567890123456789012
END

```

SUBROUTINE PROC(SIMST,ADSO2,ADSTOT,O2MOL,U,CTIME,ENDS,PBED,
+             TFINAL)
C   This is the main processing routine

PARAMETER (NZ=5,MAXSTPS=5,MAXLUMP=101,NZP1=6)

COMMON/INIT/PATM,TEMPI,O2BLKD
COMMON/GASES/O2MW,RN2MW,RGAS1
COMMON/RDAT2/ DELTAX,DELTAT,SIMTIM,LUMPS,LUMPS1,NTIM
COMMON/PLEN/YO2,YO2M1
COMMON/BDAT2/ VALVB,VALVO,VALVS,VOLPLEN

DIMENSION ADSO2(MAXLUMP,2),ADSTOT(MAXLUMP,2),O2MOL(MAXLUMP,2),
+         U(MAXLUMP,2),ENDS(2,2)
DIMENSION CINTO2(MAXLUMP,2),CINTOT(MAXLUMP,2),
+         CINTINT(2),BCINT(2),BETACNT(MAXLUMP,2),QZMASS(2),PIN(2),
+         PBED(2),CAVG(2),UL(2),UZERO(2),YNUZ(2),YNUZM1(2),NUZERO(2)

C   PLEN COMMON variables are defined in STARTUP
C   RDAT2 COMMON variables are defined in RUNTBL
C   BDAT2 COMMON variables are defined in BEDDAT
C   GASES COMMON variables are defined in BINKLEY
C   INIT COMMON variables are defined in BINKLEY
C   PARAMETERS are defined in BINKLEY
C   Arguments are defined in BINKLEY

C   Variable definitions:
C   BETACNT              BETACNT(L+1)=BETA*DELTAX*(CINTOT(L+1) +
C                       CINTOT(L))/2. (kg-mol m/s cu m)
C                       Stored by (LUMP,bed#)
C   BCINT                This is the integral of BETA*AREA over the
C                       length of the bed (cu m/s). Stored by (bed#)
C   CAVG                 The total average bulk concentration
C                       (kg-mol/cu m)
C   CINTINT              The integral of the total interfacial
C                       concentration over the length of the
C                       bed (kg-mol m/cu m). Stored by (bed#)
C   CINTOT               The total interfacial concentration in bed#
C                       at LUMP, (kg-mol/cu m), stored by (LUMP,bed#)
C   CINTO2               The oxygen interfacial concentration in bed#
C                       at LUMP, (kg-mol/cu m), stored by (LUMP,bed#)
C   FO2MOL               The final oxygen mole fraction out of the
C                       plenum
C   NUZERO               The LUMP where the velocity in the bed
C                       changes direction
C   O2IN                 The inlet oxygen mole fraction
C   PBED                 The bed pressure in bed#, stored by
C                       (bed#), in MPa
C   PCYC                 The print cycle time (seconds)
C   PIN                  The inlet pressure to bed#, stored by
C                       (bed#), in MPa
C   PTIME                The time into the current cycle (seconds)
C   QZMASS               The total mass flow into bed#, stored by
C                       (bed#), in grams/sec
C   TMULT                Multiplier for writing a placation
C                       message to the screen
C   TOTTIM               The total time for simulation (sec)
C   UL                   The linear velocity at the product end of

```

```

C          bed#, stored by (bed#), in meter/sec
C      UZERO      The linear velocity at the supply end of
C          bed#, stored by (bed#), in meter/sec
C      YNUZ      The oxygen flux at lump NUZERO
C          (kg-mol/(sq m) (sec)) stored by (bed#)
C      YNUZM1     The oxygen flux at lump (NUZERO - 1)
C          (kg-mol/(sq m) (sec)) stored by (bed#)

C      Initialize variables
      DO I=1,2
        O2MOL(1,I)=ENDS(1,I)
        O2MOL(LUMPS1,I)=ENDS(2,I)
      END DO

      TIME=SIMST
      TOTTIM=SIMTIM-SIMST
      TMULT=1.
      PTIME=0.
      PCYC=TOTTIM/FLOAT(NTIM)

C      Write table headings
      WRITE(1,100)

C      Start the simulation time loop. TGETPAM gets the current value of
C      time varying parameters. GETCYC gets the current inlet bed
C      pressures. EQUIL takes the current adsorbed phase concentrations
C      and finds the corresponding interfacial concentrations. It also
C      integrates the total interfacial concentration and multiplies
C      by BETA. BEDP finds the pressure in the beds by solving a system
C      of two coupled non-linear equations. It also finds the inlet mass
C      rate to the beds. FINDU integrates to find the velocity profile
C      in the beds. Initial fluxes as well as the point of zero velocity
C      (if there is one) are returned for FINDFLX. FINDFLX integrates in
C      the direction of flow to get the oxygen flux. The mole fractions
C      (O2MOL) are returned. PLENUM models the plenum using a second
C      order transfer function relating inlet and outlet mole fractions.
      DO WHILE (TIME.LT.SIMTIM)
        CALL TGETPAM(BFLOW,O2IN,PSUP,POUT,TCYC,SNAP,TIME)
        CALL GETCYC(TCYC,CTIME,PIN,PSUP,POUT)
        CALL EQUIL(ADSO2,ADSTOT,LUMPS1,LUMPS,CINTOT,CINTO2,
+          CINTINT,BETACNT,BCINT,DELTA)
        CALL BEDP(UL,UZERO,BFLOW,O2IN,PIN,PBED,CAVG,CINTINT,
+          O2MOL,LUMPS1,PSUP,POUT,QZMASS,BCINT)
        CALL FINDU(NUZERO,U,BETACNT,CAVG,UL,UZERO,YNUZ,YNUZM1,
+          CINTO2)
        CALL FINDFLX(NUZERO,YNUZ,YNUZM1,U,O2MOL,CAVG,CINTO2,
+          UL,UZERO,O2IN,CINTOT)
        CALL PLENUM(BFLOW,VOLPLEN,CAVG,O2MOL,LUMPS1,
+          DELTAT,FO2MOL)

C      Calculate oxygen mass rate out (gram/sec) and total mass rate
C      in (gram/sec). 22.4 converts STDlpm to gm-mol; 60 converts
C      minutes to seconds.
        O2MASS=FO2MOL*BFLOW*O2MW/(22.4*60.)
        QZZ=0.
        DO I=1,2

```

```

        IF (QZMASS(I).GT.0) THEN
            QZZ=QZZ+QZMASS(I)
        END IF
    END DO

C    Write output at output intervals
    IF (PTIME.GT.PCYC) THEN
        PTIME=0.
        WRITE(1,110) TIME,FO2MOL,O2MASS,QZZ
    END IF

C    Check to see if it is time for a "snapshot".  Output to UOUT.
    IF (SNAP.GT.3) THEN
        NUOUT=INT(SNAP)
        CALL SNAPOUT(NUOUT,TIME,CTIME,ADS02,ADSTOT,
+           O2MOL,U,PBED,LUMPS1,DELTAX)
    END IF

C    Call INTTIM to integrate the adsorbed gas concentrations in time
    CALL INTTIM(ADS02,ADSTOT,CAVG,O2MOL,CINT02,
+           CINTOT)

C    Update TIME and CTIME and write the placation message
    TIME=TIME + DELTAT
    CTIME=CTIME + DELTAT
    PTIME=PTIME + DELTAT
    IF (TIME.GT.(SIMST+TMULT*TOTTIM/20.)) THEN
        TMULT=TMULT + 1.
        WRITE(6,*) ' Please wait; TIME = ',TIME
    END IF

    END DO
    TFINAL = TIME - DELTAT

    RETURN

100  FORMAT(T5,'OBOGS Output'/3X,'Time',T18,'O2 Mole',T32,
+   'O2 Mass',T44,'Total Mass'/2X,'Seconds',T18,'Fraction',
+   T32,'Out g/s',T46,'In g/s')
110  FORMAT(1X,F10.4,T16,F10.6,T30,F10.4,T44,F10.4)

C23456789012345678901234567890123456789012345678901234567890123456789012
END

```

```
SUBROUTINE RUNTBL(IWRM,FILNM2,COMENTS)
```

```
C This routine reads and modifies the files that control the
C "run time" parameters of a simulation. Specifically, this
C routine loads the COMMON blocks RDAT and RDAT2 (except LUMPS)
C and outputs data.
```

```
PARAMETER (NZ=5,MAXSTPS=5,MAXLUMP=101,NZP1=6)
COMMON/RDAT/ RBFLOW(MAXSTPS,2),RO2IN(MAXSTPS,2),
+ RPSUP(MAXSTPS,2),RPOUT(MAXSTPS,2),RTCYC(MAXSTPS,2),
+ SNAPTIM(MAXSTPS,2)
COMMON/RDAT2/ DELTAX,DELTAT,SIMTIM,LUMPS,LUMPS1,NTIM
```

```
CHARACTER*80 COMENTS(5)
CHARACTER*7 FILNAM,VALC*1,EXT*4,FILNM*11,FILNM2*11,VALC1*1
CHARACTER*2 NUMS(10),OLDFIL*11
DIMENSION PARAMS(10,MAXSTPS,2)
```

```
LOGICAL MODIFY,EXST,LOGDUM,OPN
```

```
DATA MODIFY/.FALSE./,EXST/.FALSE./,LOGDUM/.TRUE./
DATA EXT/'.TBL'/
DATA NUMS/'1.','2.','3.','4.','5.','6.','7.','8.','9.',
+ '10'/
```

```
C Variable definitions follow:
```

```
C COMENTS           Character array of comments describing
C                   the data set
C DELTAT            The time step (sec)
C DELTAX            The lump size. The user inputs this in
C                   centimeters. However, DELTAX is adjusted
C                   slightly in STARTUP in order to get an
C                   integer number of LUMPS and then changed
C                   to meters.
C EXST              Logical variable used to tell if a file
C                   exists or not
C EXT              Character variable that contains the
C                   default extension for the data files for
C                   the simulator
C FILNAM            Character variable for the file to be
C                   modified/created (7 characters, no ext.)
C FILNM            Character variable for the complete filename
C                   (FILNAM + EXT)
C FILNM2           Character variable for the filename that
C                   the file is to be saved under (may or may
C                   not be the same as FILNM)
C IWRM             Flag for warm starts (=0 for normal start,
C                   =1 for warm start)
C LUMPS            The number of space lumps that results
C                   from DELTAX and the total length of the
C                   bed (calculated in STARTUP)
C LUMPS1           LUMPS + 1, used for array dimensions and
C                   loop counters
C MAXSTPS          Parameter defining the maximum number of
C                   time steps that can be input for each
C                   time varying parameter
C MODIFY           Logical variable used to tell if a file
C                   is to be created or modified
```

```

C      NTIM      Number of output points (in time)
C      NUMS      Character array of numbers for interactive
C                output labeling
C      NZ        Parameter defining the maximum number of
C                zones in the bed
C      OPN        Logical variable used to tell if a file is
C                open or not
C      PARAMS     Real array of parameter values for use in
C                DO loops
C      RBFLOW     Breathing (output) flowrate,
C                STD lpm
C      RO2IN      Inlet oxygen concentration, mole fraction
C      RPOUT      Outlet pressure, MPa, absolute
C      RPSUP      Supply pressure, MPa, absolute
C      RTCYC      Cycle time for beds, seconds
C      SIMTIM     Simulation stop time, seconds
C      SNAPTIM     Array that stores the output units and
C                times for snapshots (times are in (1,2),
C                seconds)
C      VALC,VALC1 Character variables used in reading
C                responses to yes/no questions
C      The "R" variables above are all 2-D arrays (MAXSTPS x 2). The
C      first row contains the parameter values; the second contains the
C      start time for the corresponding parameter value.

```

```

C      Redefine the variables for the DO loops

```

```

      NN=MAXSTPS
      PARAMS(1,1,1)=DELTAX
      PARAMS(2,1,1)=DELTAT
      PARAMS(3,1,1)=SIMTIM
      PARAMS(4,1,1)=FLOAT(NTIM)
      DO J=1,NN
        DO K=1,2
          PARAMS(5,J,K)=RBFLOW(J,K)
          PARAMS(6,J,K)=RO2IN(J,K)
          PARAMS(7,J,K)=RPSUP(J,K)
          PARAMS(8,J,K)=RPOUT(J,K)
          PARAMS(9,J,K)=RTCYC(J,K)
          PARAMS(10,J,K)=SNAPTIM(J,K)
        END DO
      END DO

```

```

C      IF(IWRM.EQ.0) THEN
C      Ask for the filename
      WRITE(6,100)

```

```

      DO WHILE(.NOT.EXST)
        WRITE(6,110)
        READ(5,310) FILNAM

```

```

C      FILNAM is only the seven character name. Call EXTADD to add
C      the proper extension and then ask if the file exists already.
      CALL EXTADD(FILNAM,EXT,FILNM)
      CALL FILCHK(FILNM,EXST)

```

```

C      If the file exists, ask if the user wants to modify
C      it.

```

```

      IF(EXST) THEN

```

```

WRITE(6,120) FILNM
READ(5,300) VALC1
CALL YES(VALC1,MODIFY)

OPEN(UNIT=8,FILE=FILNM,STATUS='OLD')
NUMU=8
CALL TREAD(NUMU,FILNM,COMENTS,PARAMS)

C      If the file does not exist, start over.
      ELSE
        WRITE(6,125) FILNM
      END IF

      END DO

C      If the user wants to modify the data, step through the change
C      routines. Also, set IFLG to zero in order to set interactive
C      mode for the routines

      IF(MODIFY) THEN
        IFLG=0
        WRITE(6,220)
        WRITE(6,230)
        CALL CCNG(COMENTS,FILNM,IFLG)

C      Write all the parameters to the screen
      DO WHILE(LOGDUM)
        WRITE(6,240) NUMS(1),PARAMS(1,1,1),NUMS(2),
+          PARAMS(2,1,1),NUMS(3),PARAMS(3,1,1),NUMS(4),
+          PARAMS(4,1,1),NUMS(5),PARAMS(5,1,1),NUMS(6),
+          PARAMS(6,1,1),NUMS(7),PARAMS(7,1,1),NUMS(8),
+          PARAMS(8,1,1),NUMS(9),PARAMS(9,1,1),NUMS(10),
+          PARAMS(10,1,2)
C      Note that the indices on SNAPTIM are NOT in error. The time of the
C      first snapshot is desired here.

C      Get the number that corresponds to the parameter to be changed.
        WRITE(6,250)
        READ(5,*) INVAL

C      INVAL=0 exits; otherwise, call TPARAM to change the parameter
        IF(INVAL.LE.0) THEN
          LOGDUM=.FALSE.
        ELSE IF(INVAL.LE.10) THEN
          CALL TPARAM(PARAMS,NN,INVAL,IFLG)
        END IF

      END DO

C      Ask if the file is to be saved as FILNM. If it is not,
C      get the new filename and save. Otherwise, delete the
C      original file and save the modified one under the same
C      name.
        WRITE(6,180) FILNM
        READ(5,300) VALC1
        CALL YES(VALC1,LOGDUM)

```



```

C   If you want to save under another name, get the name, see
C   if it already exists, and save (if it doesn't exist already).
C   If the file already exists, get a new name.
      IF(.NOT.LOGDUM) THEN
        EXST=.TRUE.
        DO WHILE(EXST)
          WRITE(6,190)
          READ(5,310) FILNAM
          CALL EXTADD(FILNAM,EXT,FILNM2)
          CALL FILCHK(FILNM2,EXST)
          IF(EXST) THEN
            WRITE(6,200)
          ELSE
            WRITE(6,210) FILNM2
          END IF
        END DO
      CLOSE(8)

C   If a data set is being modified, delete the old file so
C   that it can be superceded.
      ELSE
        INQUIRE(8,OPENED=OPN)
        FILNM2=FILNM
        IF(OPN) CLOSE(8,STATUS='DELETE')
      END IF

C   Now open the file for storing the new data
      OPEN(UNIT=3,FILE=FILNM2,STATUS='NEW')

      END IF

      END IF

C   Whether or not the user wanted to change the data, the data are
C   saved. Unit 1 is the main output file for the simulator. Unit 2
C   is the "warm start" file (which contains all the input data and the
C   adsorbed phase concentrations, if the program ends normally).
C   Unit 3 is the new (modified) bed data file, and is only saved if
C   the file has been changed.
      LIM=2
      IF(MODIFY) THEN
        LIM=3
      ELSE
        FILNM2=FILNM
      END IF

C   If this a warm start, read in the data
      IF(IWRM.EQ.1) THEN
        NUMU=8
        CALL TREAD(NUMU,FILNM2,COMENTS,PARAMS)
      END IF

      DO IFLG=1,LIM
C   Comments
        CALL CCNG(COMENTS,FILNM2,IFLG)

C   When IFLG is non-zero, TPARAM outputs all data in a single call
        CALL TPARAM(PARAMS,NN,INVAL,IFLG)
      END DO

```


END

```

SUBROUTINE SNAPOUT(NUOUT,XTIME,CTIME,ADSO2,ADSTOT,
+   O2MOL,U,PBED,LUMPS1,DELTAX)
C   This subroutine writes the "snapshot" output files

PARAMETER (NZ=5,MAXSTPS=5,MAXLUMP=101,NZP1=6)

COMMON/PLEN/YO2,YO2M1

DIMENSION ADSO2(MAXLUMP,2),ADSTOT(MAXLUMP,2),O2MOL(MAXLUMP,2),
+   U(MAXLUMP,2),PBED(2)

CHARACTER*11 FILNM,VAR*2,DTIME*8,DDATE*9

C   PARAMETERS are defined in BINFLEY
C   PLEN COMMON variables are defined in STARTUP
C   Arguments are defined in PROC

C   Variable definitions:
C   DDATE      Character variable for the date
C   DTIME      Character variable for the time
C   FILNM      Character variable for the name of
C               the file to which the data will be
C               saved (if NUOUT < 3, the save file is the
C               warm start file)
C   VAR        Character variable used in construction
C               of FILNM
C   Note that XTIME in the argument list is the same as TIME in the
C   other program units. XTIME is used here because of the call to
C   the system routine TIME.

C   Get the date and the time
CALL DATE(DDATE)
CALL TIME(DTIME)

C   Construct FILNM if NUOUT > 3
IF(NUOUT.GT.3) THEN
    NUVAR=NUOUT
    ENCODE(2,100,VAR) NUVAR
    FILNM='SNAP0'//VAR//'.DAT'
    OPEN(UNIT=NUOUT,NAME=FILNM,STATUS='NEW')
    END IF

C   Write the output
WRITE(NUOUT,110) XTIME,CTIME,YO2,YO2M1,DDATE,DTIME
DO I=1,2
    XTOT=0.
    WRITE(NUOUT,120) I,PBED(I)
    DO J=1,LUMPS1
        WRITE(NUOUT,130) J,XTOT,O2MOL(J,I),U(J,I),
+           ADSO2(J,I),ADSTOT(J,I)
        XTOT=XTOT + DELTAX
    END DO

    END DO

CLOSE(NUOUT)

RETURN

```

```

100  FORMAT(I2)
110  FORMAT(T5,'Bed Profiles. Time= ',F10.4,' sec',T40,
+    'Time in Cycle= ',F8.4,' sec'/T5,'Plenum Vars: YO2= ',F6.4,
+    3X,'YO2M1= ',F6.4/T5,'Run on ',A9,T25,'at ',A8)
120  FORMAT(T5,'Bed#',I2,T13,'Pressure= ',E10.4,1X,'MPa'/
+    T7,'Distance',2X,'O2 Mole',2X,'Velocity',2X,'O2 Ads',
+    5X,'Tot Ads'/2X,'#',5X,'(m)',7X,'Frac',4X,'(m/s)',
+    3X,'(kgmol/m3)',1X,'(kgmol/m3)')
130  FORMAT(1X,I3,1X,E10.4,1X,F6.4,1X,E10.4,1X,E10.4,1X,E10.4)

C2345678901234567890123456789012345678901234567890123456789012
END

```

```

SUBROUTINE STARTUP(SIMST,ADSO2,ADSTOT,CTIME,ENDS,PBED)
C   This subroutine gets things started

PARAMETER (NZ=5,MAXSTPS=5,MAXLUMP=101,NZP1=6)

COMMON/RDAT/ RBFLOW(MAXSTPS,2),RO2IN(MAXSTPS,2),
+  RPSUP(MAXSTPS,2),RPOUT(MAXSTPS,2),RTCYC(MAXSTPS,2),
+  SNAPTIM(MAXSTPS,2)

COMMON/RDAT2/ DELTAX,DELTAT,SIMTIM,LUMPS,LUMPS1,NTIM

COMMON/BDAT/ BEDD(NZ),ZONEL(NZ),DIFF(NZ),VOIDF(NZ),TZONE(NZ),
+  ZONEKA(NZ),ZONEKB(NZ),ZONEB(NZ),NZONE,FAREA(NZ),BBETA(NZ)

COMMON/INIT/PATM,TEMPI,O2BLKD
COMMON/GASES/O2MW,RN2MW,RGAS1
COMMON/PLEN/YO2,YO2M1

DIMENSION ADSO2(MAXLUMP,2),ADSTOT(MAXLUMP,2),ENDS(2,2),
+  PBED(2)
CHARACTER*1 VAL,COMENTB(5)*80,COMENTR(5)*80,FILNMB*11,
+  FILNMR*11
LOGICAL LOGDUM,LOGD2

C   RDAT COMMON variables are defined in RUNTBL
C   RDAT2 COMMON variables are defined in RUNTBL
C   GASES COMMON variables are defined in BINKLEY
C   BDAT COMMON variables are defined in BEDDAT
C   INIT COMMON variables are defined in BINKLEY
C   PARAMETERS are defined in BINKLEY
C   Arguments are defined in BINKLEY

C   Variable definitions:
C   COMENTB           Array of comment lines for the bed data
C   COMENTR           Array of comment lines for the run
C                     table data
C   FILNMB            File name for the bed data
C   FILNMR            File name for the run table data
C   LOGDUM,LOGD2      Dummy logical variables
C   SIMACT            The actual stop time of a simulation being
C                     read for a warm start
C   VAL               Character variables used in reading
C                     responses to yes/no questions
C   YO2,YO2M1         Mole fraction outputs from PLENUM (current,
C                     previous, respectively)

C   Write intro and ask if this is a warm start
WRITE(6,90)
WRITE(6,100)
READ(5,300) VAL

CALL YES(VAL,LOGDUM)

C   If it is a warm start, call WRMST, get new SIMTIM, and reset SIMST
IF(LOGDUM) THEN
  CALL WRMST(SIMACT,ADSO2,ADSTOT,CTIME,ENDS,PBED,COMENTB,
+  FILNMB,COMENTR,FILNMR)
  WRITE(6,110) SIMACT

```

```

      READ(5,*) SIMTIM
      SIMST=SIMACT

      ELSE
C      It is not a warm start.  Initialize variables.
        IWRM=0
        SIMST=0.
        CTIME=0.
        YO2=O2BLKD
        YO2M1=O2BLKD
        DO I=1,2
          PBED(I)=PATM
          DO J=1,2
            ENDS(I,J)=O2BLKD
          END DO
        END DO
C      Call the data reading routines.  BEDDAT fills the BDAT COMMON;
C      RUNTBL fills the RDAT COMMON.
        CALL BEDDAT(IWRM,FILNMB,COMENTB)
        CALL RUNTBL(IWRM,FILNMR,COMENTR)

C      Ask if the user wants to quit after editing the data sets
        WRITE(6,120)
        READ(5,300) VAL
        CALL YES(VAL,LOGD2)
        IF(LOGD2) THEN
          CLCSE(UNIT=1,STATUS='DELETE')
          CLOSE(UNIT=2,STATUS='DELETE')
          STOP
        END IF

C      Now adjust DELTAX to get an integer number of lumps.  Also, calculate
C      LUMPS1 and convert DELTAX and ZONEL(I) to meters.
        Z=0.
        DO I=1,NZONE
          Z=Z + ZONEL(I)
          ZONEL(I)=ZONEL(I)/100.
        END DO
        LLIM=MAXLUMP-1
        LUMPS=INT(Z/DELTAX)
        IF(LUMPS.GT.LLIM) LUMPS=LLIM
        LUMPS1=LUMPS+1
        DELTAX=Z/(FLOAT(LUMPS)*100.)

C      Now initialize the adsorbed gas arrays, if this is a normal start.
        DO I=1,2
          XTOT=0.
          NUMZ=1
C      Call XGETPAM to get parameter values at position XTOT.
          CALL XGETPAM(XTOT,ZBEDD,ZLEN,ZDIFF,ZVOIDF,ZTEMP,
+           ZKA,ZKB,ZB,NUMZ,BETA,AREA)
          NUMZ=NUMZ + 1
          XZONE=ZLEN
          CINT=PBED(I)*1.E6/(RGAS1*(TZONE(1) + 273.16))
          CINTO2=CINT*O2BLKD
          CINTN2=CINT*(1. - O2BLKD)
C      Loop on lumps.  Only call XGETPAM when a zone boundary has
C      been crossed.

```

```

      DO J=1,LUMPS1
        IF(XTOT.GT.XZONE) THEN
C      Update the zone parameters
          CALL XGETPAM(XTOT,ZBEDD,ZLEN,ZDIFF,ZVOIDF,ZTEMP,
+          ZKA,ZKB,ZB,NUMZ,BETA,AREA)
          XZONE=XZONE + ZLEN
          NUMZ=NUMZ + 1
        END IF
        ADSO2(J,I)=CINTO2/ZKA
        ADSN2=CINTN2*ZB/(ZB*ZKB + CINTN2)
        ADSTOT(J,I)=ADSO2(J,I) + ADSN2
        XTOT=XTOT + DELTAX
      END DO
    END IF

    RETURN
90    FORMAT(T10,'Welcome to the OBOGS simulator'/,
+      T10,'This version completed in April, 1988'/////////)
100   FORMAT(T10,'Is this a warm start?')
110   FORMAT(T10,'The last stop time was ',F10.4,' seconds'/
+      T10,'Enter new stop time')
120   FORMAT(T10,'Data files are edited'/T15,
+      'Do you want to quit?')
300   FORMAT(A1)

C2345678901234567890123456789012345678901234567890123456789012
END

```



```

SUBROUTINE TCHECK(ARRAY,LIMIT,TIME,RESULT)
C   This routine checks ARRAY to see if, at the current TIME, RESULT
C   has a new value. Parameter values are in ARRAY(N,1); ARRAY(N,2)
C   contains the times. LIMIT is the number of possible steps.

DIMENSION ARRAY(LIMIT,2)

DO I=1,LIMIT
  IF (TIME.GE.ARRAY(I,2)) THEN
C   ARRAY(I,2) values are set to -1 after that time has passed.
C   Thus, RESULT is only updated when a new positive time has
C   been reached. ARRAY(I,2) should be set to a large number
C   (at least greater than SIMTIM) when no more step changes are
C   desired.
      IF (ARRAY(I,2).GE.0) THEN
        RESULT=ARRAY(I,1)
        ARRAY(I,2)=-1.
      END IF
    END IF
  END DO

RETURN

C2345678901234567890123456789012345678901234567890123456789012
END

```

SUBROUTINE TGETPAM(BFLOW,O2IN,PSUP,POUT,TCYC,SNAP,TIME)

C This routine gets the proper values for the scalar variables
C in the argument list from the COMMON blocks RDAT and RDAT2, based
C on TIME. The COMMON block variables are defined in RUNTBL; the
C variables in the argument list have the same definitions as their
C analogs in the COMMON blocks except that those above are scalars.

C SNAP is a flag for "snapshots," where the data for each bed are
C output as a function of position at a specified time. If SNAP is
C negative, nothing is saved; if SNAP is greater than 3, PROC calls
C SNAPOUT to output the data.

PARAMETER (NZ=5,MAXSTPS=5,MAXLUMP=101,NZP1=6)
COMMON/RDAT/ RBFLOW(MAXSTPS,2),RO2IN(MAXSTPS,2),
+ RPSUP(MAXSTPS,2),RPOUT(MAXSTPS,2),RTCYC(MAXSTPS,2),
+ SNAPTIM(MAXSTPS,2)
COMMON/RDAT2/ DELTAX,DELTAT,SIMTIM,LUMPS,LUMPS1,NTIM

NN=MAXSTPS
SNAP=-1.

C TCHECK actually checks the values
CALL TCHECK(RBFLOW,NN,TIME,BFLOW)
CALL TCHECK(RO2IN,NN,TIME,O2IN)
CALL TCHECK(RPSUP,NN,TIME,PSUP)
CALL TCHECK(RPOUT,NN,TIME,POUT)
CALL TCHECK(RTCYC,NN,TIME,TCYC)
CALL TCHECK(SNAPTIM,NN,TIME,SNAP)

RETURN

C2345678901234567890123456789012345678901234567890123456789012
END

```

SUBROUTINE TPARAM(PARAMS,NN,INVAL,IFLG)
C   This subroutine changes the values of the time varying parameters.
C   Variables are defined in RUNTBL

DIMENSION PARAMS(10,NN,2)
LOGICAL LOGDUM
CHARACTER*80 FMATS(10),NUMS(5)*2

DATA FMATS/'(T5,'Space Step (cm)')',
+ '(T5,'Time Step (seconds)')',
+ '(T5,'Simulation Time (seconds)')',
+ '(T5,'Number of Output Points')',
+ '(T5,'Breathing Flow (STD lpm)',T35,'Time (sec)')',
+ '(T5,'Inlet O2 (mole fraction)',T35,'Time (sec)')',
+ '(T5,'Supply Pressure (MPa,abs)',T35,'Time (sec)')',
+ '(T5,'Exhaust Pressure (MPa,abs)',T35,'Time (sec)')',
+ '(T5,'Cycle Time (seconds)',T35,'Time (sec)')',
+ '(T5,'Snap Shot Unit (>3)',T35,'Time (sec)')'/
DATA NUMS/'1.','2.','3.','4.','5.'/

LOGDUM=.TRUE.

C   Do the interactive case first
IF(IFLG.EQ.0) THEN
    WRITE(6,FMATS(INVAL))

C   The simple parameters are INVAL less than 3. Print the old value
C   first; then read in the new value.
    IF(INVAL.LE.4) THEN
        WRITE(6,100) PARAMS(INVAL,1,1)
        READ(5,*) PARAMS(INVAL,1,1)

C   For INVAL greater than 4, output the appropriate table of
C   parameter values and times
        ELSE
            DO WHILE(LOGDUM)

C   Write all the parameter values to the screen. Then, read which is
C   to be changed and take the appropriate action (ININ less than 0
C   exits)
                DO I=1,NN
                    WRITE(6,110) NUMS(I),PARAMS(INVAL,I,1),PARAMS(INVAL,I,2)
                END DO

                WRITE(6,115)
                READ(5,*) ININ

                IF(ININ.LE.0) THEN
                    LOGDUM=.FALSE.
                ELSE IF(ININ.LE.NN) THEN
                    WRITE(6,120)
                    READ(5,*) PARAMS(INVAL,ININ,1)
                    WRITE(6,125)
                    READ(5,*) PARAMS(INVAL,ININ,2)
                    END IF

            END DO

```

```

        END IF

C      Now, consider the non-interactive case.  TPARAM outputs all data
C      in a single call for this case.
      ELSE
        DO II=1,10
          WRITE(IFLG,FMATS(II))
          IF(II.LE.4) THEN
            WRITE(IFLG,105) PARAMS(II,1,1)
          ELSE
            DO J=1,NN
              NUMS(J)=' '
              WRITE(IFLG,110) NUMS(J),PARAMS(II,J,1),PARAMS(II,J,2)
            END DO
          END IF
        END DO

      END IF

    RETURN

100  FORMAT(T10,'Current value=',F10.4/T10,'Enter new value')
105  FORMAT(T10,'Current value=',T30,F10.4)
110  FORMAT(1X,A2,T5,E10.4,T30,E10.4)
115  FORMAT(T10,'Enter number to change parameter'/
+ T10,'Enter zero to exit'/)
120  FORMAT(T10,'Enter new parameter value')
125  FORMAT(T10,'Enter corresponding start time (sec)')

C2345678901234567890123456789012345678901234567890123456789012
END

```

```

SUBROUTINE TREAD(NUMU,FILNM,COMENTS,PARAMS)
C   This routine reads in the data on the run table parameters

PARAMETER (NZ=5,MAXSTPS=5,MAXLUMP=101,NZP1=6)
DIMENSION PARAMS(10,MAXSTPS,2)
CHARACTER COMENTS(*)*(*),FILNM*(*),VAR*1

C   PARAMETERS are defined in BINKLEY
C   PARAMS equivalences are in RUNTBL

C   Variable definitions:
C   COMENTS           Array of comment lines that describe the
C                       data file.
C   FILNM             The first line of the data file contains
C                       the name of the data file. FILNM is that
C                       data file name.
C   NUMU              The number of the output unit.
C   VAR               A dummy variable used in reading blank
C                       input lines.

C   Don't rewind the unit. Just read in the comments and the
C   simple parameters. Convert the real variable RNTIM to the
C   integer NTIM for COMMON RDAT2.
  READ(NUMU,100) FILNM, (COMENTS(I),I=1,5),PARAMS(1,1,1),
+   PARAMS(2,1,1),PARAMS(3,1,1),PARAMS(4,1,1)

C   Now read the tabulated data.
  DO J=5,10
    READ(NUMU,110) VAR
    DO I=1,MAXSTPS
      READ(NUMU,120) PARAMS(J,I,1),PARAMS(J,I,2)
    END DO
  END DO

  RETURN

100  FORMAT(T60,A11/5(1X,A72//)/T30,F10.4//T30,F10.4//
+   T30,F10.4//T30,F10.4)
110  FORMAT(A1)
120  FORMAT(T5,E10.4,T30,E10.4)

C2345678901234567890123456789012345678901234567890123456789012
END

```

```

SUBROUTINE VALVE(PUP,PDWN,W,NUMVLV,TEMP)
C   Given the upstream (PUP) and downstream (PDWN) pressures and the
C   valve number (NUMVLV), this routine returns the mass flowrate
C   (W) in (kg/s). The temperature (TEMP, K) now enters into the
C   calculation.

COMMON/VLVCON/CONST1,CONST2,CVCP,CRITRAT
COMMON/BDAT2/ VALVB,VALVO,VALVS,VOLPLEN

C   BDAT2 COMMON variables are defined in BEDDAT
C   VLVCON COMMON variables are defined in BINKLEY
C   Arguments are defined above.

C   Variable definitions:
C   NUMVLV           Valve identifier(1=supply,2=exhaust,
C                   3=bypass)
C   PDWNL,PUPL       The valve constants are for pressures in
C                   Pa. PUP and PDWN are in MPa. PDWNL,PUPL
C                   are the local equivalents in Pa.
C   PRAT             Ratio of downstream over upstream pressures
C   SIGN             This is negative when the downstream
C                   pressure is greater than the upstream
C                   pressure

PUPL=PUP*1.E6
PDWNL=PDWN*1.E6
SIGN=1.
PRAT=PDWNL/PUPL

IF(NUMVLV.EQ.1) CDA=VALVS
IF(NUMVLV.EQ.2) CDA=VALVO
IF(NUMVLV.EQ.3) CDA=VALVB

IF(PRAT.GT.1.) THEN
    SIGN=-1.
    PRAT=1./PRAT
    PUPL=PDWNL
    END IF

IF(PRAT.EQ.1.) THEN
    W=0.
    RETURN
    END IF

IF(PRAT.LT.CRITRAT) THEN
C   Choked flow
    W=SIGN*CDA*CONST2*PUPL/SQRT(TEMP)
ELSE
    FACTOR=1.- PRAT**(CVCP*(1./CVCP - 1.))
    W=SIGN*CDA*CONST1*PUPL*(PRAT**CVCP)*SQRT(FACTOR)/SQRT(TEMP)
    END IF

RETURN

C23456789012345678901234567890123456789012345678901234567890123456789012
END

```

```

SUBROUTINE VPCNG(DIAMB,DIAMS,DIAMO,DCOEFB,DCOEFS,DCOEFO,VOLPLEN,
+ NZONE,IFLG)

```

```

C   This subroutine allows for the interactive manipulation of
C   valve parameters for the OBOGS simulation. Specifically,
C   the bypass, supply, and outlet valve diameters and discharge
C   coefficients can be manipulated.

```

```

C   Original routine by Glenn Munkvold 2/7/87
C   Modified by Glenn Munkvold 4/87

```

```

DIMENSION PARAM(7)
CHARACTER*2 NUMS(8),FMATS(3)*80
LOGICAL LOGDUM
DATA NUMS/'1.','2.','3.','4.','5.','6.','7.','8.'/
DATA FMATS/'(T4,'Bypass',T15,A2,F10.4,T34,A2,F10.4)',
+ '(T4,'Supply',T15,A2,F10.4,T34,A2,F10.4)',
+ '(T4,'Outlet',T15,A2,F10.4,T34,A2,F10.4)'/

```

```

C   Variable definitions follow:
C   DCOEFB           Discharge coefficient of the bypass valve
C   DCOEFO           Discharge coefficient of the outlet valve
C   DCOEFS           Discharge coefficient of the supply valve
C                   Positive discharge coefficients passed
C                   to the routine are used; negative values
C                   cause the discharge coefficients to be
C                   calculated
C   DIAMB            Diameter of the bypass valve (cm)
C   DIAMO            Diameter of the outlet valve (cm)
C   DIAMS            Diameter of the supply valve (cm)
C   FMATS            Character array of formats for the real
C                   variables on output
C   IFLG             Flag that indicates whether this call is
C                   interactive (IFLG=0) or a call to print
C                   to unit number IFLG.
C   LOGDUM           Dummy logical variable used in return
C                   from YES routine, which tests for
C                   affirmative interactive responses
C   NZONE            The number of different zones in the bed
C   PARAM            Real array of parameter values for use in
C                   DO loops
C   VOLPLEN          The volume of the mixing plenum,
C                   cubic meters

```

```

LOGDUM=.TRUE.
PARAM(1)=DIAMB
PARAM(2)=DCOEFB
PARAM(3)=DIAMS
PARAM(4)=DCOEFS
PARAM(5)=DIAMO
PARAM(6)=DCOEFO
PARAM(7)=VOLPLEN

```

```

C   Interactive mode
IF(IFLG.EQ.0) THEN
  WRITE(6,100)
  DO WHILE (LOGDUM)

```

```

C      Print all parameter values to the screen
      WRITE(6,110)
      DO I=1,3
        J=2*I-1
        WRITE(6,FMATS(I)) NUMS(J),PARAM(J),NUMS(J+1),PARAM(J+1)
      END DO
      WRITE(6,200) NUMS(7),PARAM(7)
      WRITE(6,210) NUMS(8),NZONE

C      Read the number to change (zero to exit)
      WRITE(6,115)
      READ(5,*) INVAL

C      Classify the response and take action.  INVAL=0 exits routine.
      IF(INVAL.LE.0) THEN
        LOGDUM=.FALSE.
      ELSE
        IF(INVAL.LE.6) THEN
          IF(MOD(FLOAT(INVAL),2.).GT.1.E-4) THEN
            I=(INVAL+1)/2
          ELSE
            I=INVAL/2
          END IF
          J=2*I-1
          WRITE(6,FMATS(I)) NUMS(J),PARAM(J),NUMS(J+1),PARAM(J+1)
          WRITE(6,120) PARAM(INVAL)
          READ(5,*) PARAM(INVAL)
        ELSE
          IF(INVAL.EQ.7) THEN
            WRITE(6,120) PARAM(INVAL)
            READ(5,*) PARAM(INVAL)
          END IF
          IF(INVAL.EQ.8) THEN
            WRITE(6,125) NZONE
            READ(5,*) NZONE
          END IF
        END IF
      END IF

C      End the DO WHILE loop
      END DO

C      Now consider the non-interactive case
C      Write parameter values to unit number IFLG
      ELSE
        WRITE(IFLG,110)
        DO I=1,3
          J=2*I-1
          NUMS(J)=' '
          NUMS(J+1)=' '
          WRITE(IFLG,FMATS(I)) NUMS(J),PARAM(J),NUMS(J+1),PARAM(J+1)
        END DO
        WRITE(IFLG,200) NUMS(6),PARAM(7)
        WRITE(IFLG,210) NUMS(6),NZONE

      END IF

      DIAMB=PARAM(1)
      DCOEFB=PARAM(2)

```



```
DIAMS=PARAM(3)
DCOEFS=PARAM(4)
DIAMO=PARAM(5)
DCOEFO=PARAM(6)
VOLPLEN=PARAM(7)
```

```
RETURN
```

```
100  FORMAT(///' Valve Parameter Changes'/
+   T10,'Enter number to change parameter'/
+   T10,'Enter zero to exit'/)
110  FORMAT(/T4,'Valve',T16,'Diameter(cm)',T35,
+   'Discharge Coeff ')
115  FORMAT(T10,'Enter number (zero to exit):')
120  FORMAT(T10,'Enter new value'/T15,'Old value =',F11.5)
125  FORMAT(T10,'Enter new value'/T15,'Old value =',I4)
200  FORMAT(1X,A2,'Plenum Volume= ',E10.3,T31,' (cubic meters)')
210  FORMAT(1X,A2,'Number of Bed Zones=',I2,T28,' (maximum is 5)')
```

```
C2345678901234567890123456789012345678901234567890123456789012
END
```

```

SUBROUTINE WRMST(SIMACT,ADSO2,ADSTOT,CTIME,ENDS,PBED,COMENTB,
+      FILNMB,COMENTR,FILNMR)
C      This routine reads in all the appropriate data to restart a
C      simulation run that stops normally but did not run long
C      enough. The routine can also be used to restart a run
C      from the last snapshot. This would require the user to
C      create a file where the snapshot output is appended to
C      the end of the run table data in a warm start or the main
C      output file.

PARAMETER (NZ=5,MAXSTPS=5,MAXLUMP=101,NZP1=6)

COMMON/RDAT2/ DELTAX,DELTAT,SIMTIM,LUMPS,LUMPS1,NTIM

COMMON/BDAT/ BEDD(NZ),ZONEL(NZ),DIFF(NZ),VOIDF(NZ),TZONE(NZ),
+  ZONEKA(NZ),ZONEKB(NZ),ZONEB(NZ),NZONE,FAREA(NZ),BBETA(NZ)

DIMENSION ADSO2(MAXLUMP,2),ADSTOT(MAXLUMP,2),ENDS(2,2),
+  PBED(2)
CHARACTER*1 VAL,COMENTB(5)*80,COMENTR(5)*80,FILNMB*11,
+  FILNMR*11

C      RDAT2 COMMON variables are defined in RUNTBL
C      BDAT COMMON variables are defined in BEDDAT
C      PARAMETERS are defined in BINKLEY
C      Arguments are defined in STARTUP

C      Open the file and call the reading routines.
NUMU=8
IWRM=1
OPEN(UNIT=8,FILE='START.WRM',STATUS='OLD')

C      Input and output all data through BEDDAT and RUNTBL with IWRM=1
C      to suppress the interactive mode
CALL BEDDAT(IWRM,FILNMB,COMENTB)
CALL RUNTBL(IWRM,FILNMR,COMENTR)

C      Now adjust DELTAX to get an integer number of lumps. Also, calculate
C      LUMPS1 and convert DELTAX and ZONEL(I) to meters.
Z=0.
DO I=1,NZONE
  Z=Z + ZONEL(I)
  ZONEL(I)=ZONEL(I)/100.
END DO
LLIM=MAXLUMP-1
LUMPS=INT(Z/DELTAX)
IF(LUMPS.GT.LLIM) LUMPS=LLIM
LUMPS1=LUMPS+1
DELTAX=Z/(FLOAT(LUMPS)*100.)

C      Now read in the adsorbed phase data
CALL ADSREAD(NUMU,TIME,CTIME,ADSO2,ADSTOT,ENDS,
+  PBED,LUMPS1)

C      And set SIMACT=TIME (the time to start, SIMACT, is the last
C      time output to the warm start file, TIME)
SIMACT=TIME

```

RETURN

C2345678901234567890123456789012345678901234567890123456789012
END

```

SUBROUTINE XGETPAM(XTOT,ZBEDD,ZLEN,ZDIFF,ZVOIDF,ZTEMP,
+   ZKA,ZKB,ZB,NUMZ,BETA,AREA)
C   This routine returns the proper values for the above parameters
C   based on NUMZ, the zone number (which ought to make the thing
C   a pain for backward integrations). Variables are defined in
C   BEDDAT. If NUMZ < 1, then the routine uses XTOT, the absolute
C   distance down the bed, to get the parameters for return.

PARAMETER (NZ=5,MAXSTPS=5,MAXLUMP=101,NZP1=6)
COMMON/BDAT/ BEDD(NZ),ZONEL(NZ),DIFF(NZ),VOIDF(NZ),TZONE(NZ),
+   ZONEKA(NZ),ZONEKB(NZ),ZONEB(NZ),NZONE,FAREA(NZ),BBETA(NZ)

IF(NUMZ.LT.1) THEN
  XX=0.
  IFG=1
  DO WHILE(IFG.LE.NZONE)
    XX=XX+ZONEL(IFG)
    IF(XTOT.LE.XX) THEN
      NUMZ=IFG
      IFG=NZONE + 1
    ELSE
      IFG=IFG + 1
    END IF
  END DO
END IF

IF (NUMZ.LE.NZONE.AND.NUMZ.GE.1) THEN
  ZBEDD=BEDD(NUMZ)
  ZLEN=ZONEL(NUMZ)
  ZDIFF=DIFF(NUMZ)
  ZVOIDF=VOIDF(NUMZ)
  ZTEMP=TZONE(NUMZ)
  ZKA=ZONEKA(NUMZ)
  ZKB=ZONEKB(NUMZ)
  ZB=ZONEB(NUMZ)
  AREA=FAREA(NUMZ)
  BETA=BBETA(NUMZ)
ELSE
C   If NUMZ is out of range, default to NUMZ = NZONE, and set NUMZ = -1
C   so that XTOT will be used on the next call.
  ZBEDD=BEDD(NZONE)
  ZLEN=ZONEL(NZONE)
  ZDIFF=DIFF(NZONE)
  ZVOIDF=VOIDF(NZONE)
  ZTEMP=TZONE(NZONE)
  ZKA=ZONEKA(NZONE)
  ZKB=ZONEKB(NZONE)
  ZB=ZONEB(NZONE)
  AREA=FAREA(NZONE)
  BETA=BBETA(NZONE)
  NUMZ=-1
END IF

RETURN
END

```

SUBROUTINE YES (VAL, RES)

C This subroutine checks the response to see if it is
C affirmative during an interactive session.
C VAL is the input character variable, RES is the logical
C variable returned.

CHARACTER VAL*(*)
LOGICAL RES

IF (VAL.EQ.'y'.OR.VAL.EQ.'Y') THEN
RES=.TRUE.
ELSE
RES=.FALSE.
END IF

RETURN

C2345678901234567890123456789012345678901234567890123456789012
END

```
SUBROUTINE ZCNG(IZ,BEDD,ZONEL,DIFF,VOIDF,TZONE,ZONEKA,ZONEKB,
+ ZONEB,IFLG)
```

```
C This subroutine allows for the manipulation of zone specific
C parameters.
```

```
LOGICAL LOGDUM
CHARACTER*2 NUMS(9),FMATS(8)*80
DIMENSION PARAM(8)
```

```
DATA NUMS/'1.','2.','3.','4.','5.','6.','7.','8.','9.'/
```

```
DATA FMATS/
```

```
+ '(1X,A2,'Bed Diameter=',F10.4,2X,'(cm)')',
+ '(1X,A2,'Zone Length=',F10.4,2X,'(cm)')',
+ '(1X,A2,'Diffusion Coefficient=',F10.4,2X,'(1/sec)')',
+ '(1X,A2,'Void Fraction=',F6.4)',
+ '(1X,A2,'Temperature=',F6.2,2X,'(degrees C)')',
+ '(1X,A2,'KA=',F10.4,2X,'(kgmol gas/kgmol ads, O2)')',
+ '(1X,A2,'KB=',F10.4,2X,'(kgmol gas/kgmol ads, N2)')',
+ '(1X,A2,'B=',F8.3,2X,'(kgmol N2 ads/cubic meter)')'/
```

```
C Variable definitions follow:
```

```
C BEDD      Bed diameters, one diameter for
C           each zone (cm)
C DIFF      Mass transfer coefficient
C           for each zone (1/second)
C FMATS      Character array of formats for the real
C           variables on output
C IFLG      Flag that indicates whether this call is
C           interactive (IFLG=0) or a call to print
C           to unit number IFLG.
C IZ        Bed zone number. This routine handles
C           only one zone at a time
C LOGDUM      Dummy logical variable used for control
C NUMS      Character array of numbers for interactive
C           output labeling
C PARAM      Real array of parameter values for use in
C           DO loops
C TZONE      Temperature for each zone,
C           degrees C
C VOIDF      Void fractions for each zone
C ZONEB      Nitrogen equil constant,
C           one for each zone (kgmol ads/cubic meter)
C ZONEKA      Oxygen equilibrium constant,
C           one for each zone (kgmol gas/kgmol ads)
C ZONEKB      Nitrogen equil constant,
C           one for each zone (kgmol gas/kgmol ads)
C           For a more extended explanation of the
C           equilibrium constants, see Beaman's
C           June, 1985, paper (noting that the B
C           defined in that paper has incorrect units)
C ZONEL      Length of each bed zone (cm)
```

```
LOGDUM=.TRUE.
PARAM(1)=BEDD
PARAM(2)=ZONEL
PARAM(3)=DIFF
PARAM(4)=VOIDF
```

```

PARAM(5)=TZONE
PARAM(6)=ZONEKA
PARAM(7)=ZONEKB
PARAM(8)=ZONEB

C    Call KTCORR to get KA and KB as functions of temperature
    CALL KTCORR(PARAM(5),PARAM(6),PARAM(7))

C    Interactive mode for IFLG=0
    IF(IFLG.EQ.0) THEN
        WRITE(6,100)
        DO WHILE(LOGDUM)
C    Write all parameters to the screen
            WRITE(6,110) NUMS(1),IZ
            DO I=1,8
                WRITE(6,FMATS(I)) NUMS(I+1),PARAM(I)
            END DO

            WRITE(6,120)

C    Write prompt and read response for action
            WRITE(6,130)
            READ(5,*) INVAL

C    Classify the response
            IF(INVAL.LE.0) THEN
                LOGDUM=.FALSE.
                IZ=0
            ELSE IF(INVAL.LE.9) THEN
                IF(INVAL.EQ.1) THEN
                    WRITE(6,110) NUMS(1),IZ
                    WRITE(6,140)
                    READ(5,*) IZ
                    LOGDUM=.FALSE.
                ELSE
                    WRITE(6,FMATS(INVAL-1)) NUMS(INVAL),PARAM(INVAL-1)
                    WRITE(6,140)
                    READ(5,*) PARAM(INVAL-1)
                END IF
            END IF

C    End the DO WHILE loop
        END DO

C    Now consider the non-interactive case
    ELSE
        NUMS(1)=' '
        WRITE(IFLG,110) NUMS(1),IZ
        DO I=1,8
            NUMS(I+1)=' '
            WRITE(IFLG,FMATS(I)) NUMS(I+1),PARAM(I)
        END DO
        WRITE(IFLG,120)
    END IF

C    Call KTCORR here again to update any changes
    CALL KTCORR(PARAM(5),PARAM(6),PARAM(7))

```

```
BEDD=PARAM(1)
ZONE1=PARAM(2)
DIFF=PARAM(3)
VOIDF=PARAM(4)
TZONE=PARAM(5)
ZONEKA=PARAM(6)
ZONEKB=PARAM(7)
ZONEB=PARAM(8)
```

```
RETURN
```

```
100  FORMAT(///' Zone Parameter Changes'/
+   T10,'Enter number to change parameter'/
+   T10,'Enter zero to exit'//)
110  FORMAT(/1X,A2,'Zone # ',I2)
120  FORMAT(T6,'KA, KB <0 are calculated from temperature')
130  FORMAT(T10,'Enter number (zero to exit):')
140  FORMAT(T10,'Enter new value')

C2345678901234567890123456789012345678901234567890123456789012
END
```


COUPISO.FOR

```

SUBROUTINE ISOTHM(ZKA,ZKB,ZB,ADSO2,ADSTOT,CINTO2,CINTOT)
C   This routine is the coupled isotherm case. The routine
C   returns the interfacial concentrations given the adsorbed
C   phase concentrations. Variables have the same meanings
C   and units here as defined before. However, all of the
C   variables in this routine are scalars (no arrays). See
C   Beaman's 1983 paper for details and explanations of this
C   routine and the function FEQ1. COMMON EQFCN is used by
C   FEQ1.

EXTERNAL FEQ1

COMMON/EQFCN/ALPHA,FRACN2

C   ADSN2          The adsorbed concentration of nitrogen
C                   (kg-mol/cu m)
C   CINTN2          The interfacial concentration of nitrogen
C                   (kg-mol/cu m)
C
C   ADSN2=ADSTOT-ADSO2
C   FRACN2=ADSN2/ADSTOT
C   ALPHA=ADSTOT/ZB
C   ZUPPER=FRACN2
C   ZLOWER=ALPHA*FRACN2/(1.-ALPHA*FRACN2)

CALL FALSEP(FEQ1,ZUPPER,ZLOWER,ZN2)

CINTN2=ZKB*ZB*ZN2
CINTO2=ZKA*ZB*ALOG(1.+CINTN2/(ZKB*ZB))

CINTN2=CINTN2*ADSN2/ADSTOT
CINTO2=CINTO2*ADSO2/ADSTOT

CINTOT=CINTO2+CINTN2

RETURN
END

FUNCTION FEQ1(Z)
C   This function is used for the coupled isotherm case. See
C   Beaman's 1983 paper for details and explanations of this
C   function. COMMON EQFCN is generated by ISOTHM.

COMMON/EQFCN/ALPHA,FRACN2

VAR1=(1.-FRACN2)*Z/ALOG(1.+Z)
VAR2=FRACN2*(1.+Z)

FEQ1=VAR1 + VAR2 -Z/ALPHA

RETURN
END

```

ISOTHM.FOR

```
SUBROUTINE ISOTHM(ZKA,ZKB,ZB,ADSO2,ADSTOT,CINTO2,CINTOT)
C   This routine is the uncoupled isotherm case. The routine
C   returns the interfacial concentrations given the adsorbed
C   phase concentrations. Variables have the same meanings
C   and units here as defined before. However, all of the
C   variables in this routine are scalars (no arrays).

C   ADSN2           The adsorbed concentration of nitrogen
C                   (kg-mol/cu m)
C   CINTN2           The interfacial concentration of nitrogen
C                   (kg-mol/cu m)

      ADSN2=ADSTOT-ADSO2
      CINTN2=ZKB*ADSN2/(1.-ADSN2/ZB)
      CINTO2=ZKA*ADSO2
      CINTOT=CINTO2+CINTN2
      RETURN

C23456789012345678901234567890123456789012345678901234567890123456789012
      END
```

FALSEP.FOR

```
SUBROUTINE FALSEP(FUNZ,XR,XL,XM)
C   This subroutine uses the false position algorithm to find the root
C   of the function FUNZ.  XR and XL are the bounds on the root; XM is
C   the root returned on completion of the routine.

C   EXTERNAL FUNZ

EPS=1.E-6
NUMITS=0
LIMITS=20
C   EPS is the error limit on the root.  NUMITS is the number of iterations
C   made to find the root.  The routine exits whenever EPS is met or the
C   number of iterations exceeds LIMITS.

YL=FUNZ(XL)
YR=FUNZ(XR)

XM=(XL*YR - XR*YL)/(YR - YL)
YM=FUNZ(XM)

DO WHILE (ABS(YM).GT.EPS.AND.NUMITS.LE.LIMITS)
  NUMITS=NUMITS + 1
  IF ((YM*YL).GT.0.) THEN
    XL=XM
    YL=YM
  ELSE
    XR=XM
    YR=YM
  END IF

  XM=(XL*YR - XR*YL)/(YR - YL)
  YM=FUNZ(XM)
END DO

RETURN
END
```

Example Post Processor--PLOTPSA.FOR

```

PROGRAM PLOTPSA
C   This program plots output from the binary OBOGS model
CHARACTER*30 LINE_IN
REAL DIST,O2MOLF,VEL,O2ADS,TOTADS
REAL TIME,O2MASS,TOTMASS
INTEGER LOOPEX,I,LUMP,LOOP LIM
DATA LOOPEX/0/
OPEN(2,FILE='PSA.DAT',STATUS='OLD')
DO WHILE(LOOPEX.EQ.0)
  READ(2,100) LINE_IN
  IF(INDEX(LINE_IN,'OBOGS').NE.0) THEN
    IF(INDEX(LINE_IN,'Output').NE.0) THEN
      LOOPEX=1
      OPEN(3,FILE='PSAPLOT.DAT',STATUS='NEW')
      DO 10 I=1,2
        READ(2,100) LINE_IN
        CONTINUE
      DO 20 I=1,2000
        READ(2,*,END=200) TIME,O2MOLF,O2MASS,TOTMASS
        WRITE(3,*) TIME,O2MOLF,O2MASS,TOTMASS
        CONTINUE
      CONTINUE
      END IF
    END IF
  END DO
  CLOSE(3)
  CLOSE(2)

  OPEN(6,FILE='START.WRM',STATUS='OLD')
  WRITE(*,*) ' Input the number of points per bed'
  READ(*,*) LOOP LIM
  LOOPEX=0
  DO WHILE(LOOPEX.EQ.0)
    READ(6,100) LINE_IN
    IF(INDEX(LINE_IN,'Bed').NE.0) THEN
      IF(INDEX(LINE_IN,'Profiles').NE.0) THEN
        LOOPEX=1
        OPEN(5,FILE='PLTW RM1.DAT',STATUS='NEW')
        OPEN(4,FILE='PLTW RM2.DAT',STATUS='NEW')
        DO 30 I=1,5
          READ(6,100) LINE_IN
          CONTINUE
        DO 40 I=1,LOOP LIM
          READ(6,*) LUMP,DIST,O2MOLF,VEL,O2ADS,TOTADS
          WRITE(5,*) DIST,O2MOLF,VEL,O2ADS,TOTADS
          CONTINUE
        DO 50 I=1,3
          READ(6,100) LINE_IN
          CONTINUE
        DO 60 I=1,LOOP LIM
          READ(6,*) LUMP,DIST,O2MOLF,VEL,O2ADS,TOTADS
          WRITE(4,*) DIST,O2MOLF,VEL,O2ADS,TOTADS
          CONTINUE
        END IF
      END IF
    END IF
  END DO

```

```
        END DO
        CLOSE(6)
        CLOSE(5)
        CLOSE(4)

        CALL MAPFLT('ENDRUN1')
        CALL MAPFLT('ENDRUN2')
        CALL MAPFLT('ENDRUN3')

        STOP
100    FORMAT(A30)
        END
```

DISTRIBUTION LIST

1 copy	Commander US Army Medical Research and Development Command ATTN: SGRD-RMI-S Fort Detrick, Frederick, Maryland 21701-5012
12 copies	Defense Technical Information Center (DTIC) ATTN: DTIC-DDAC Cameron Station Alexandria, VA 22304-6145
1 copy	Dean School of Medicine Uniformed Services University of the Health Sciences 4301 Jones Bridge Road Bethesda, MD 20814-4799
1 copy	Commandant Academy of Health Sciences, US Army ATTN: AHS-CDM Fort Sam Houston, TX 78234-6100