

**Carnegie Mellon University
Research Showcase @ CMU**

Department of Electrical and Computer
Engineering

Carnegie Institute of Technology

1991

Developing reusable model libraries in the ASCEND environment

Joseph J. Zaher
Carnegie Mellon University

Carnegie Mellon University. Engineering Design Research Center.

Follow this and additional works at: <http://repository.cmu.edu/ece>

Recommended Citation

This Technical Report is brought to you for free and open access by the Carnegie Institute of Technology at Research Showcase @ CMU. It has been accepted for inclusion in Department of Electrical and Computer Engineering by an authorized administrator of Research Showcase @ CMU. For more information, please contact research-showcase@andrew.cmu.edu.

NOTICE WARNING CONCERNING COPYRIGHT RESTRICTIONS:

The copyright law of the United States (title 17, U.S. Code) governs the making of photocopies or other reproductions of copyrighted material. Any copying of this document without permission of its author may be prohibited by law.

**Developing Reusable Model Libraries
in the ASCEND Environment**

Joseph J. Zaher

EDRC 06-108-91

CARNEGIE MELLON UNIVERSITY

**DEVELOPING REUSABLE MODEL LIBRARIES
IN THE ASCEND ENVIRONMENT**

Engineering Design Research Center
Research Report Series

submitted by

JOSEPH J.ZAHER

Pittsburgh, PA 15213
My, 1991

This work has **been supported** by the Engineering Design Research Center, a NSF Engineering Research Center.

Table of Contents

1. Introduction	1
2. The Modeling Language	1
3. Building a Mathematics and Thermodynamics Library	3
4. Example	7
5. Conclusions	11

List of Figures

Figure 3-1 An ASCEND Structure for Mathematics Modeling	4
Figure 3-2 An ASCEND Structure for Thermodynamics Modeling	6
Figure 4-1 Isomenization Process Flow Diagram	8
Figure 4-2 Adsorption, Isomenization and Cracking Reactions	8
Figure 4-3 The ASCEND Environment	10
Figure A-1 Atoms.lib	13
Figure A-2 Mathematics.lib	15
Figure A-3 ComponentsJib	21
Figure A-4 Thermodynamics.lib	26
Figure A-5 Isom.asc	30

1. Introduction

In this report, the model-building capabilities of the ASCEND (Advanced System for Computations in Engineering Design) environment are demonstrated. In chemical engineering, many simulation systems, including conventional flowsheeting programs, provide the user with a predefined set of abstractions, such as unit operations (Piela, *et al.*, 1991). These abstractions remain hidden from the user and cannot be readily tailored for specific modeling needs. The model-building language within ASCEND, however, is equation-based and offers more flexibility. It provides many of the object-oriented abstractions that are conducive to a more exact approach to modeling by allowing models to be structured more like the physical systems they are meant to represent (Westerberg, *et al.*, 1991). This is done through the use of two objects (ATOMs and MODELs) and five operators (REFINES, IS_A, IS_REHNED_TO, ARE_THE_SAME, and ARE_ALKE). Analogous to the use of unit operations to aid engineers in constructing flowsheets, libraries can be developed in ASCEND to aid in the proper formulation of equational models. It is the purpose of this work to illustrate the above language primitives in the development of standardized ASCEND structures for mathematics and thermodynamics modeling. The use of these libraries is further demonstrated by formulating and solving an adiabatic catalytic reactor. The rationales for taking such a highly structured approach toward model construction will be explained.

2. The Modeling Language

(Piela, *et al.*, 1991) gives a detailed syntactical description of the language. The following is a summary of the language features required to understand this report

The ASCEND language is strongly typed, requiring that the type of every part in every model be specified. Data in ASCEND are limited to be of the five pre-defined elementary types (*integer*, *string*, *boolean*, *unit*, and *real*). *Integers* and *strings* are used mainly for indexing arrays of objects and are required to be given at compilation time. *Boolean*, *unit*, and *real* valued ATOMs, on the other hand, may have their values changed. *Reals* are unique in that they have an added notion of dimensionality and are used often as meaningfully named constants in the modeling equations. The implicitly valued ATOM is a structured object which may only contain elementary type attributes. Algebraic variables are most generally declared to be of the pre-defined type *generic_real* which is a REFINEment of the elementary type *real*. Because the ATOM *generic_real* REFINES *real*, it inherits the numeric and dimensional

features and provides some additional elementary type attributes as parts to distinguish it as a variable. These attributes include a *real* lower and upper bound, a *boolean* fixed flag, a *real* nominal value, etc., to define its interaction with the equation solver. The *genericjreal* can be further REFINEd to any commonly encountered engineering variable type where the dimensionality and perhaps other attributes can be specified. Figure A-1 in the Appendix contains a listing of an ATOMs library file defining some useful thermodynamic variable types.

The REFINES operator may also be used to inherit the attributes of a previously written MODEL definition. Building MODELS in ASCEND involves describing MODEL types which are composed of instances and configurational relationships between them. Instances are declared locally within a MODEL using the IS_A operator preceded by a list of instance names and followed by a type. If the type is *genericjrecd* (or a refinement thereof), then the instances become system variables and may also exist in equational relationships with other ATOMS. Equations in ASCEND are written in a declarative style, and are treated as objects which can be optionally named by the user. Dimensional consistency is automatically checked by the system. Each equation is given a *boolean* included flag attribute which can be set by the user to specify whether it is to be satisfied or ignored by the solver. Equations from separate MODELS can be accumulated in one simulation by instancing MODEL types within MODEL definitions. This illustrates how complex models can be decomposed into smaller simpler building-Mock models for a more clear understanding of the necessary equations. In addition to a declarative description, all MODELS have a procedural section with which initializations and specifications can be made to prepare the MODEL for solving. Through the environment, MODELS instanced within a simulation can be isolated in the ASCEND solver for local investigation. This provides a means in which the initial values for some variables can be propagated throughout the MODEL by making temporary local specifications, i.e. assigning a TRUE value to some of the variables' fixed flag.

The REFINES operator, as it applies to MODELS, allows the defining of a MODEL as a modification of a more general definition. All of the equations and variables defined in the general MODEL become applicable in the REFINEd MODEL. Such a general MODEL definition is referred to as a base and serves as a root node of an inheritance hierarchy (Piela, *et al*, 1991). An inheritance hierarchy is formed to support varying degrees of specialization or rigorousness and to also maximize code re-use. If an instance of a base type is made within a MODEL definition, then flexibility is provided to specialize the instance via any of the ancestral paths made available through type inheritance. This deferred binding is accomplished with the ISJREFINEDJTO operator. With regard to thermodynamics.

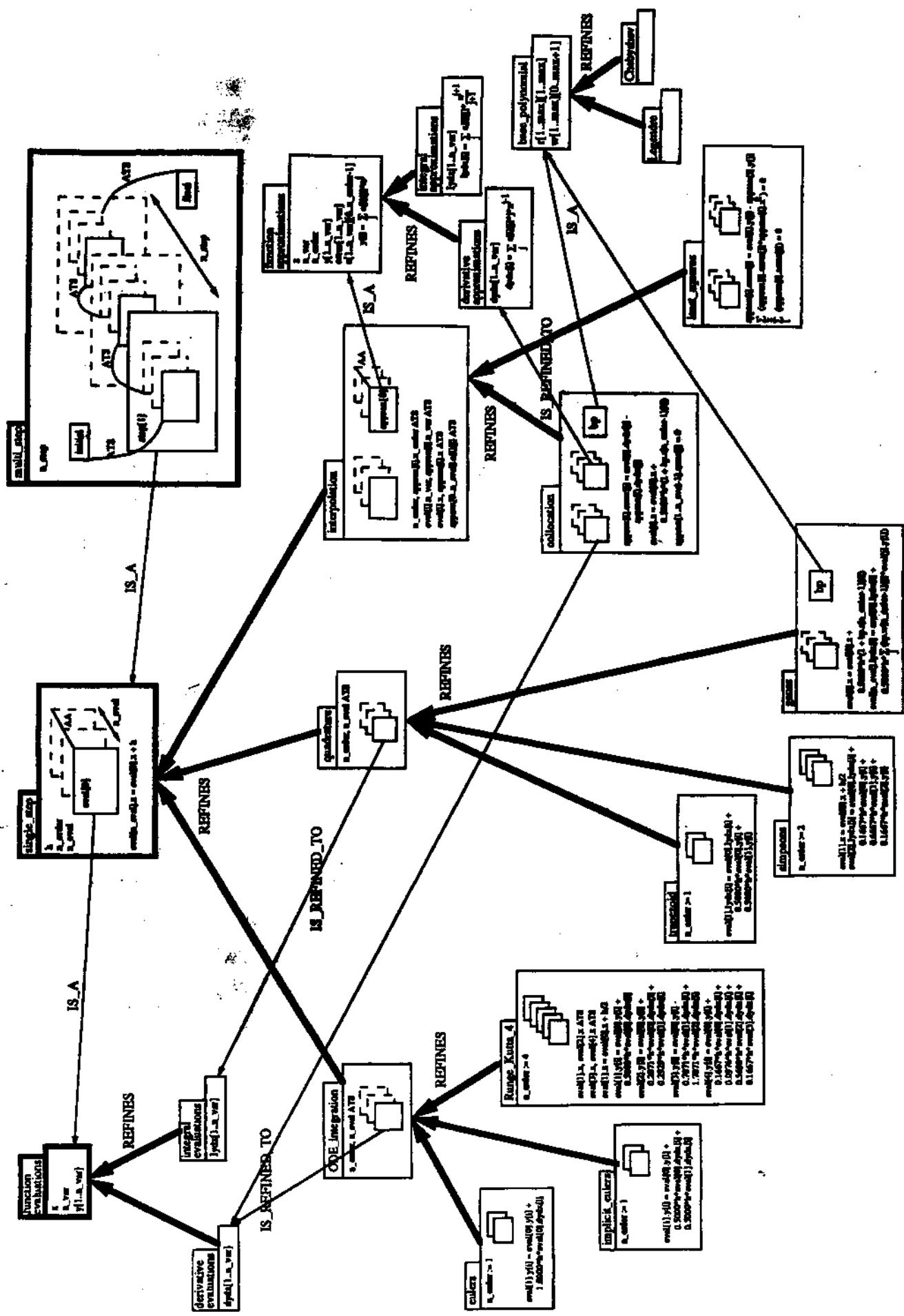
the IS_JREFINEDJIO operator can be used, for example, to upgrade a vapor property MODEL assuming ideal gas behavior to a more rigorous one which utilizes Pitzer's corresponding state correlations. Provision, however, is necessary to have previously declared a pitzer MODEL as a REFTNEment of the ideal gas MODEL.

The final operators to be introduced offer the configurational abstractions of object grouping and merging. The first form of object grouping is done using arrays which allows the creation of a stack of ATOMs or MODELS (currently indexed over *integer* or *string*), all of the same base type. Each element may undergo deferred binding individually. The ARE.ALKE command is used to form object cliques by forcing two or more objects (ATOMS or MODELS) to be of the same type, regardless of the degree to which any one is REFINEd. This is useful in propagating the effect of an IS_REFINED_TO command on a large group of objects and can prevent configurational modeling errors by forcing structural constraints. Finally, the merge operator ARE_THE_SAME goes beyond the ARE_ALIKE command in that it condenses the clique of objects into one unified object. It is the more important configurational operator by providing the means of connectivity necessary in the construction of most engineering models. In addition, size and memory requirements of a simulation can be reduced. For example, by ARE_THE_SAMEing variables themselves rather than equating their values, equations can be eliminated.

3. Building a Mathematics and Thermodynamics Library

The basic framework for solving two-point boundary value problems has been previously given (Piela, *et al.*, 1991). This has been found to provide a suitable framework for most numerical methods involving ODE and quadrature integration, and interpolation. To further illustrate the structure, some visual aids are used in Figure 3-1 (Zaher, 1990). MODELS are depicted as 2-dimensional boxes. Arrays are represented by a stack of overlapping MODELS. The ARE_THE_SAME (ATS) and ARE_ALIKE (AA) commands can be accomplished by connecting the corners of objects with labeling to help clarify which of the two is intended. Instancing is pictorialized by simply overlaying small MODELS (instances) in large ones with side to side connections from the instances to their base types. Specialization is visually formalised as a vertical hierarchy of MODELS connected top to bottom, where the most general base type lies at the top of the tree. An IS_REFINED_TO link can be drawn similar to that of an IS_A with the correct labeling.

Figure 3-1: An ASCE Q structure for mathematics ■



A listing of the ASCEND code corresponding to the building of the mathematical structure is given in Figure A-2 of the Appendix. It is shown that a mathematical support structure can be fabricated by instancing an array of MODELS which themselves contain an array of MODELS. The purpose of making an array of *single_step* MODELS is to break the domain up into one-dimensional finite elements. Each element or *step[]* consists of an array of *function_evaluations* MODELS, the length of which being determined by the accuracy of the element. Connectivity is provided by merging the last *eval[]* instance of each element with the first *eval[]* instance of the element immediately following. For integration (ODE or quadrature), specifications include the user-supplied MODEL containing the equations to be integrated, the number of steps per the integration (*nstep*), the step-wise integration method to be used, and any additional specifications required by the integration method. To solve a system of DAEs (Differential-Algebraic Equations), for example, a user must REFINE the MODEL *derivative_evaluations* to be a more descriptive MODEL where the problem-specific differential and algebraic equations can be entered. Then, within the instanced *multistep*, an IS_REFINED_TO link can be used to upgrade the *eval[]* instances to be of the problem-specific type. This is simplified by ARE_ALIKEing all *eval[]* instances in all *step[]* instances of *multistep*. Then, the upgrade can be accomplished by simply binding only one (namely the *initial*) instance. In addition, the number of elements to be used is specified and each *step[]* instance can be bound to any of the available methods. For interpolation, the order of the approximating function (*n_prder*) to be used for fitting along with the number of elements and the type of interpolating method to be used must be given. If the collocation method (Finlayson, 1980) is selected, DAEs must be provided as needed by an integration method. Otherwise, data interpolation is done by entering the data points directly at the *multistep* instance level.

Arrays also play a major part in the development of a thermodynamics framework to provide a means for implementing non-ideal mixture thermodynamics. Figure 3-2 provides a layout of the structure. First, a library of component data is put together where the physical property constants for each individual species can be made UNIVERSAL, as listed in Figure A-3 for some selected components (Reid, Prausnitz, and Sherwood, 1977). The ASCEND operator, UNIVERSAL, is used for automatically invoking an ARE_THE_SAME command on all instances of the applied type. In Figure A-4, the ASCEND listing corresponding to pure and mult-component thermodynamic property calculations is given (Smith and Van Ness, 1987). For pure component thermodynamic property calculations, some non-ideal methods are given for both the vapor and liquid phases as REFINEMENTs of the general ideal and incompressible MODELS.

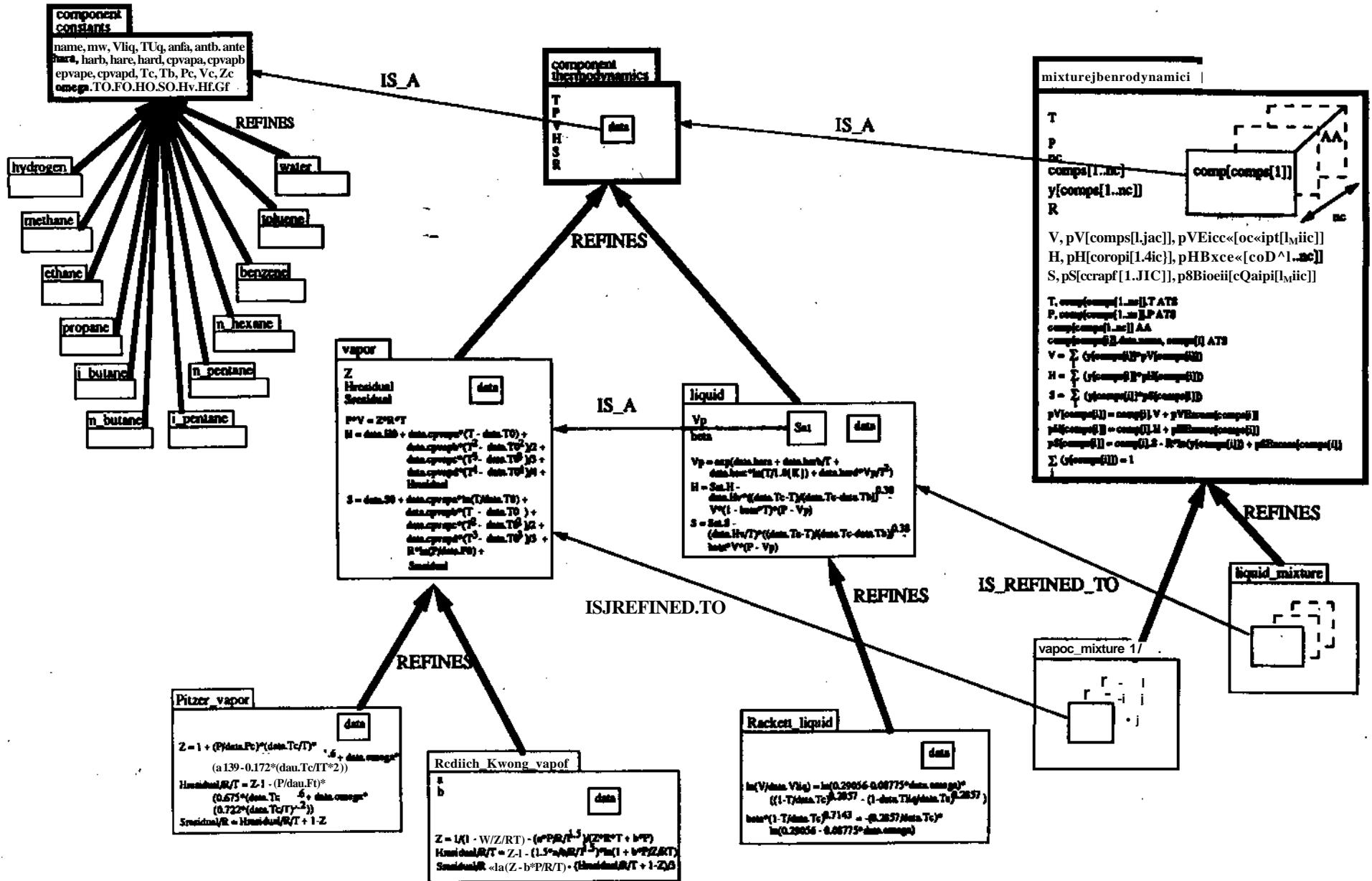


Figure 3-2: An ASCEND structure for thermodynamics modeling

Non-ideal vapor MODELs provide a means with which to estimate compressibility factors other than 1.0 while non-incompressible liquid MODELs provide a means with which to estimate the effect of temperature and pressure on the liquid density. For phases of more than one component only ideal mixture MODELs are available in the current library. It is desired to implement a UNIQUAC REFINEMENT of the *liquid mixture* MODEL as well as a mixing rule REFINEMENT of the *vapor mixture* MODEL in the near future. Although not shown in the pictorial structure, the library also provides stream MODELs for basic flow process simulation. *Single phase stream* MODELs will contain a mixture model to calculate the total (V, //, and S) and partial molar properties ($pV[J]$, $pH[J]$, and $pS[J]$) from the temperature, pressure, and composition using an array of *component^thermodynamics* MODELs. An additional specification for streams is the flow rate which becomes the key attribute for converting all intensive properties to extensive properties. *Multiphase stream* MODELs will require a means to estimate the distribution of components among the phases. This is done using a *phase miscibility* MODEL (or its more REFINED *phase^equilibrium* MODEL) which contains an array of *mixture^thermodynamics* instances, one for each phase present. The miscibility calculation becomes an important part of most equilibrium separation simulations.

4. Example

To demonstrate the use of the above libraries in an engineering application, a chemical reactor will be simulated. The reaction kinetics to be studied is that of vapor phase isomerization of normal pentane in the presence of hydrogen. The isomerization process is widely used among petroleum refineries as a non-additive method for the octane upgrading of hydrocarbon streams. The process explored in this example is shown in Figure 4-1. It was designed to treat approximately 3000 barrels per day of a $75\% nC_5H_{12}/25\% iC_5H_{12}$ feedstock. A vent was to be controlled to maintain a recycle with no less than $9 O^{m0M}$ hydrogen. A zeolite-based platinum catalyst is used to site the isomerization reaction. The reaction is carried out at $525^{\circ}F$ in the presence of 1.25 moles of hydrogen recycle per mole of hydrocarbon feed in order to suppress excessive decomposition of the pentanes to the undesired cracked gases, ethane and propane. The competing reactions are shown in Figure 4-2.

Kinetic models have been proposed to quantify their selectivity (Voorhies and Bryant, 1968). Experimental data have suggested that all species present are assumed to be in adsorption equilibrium and that the adsorption equilibrium constants for all components are equivalent. A unified correlation K_o has been found to be a function of temperature, T, in the range 900-1100 Rankine.

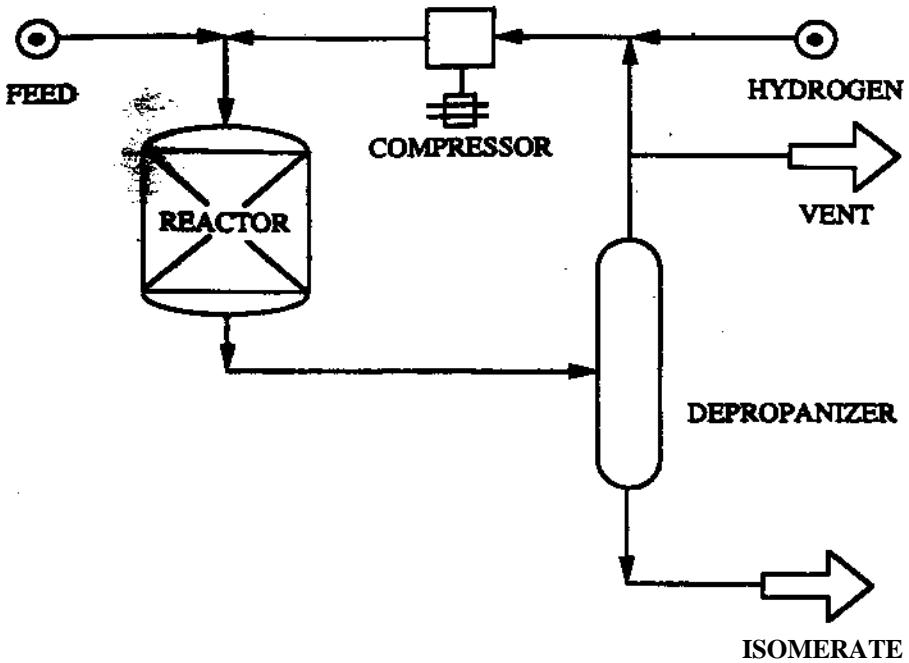


Figure 4-1: Isomerization Process Flow Diagram

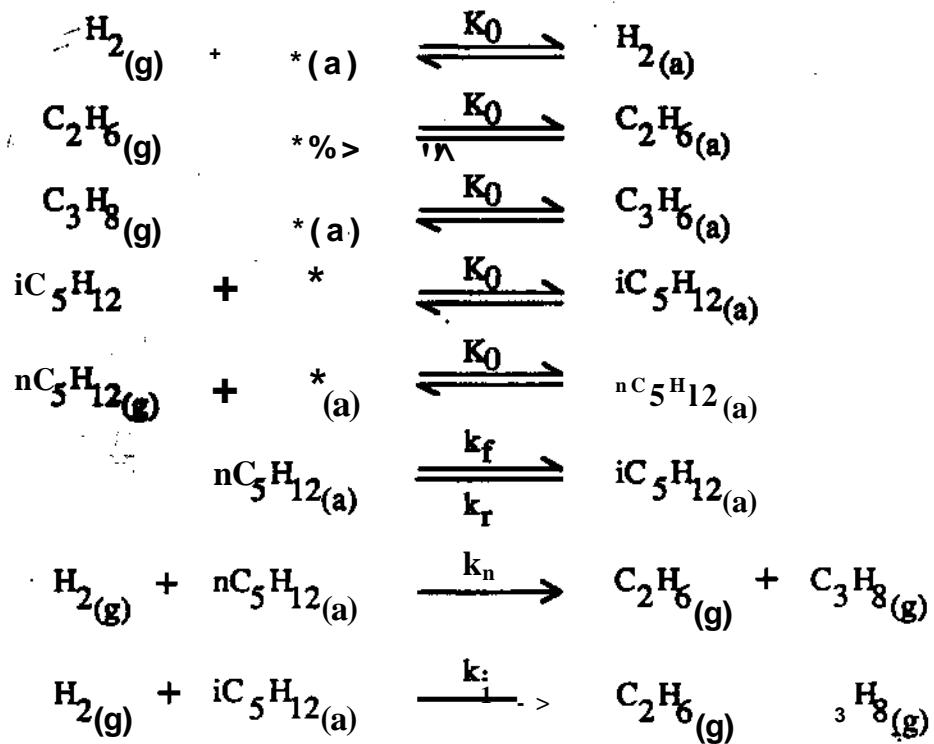


Figure 4-2: Adsorption, Isomerization, and Cracking Reactions

$$K_o \text{ m } 0.63726 - 0.0010452r - 4.2182 \ll 7r^2$$

The rate constants have been expressed in the Arrhenius form as functions of temperature as well.

$$\begin{aligned} k_r &= e^{-\frac{10100}{T}} \quad (\text{in } \text{m} \text{ s}^{-1}) \\ k_t &= \frac{10000}{T} \quad (\text{in } \text{s}^{-1}) \\ k_n &= e^{(7.1 - \frac{11000}{T})} \end{aligned}$$

Finally, the rate equations for components C_2H_6 , iC_5H_{12} , and nC_5H_{12} are given by

$$\begin{aligned} \frac{dy_{C_2H_6}}{dx} &= \frac{i}{W} (k_r + k_t) \\ \frac{dy_{iC_5H_{12}}}{dx} &= \frac{W}{VF(1+K_0P)^2} (k_f y_{nC_5H_{12}} - (k_r + k_i) y_{iC_5H_{12}}) \\ \frac{dy_{nC_5H_{12}}}{dx} &= \frac{W}{VF(1+K_0P)^2} (k_r y_{iC_5H_{12}} - (k_f + k_n) y_{nC_5H_{12}}) \end{aligned}$$

where P is the pressure, W is the mass of catalyst, and V, F , and y are the molar specific volume, molar flow rate, and component mole fraction, respectively, of the gas phase in the reactor.

The code used to formulate this reactor is given in Figure A-5. The thermodynamics library was used to calculate all stream related properties while the mathematics library was used to integrate the above system of differential equations. The pressure drop across the reactor and heat transfer to and from the reactor is neglected to simulate an adiabatic or isenthalpic process for generation of a temperature profile. The *derivative^evaluations* MODEL is first REFINED to the MODEL *kinetics* where an instance of a vapor phase *singlephasejtteam* is created. This provides a molar specific volume and molar specific enthalpy calculation at the reactor conditions for use in the differential equations. Then, a MODEL *reactor* is written which includes an instance of a *multijep* called *profile*. By inspection of the reaction stoichiometry, the molar flow rate is considered constant throughout the reactor. Here, the conditions for isenthalpic and isobaric constant flow is declared. For this example, finite-element orthogonal collocation is used to integrate the rate equations with five elements and two collocation points per element.

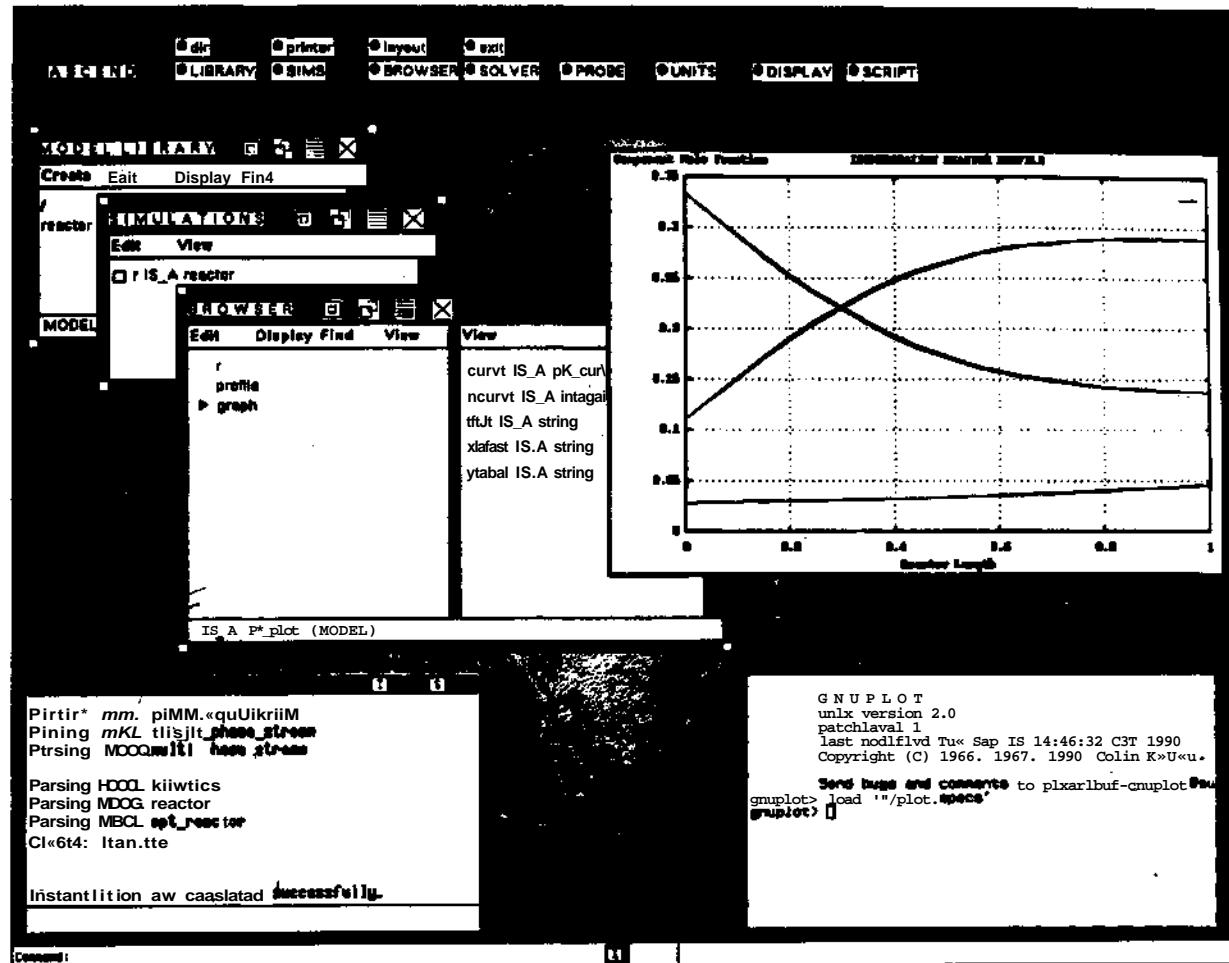


Figure 4-3: The ASCEND environment

A typical illustration of the environment following execution is shown in Figure 4-3. The simulation was performed with both ideal gas and pitzer correlated stream property calculations for comparison. First the ideal case was solved to generate values for all of the system variables. Then the non-ideal case was constructed easily by using the IS_REFINED_TO tool of the environment to upgrade each `comp[]` instance of each stream's mixture MODEL to the type `Pitzerjtapor`. By solving the ideal case first, the ideal solution provided excellent starting values for the non-ideal case. As can be seen in the figure, the ASCEND environment offers plotting programs which can be accessed through the language. The MODELS `singlejstep` and `multi_step` have been designed to interface with the plotters automatically through use of the merge operator. In the graph, there are two plots (ideal and non-ideal) for the mole fractions of each of the components $C_2//_6$, iC_5H_{l2} , and nC_5H_{l2} as they vary throughout the length of the reactor. It can be seen that non-idealities were negligible at the selected temperature and pressure.

5. Conclusions

In conclusion, it was found that the ASCEND environment promoted a very convenient handling of the modeling problem. The language offered decomposition to allow a suitable breakdown of the physics of the problem for easier formulating. Through type inheritance, flexibility was provided to choose a method of integration and a method for estimating thermodynamic properties easily. Data handling was facilitated with the use of nested array structures where the qualifying names for all of the system variables became more literal and easy to associate. Finally, through merging, it was made possible to communicate information across levels of decomposition, such as by merging the temperature of the reactor with that at which the individual component thermodynamic properties within the stream were to be calculated.

It is felt by this author that a more clear understanding of the process is inevitable when the above structure is applied. Should a modification to the reactor process ever need to be implemented, the abstractions offered by the language will minimize the amount of code re-writing by isolating the MODELS that will be affected.

APPENDIX

Figure A-1

34

atoms.lib

A T O M S . L I B

ASCEND definitions for engineering variable types.

Joseph J. Zaher
07/91

(* C O N S T A N T S *)

```
ATOM constant REFINES real;
END constant;

UNIVERSAL ATOM gas_constant REFINES real DIMENSION m*1^2/t^2/mole/tmp
DEFAULT 1.987(cal/gm_mole/K);
END gasconstant;
```

(* T E M P E R A T U R E *)

```
ATOM temperature REFINES generic_real DIMENSION tap
DEFAULT 296.0(K);
lowbound :- 0.0(K);
nominal :- 298.0(K);
display_unit :- (K);
END temperature;
```

(* P R E S S U R E *)

```
ATOM pressure REFINES genericjreal DIMENSION m/l/t^2
DEFAULT 1.0(atm);
lowbound :- 0.0(psla);
nominal :- 1.0(atm);
display_unit :- (psla);
END pressure;
```

(* M A S S / M O L E Q U A N T I T I E S *)

```
ATOM molar_mass REFINES generic_real DIMENSION m/mole
DEFAULT 100.0(g/mole);
low_bound :- 0.0(g/mole);
nominal :- 100.01kg/mole;
display_unit :- 1lbm/lb_mole;
END molar mass;
```

```
AloM m^k HULNLB generic real DlMLNSIOH *
UttaULT 10.0(g/mol);
```

END iSS;

```
ATOM mole REFINES genericjreal DIMENSION mole
DEFAULT 10.0(lb_>ole);
lowbound :- 0.0(lb_mole);
nominal :- 10.0(lbjole);
display_unit :- (lbjole);
END mole;
```

```
ATOM mass_flow REFINES genericjreal DIMENSION m/t
DEFAULT 50(g/s);
low_bound :- 0.01g/s;
nominal :- 100.01g/s;
displayunit :- 1lbm/hour;
END mass_flow;
```

```
ATOM aolar_flow REFINES genericjreal DIMENSION mole/t
DEFAULT 100.0(lb_mole/hour);
lowbound :- 0.0(lbmole/hour);
nominal :- 100.01lbjnole/hour);
displayunit :- {lb_mol<<hour};
END molar_flow;
```

(* V O L U M E Q U A N T I T I E S *)

```
ATOM molar_volume REFINES genericjreal DIMENSION 1^3/mole
DEFAULT 1.0(cm^3/gm_mole);
lowbound :- 0.0(cm^3/gm_mole);
nominal :- 1.0(cm^3/gm_mole);
display.unit :- (cm^3/ga_mole);
END molar_volume;
```

```
ATOM volume REFINES generic_real DIMENSION 1^3
DEFAULT 100.0(ft^3);
low_bound :- 0.0(ft^3);
nominal :- 100.0(ft^3);
display_unit :- (ft^3);
END volume;
```

```
ATOH volume_flow REFINES genericjreal DIMENSION 1^3/t
DEFAULT 100.0(gpm);
low_bound :- 0.0(gpm);
nominal :- 100.0(gpm);
displayjmit :- (ft^3/hour);
END volume_flow;
```

```
ATOM volume_expansivity REFINES generic_real DIMENSION 1/tmp
DEFAULT 0.011/KJ;
low_bound :- 0.0(1/K);
nominal :- 0.00311(1/K);
display_unit :- (1/R);
END volumeJupanslvlyj
```

(* E N T H A L P Y Q U A N T I T I E S *)

```
ATOM molar_enthalpy REFINES generic real DIMENSION m*l^2/t^2/mole
DEFAULT 10000.0(BTU/lb_mole);
nominal :- 10000.01STU/lb_>ole];
display_unit :- (BTU/lb_mole);
END moUrgenthalpy;
```

atoms.lib

```
ATOM enthalpy REFINES genericreal DIMENSION a*1^2/t^2
    DEFAULT 100000.01BTU);
nominal :- 100000.0(BTU);
displayunit :- (BTU|;
END enthalpy;

ATOM enthalpyflow REFINES genericreal DIMENSION B*1^2/t^3
    DEFAULT 100000.0{BTU/hour|;
nominal :- 100000.0(BTU/hour);
displayunit :- (BTU/hour|;
END enthalpy_flow;

(* ENTROPY QUANTITIES *)
-----
ATOM molar_entropy REFINES generlcreal DIMENSION »*1^2/t^2/»ole/tBp
    DEFAULT 100.0{BTU/lb_»ol«/R|;
nominal :- 100.0(BTU/lb_»ole/R);
display_unit :- {BTU/lb_»ol«/R|;
END molarentropy;

ATOM entropy REFINES generlcreal DIMENSION ••1^2/t^2/tmp
    DEFAULT 1000.0{BTU/R|;
nominal :- 1000.0(BTU/R);
displayunit :- {BTU/R|;
END entropy;

ATOM entropy_flow REFINES generlcreal DIMENSION n*1^2/t^3/tmp
    DEFAULT 1000.01BTU/hour/R|;
nominal :- 1000.0(BTU/hour/R);
displayunit :- (BTU/hour/R|;
END entropyflow;

(* DIMENSIONLESS QUANTITIES *)
-----
ATOM factor REFINES genoricreal DIHENSIONLESS
    DEFAULT 1.0;
nominal :- 1.0;
END factor;

ATOM fraction REFINES factor DEFAULT 1.0;
low_bound :- 0.0;
nominal :- 1.0;
upper_bound :- 1.0;
END fraction;
```

Figure A-2

1

M A T H E M A T I C S • L I B

ASCEND structure of a mathematical modal library for numerically integrating function* or systems of DAE'a and performing polynomial interpolation.

Joseph J. Zaher

07/91

```
% Include "/ascend/library/plot.asc"
```

```
( * DATA - BASE MODELS *)
```

```
MODEL baft@polynomial;

rIIInteger1[integer],
w[integer][Integer]           ISA real;

FOR 1:1..6 CREATE
    r[1](1..1) := 0.0;
    »[1][0..1U] := 0.0;
END;
w(0)[0..1) := 1.0;

END basejpolynomial;
```

UNIVERSAL MODEL Legendre REFINES basejpolynomial;

INITIALIZE

```
PROCEDURE values;
(* M-0 *)
-10M0) :- 1.0000000000; -[0](1) :- 1.0000000000;
(* M-1 *)
r1D1) :- 0.0000000000; -[1](0) :- 0.3333333333;
w[1]{1} :- 1.3333333333; w[1](2) :- 0.3333333333;
(* M-2 *)
r[2]U) :- -0.5773502692; -(2)(1) :- 1.0000000000;
r[2](21) :- 0.5773502692; -121(21) :- 1.0000000000;
(* M-3 *)
r{3}{1} :- -0.7745966692; -13)(1) :- 0.5555555556;
r(3)(21) :- 0.0000000000; -(31)(2) :- 0.8888888889;
r{3}{3} :- 0.7745966692; -13)131 :- 0.5555555554;
(* M-4 *)
r{4}{1} :- -0.8611363116; -14)(1) :- 0.3478548451;
r(4)(2) :- -0.3399810436; -(41)(2) :- 0.6521451549;
r(41)(31) :- 0.3399810436; -14113) :- 0.6521451549;
r{4}{4} :- 0.8611363116; -14)141 :- 0.34785484*1;
(* M-* *)
* ON 1 :- 0 9061 /*84*/; -I*ini :- 0.23692688*1;
* 1*Ii\ :- 0.*38*|>*10); -IMN :- 0.4 /8fc28t?0*;
i1*it*1 :- 0.uuoouooouo; «I*I in :- 0.*8MM889j
| t%114| :- 0.*184t5*101; o1*iI4| :- 0.4 J8628470*;
r*1*1* :- 0.90*17984*9; w[1]{1} :- 0.2369268851;
```

mathematicsiiib

```

(* M-6 *)
r[6][1] := -0.9324695142; w[6][1] := 0.1713244924;
r[6][2] := -0.6612093865; w[6][2] := 0.3607615730;
r[6][3] := -0.2386191861; w[6][3] := 0.4679139346;
r[6][4] := 0.2386191861; w[6][4] := 0.4679139346;
r[6][5] := 0.6612093865; w[6][5] := 0.3607615730;
r[6][6] := 0.9324695142; w[6][6] := -0.1713*449*41
END values;

END Legendre;

UNIVERSAL MODEL Chebyshev REFINES base_polynomial;

INITIALIZE
PROCEDURE values;
(* M=0 *)
w(0)[0] := 1.0000000000; w[0][1] := 1.0000000000;
(* M=1 *)
r[1][1] := 0.0000000000; w[1][1] := 3.1415926540;
(* M=2 *)
r[2][1] := -0.7071067810; w[2][1] := 1.1107207350;
r[2][2] := 0.7071067810; -12)(2) := 1.1107207350;
(* M=3 *)
r[3][1] := -0.8660254030; w[3][1] := 0.5235987750;
r[3][2] := 0.0000000000; M(3)(2) := 1.0471975510;
r[3][3] := 0.8660254030; -13)(3) := 0.5235987750;
(* M=4 *)
r[4][1] := -0.9238795320; w[4][1] := 0.3005588660;
r[4][2] := -0.3826834320; M(4)(2) := 0.7256132880;
r[4][3] := 0.3826834320; -14)(3) := 0.7256132680;
r[4][4] := 0.9238795320; w[4][4] := 0.3005588660;
(* M=5 *)
r[5][1] := -0.9510565160; w[5][1] := 0.1941611040;
r[5][2] := -0.5877852520; w(5)(2) := 0.5083203690;
r[5][3] := 0.0000000000; -(5)(3) := 0.6283185300;
r[5][4] := 0.5877852520; tf(5)(4) := 0.5083203690;
r[5][5] := 0.9510565160; w[5][5] := 0.1941611040;
(* M=6 *)
r[6][1] := -0.9659258260; w[6][1] := 0.1355173350;
r[6][2] := +0.7071067810; -16)(2) := 0.3702402450;
r[6][3] := -0.2588190450; -16)(3) := 0.5057575800;
r[6][4] := 0.2588190450; w[6][4] := 0.5057575800;
r[6][5] := 0.7071067810; w[6][5] := 0.3702402450;
r[6][6] := 0.9659258260; -16)(6) := 0.1355173350
END values;

END Chebyshev|)

EVALUATION MODELS
----- *)
MODEL function^evaluations;
n_var# n^eq
», y(integer1
IS_A integer;
n_«q, n^var ARE THE SAME;
IS_A genericreal;

END function^evaluations;

MODEL derlvalve^evaluations REFINES function^evaluations;
dy<Uninteger)
IS_A generic_real;

```

```

END derivatve_evaluation*;
MODEL integralevaluation* REFINES function_evaluation;
    Iydx(Integer)           IS_A genericreal;
END integral_evaluation*;

(* APPROXIMATION MODELS *)
MODEL functionapproxlationa;
    n_var, n_eq, n_order      IS_A integer;
    x, y(Integer)             IS_A genericreal;
    error(Integer)            IS_A genericreal;
    c(Integer)(Integer)       IS_A genericreal;
    y_terr(Integer)(Integer)  IS_A genericreal;
    neq, nyar ARE THE SAME;
    FOR i:1..n_var
    CREATE
        y term[i][0], c[i][0] ARE THE SAME,
        FOR j: 1..n_order-1
        CREATE yterm[i][j] = c[i][j] * x^j;
        END;
        y[i] = SUM(y_term- (0..n_order-1));
    END;

    INITIALIZE
    PROCEDURE fxn_apca;
        x.fixed := true;
        y(l..n_var).fixed := true;
        y_terr(l..nyar).fixed := false;
        error(1..nyar) := 0.0;
        error(1..nyar).fixed := true;
        c(l..n_var)(l..n_order-1) := 0.0;
        c(l..n_var)(l..n_order-1).fixed := true;
        c[1..nyar](0).fixed := false;
    END fxn_apca;
END* function_approximation*;

MODEL derivatve_approxlationa REFINES function_approxlationa|
    dydx[Integer]           IS_A generic_real;
    dydx_terr(Integer)(integer) IS_A generic_real;
    FOR l:1..n_var
    CREATE
        dydx_terr(l)(U, c[l][1]) ARE THE SAME;
        FOR j: 2..n_order-1
        CREATE
            dydx_term[i][j] = c[i][j] * j*x^(j-1);
        END;
        dydx[i] = SUM(dydx_term[i][1..n_order-1]);
    END;
    HUN f*n %inc*, i

```

```

dydx(l..n_var).fixed := false;
dydx_terr(1..n_var)[1..n_order-1].fixed := false;
END darpaca;

END derivatve_approxlations;
MODEL intagral_approxlation* REFINES function_approxlation;
    Iydx(Integer), 10          IS_A generic_real;
    Iydxterm(integer)(integer) IS_A generic_real;
    FOR i:1..n_var
    CREATE
        FOR j: 0..n_order-1
        CREATE
            Iydx_tar>U)JI = c[i][j] * x^(j+1)/(j+1);
        END;
        y(i) = SUM(y_term(l..n_order-1));
        dydx(i) = 8SUM(dydx_tar(l)[1..n_order-1];
        Iydx(i) = 10 * SUM(Iydx_terr(l)(0..n_order-1));
    END;

    INITIALIZE
    PROCEDURE lnt_apca;
        RN fxn_apca;
        Iydx.Tn.var).fixed := false;
        Iydx.Tn.var).fixed := false;
    END lntapeca;
END Integral_approxUationa;

```

(* PROPAGATION MODELS *)

```

MODEL <ingla_atep;
    aval(integer)           IS_A function_evaluation;
    h                      IS_A genericreal;
    n_eval, n_order, n_vat IS_A integer;
    eval(0..n_eval) ARE ALIKE;
    eval(0..n_eval).n_vat ARE THE SAME;
    n_var, <val(0..n_vat).n_var ARE THE SAME;
    eval(n_eval).x = val(0).x * h;
END aingle_atep;

```

(* NUMERICAL INTEGRATION *)

```

MODEL integration REFINES ilngl<_tep;
    n^ordtr, n^aval ARE THE SAME;
END Integration;

(* ODE SYSTEMS *)
MODEL QD integration REFINES integration;

```

```

eval[0.._naval)
IS_REFINED_TO -
darlvatlve->valuations

END ODEintegration;

MODEL <ulir B&lt;u>MM QOI</u>latagration;
n_order :> 1;

FOR 1:1..n_var
CREATE
  eval[1].y[1] = eval[0].y[1] + h*val(01.dydx(1));
END;

INITIALIZE
PROCEDURE apaca;
  RUN eval(0.._naval).apaca;
  evald..n_aval).x.flxad :- falaa;
  eval(1.._aval).y(l..n_var).flxad := falaa;
  eval[n_aval].x.flxad := trua;
  h.flxad := falaa;
END specs;

END eular;

MODEL ImpUclt_aular REFINES ODEintegration;
n_order :> 1;

FOR i:1..n_var
CREATE
  eval[1].y[1] = eval[0].yd) + <h/2.0)*
  (eval(0).dydxdl + eval(1).dydx[1]);
END;

INITIALIZE
PROCEDURE spaca;
  RUN eval(0.._aval).apaca;
  eval[1.._aval].x.flxad := falaa;
  eval[1.._aval].y.ll..n_var).flxad := falaa;
  eval(n_aval).x.flxad := trua;
  h.flxad := falaa;
END apaca;

END implclt_aular;

MODEL Runge_Kutta_4 REFINES ODE_lntagration;
n_ordar :> **

aval1.x, aval(2).x ARE THE SAME;
aval(3).x, aval(4).x ARE THE SAME;
aval11..x - eval[0].x + h/2.0;
FOR 1:1..n_var
CREATE
  eval[1].ydl = eval[0].y[1] +
  eval[0].dydx[1]*h/2.0;
  eval[2].y[1] = eval[0].y[1] +
  (*eval[0].dydx[1]*h/2.071 *
  <v*1 | 0 | .dyd*| I | *U ..; *; *121i | *h;
  .v*1 | M | yll I - <v*1 | 0i | yll I *
  .j->*1 | x | .uya> II*vi. 10tiaill *
  eval[2].dydx[1] = eval[0].dydx[1];
  eval[3].y[1] = eval[1].y[1] - VII**;

```

mathematics4b

```

aval(l..n_aval-l).Iydx(l..n_var).raql{ad :- falsa;
aval(l..n_aval).Iydx(l..n_var).flxsd :- falsa;
aval[n_aval].x.flxsd :- trua;
h.flxad l- falsa; X,
END apaca;

END siapsona;

MODEL Gauaa_quadratura REFINES quadratura;
point*          * A basa_polyoai;
walght (lntagar) (lntagor)      If Ji ganaxlcraal;

FOR 1:1..n_sval-1
CREATE
    aval[1].x - aval(01.x + (h/2.0)*
        (I * points.r(n_ordar-11(1));
END;
FOR 1:1..n_var
CREATE
    FOR j:0..n_aval
    CREATE
        weightUHJ1 - poti>ts.Wn_ordar-1Uj»
            *val{j}.y(l1)
    END;
    *val(n_ml).Iy<U(l) - *val(O).Iydx[l] + (h/2.0)*
        SUMO>ght(l)(0..n_aval));
END;

INITIALIZE
PROCEDURE apacs;
    RM aval[0..n_aval].apaca;
    *val(l..n_aval).x.flxad :- falsa;
    aval(l..n_aval-l).Iydx[l..n_var].raqlra4 :- falsa;
    aval[n_aval].Iydx[l..n_var].ClxMI :- falsa;
    aval(n_aval).x.flxacl :- trua;
    walght[l..n_var](O..n_aval).flxad :- falsa;
    h.flxad :- falsa;
END apaca;

END Gauaa_quadratura;

(* POLYNOMIAL INTERPOLATION

MODEL Intpolation REFINES singla.stap;
approx(lntagar)           ISA function_*p0rods)atlon;
graph                      IS_A pltj>lot7

*pprox(0..n_aval) ARE ALIKE;
n_ordar, approx*(0..n_aval,n^ordar ARE THE SAME;
FOR 1:0..n_aval
CREATE
    •v*1|lj.n_y»r, •pro«(l).n_w«r A*E THE SAME;
    •w*1|l.«, «pro«(l.)» AAE THE SAME;
    k NL;
    l u < i . i _ n w < f
    , MIA! t
        torn ! v   » .ij ti 4
        v MIAII
    IHU,

```

```

END;

(*—————PLOTTING—————)
graph.ncurva, nyar ARE THE SAME;
graph.curva[l..graph.ncurva].npnt, n_aval AaiJTHE SAME;
FOR 1:1..graph.ncurva
CREATE
    FOR j:0..graph.curva[l].npnt
    CREATE
        graph.curva(i).pnt[j].x, aval[j].x ARE THE SAME;
        graph.curvs(U.pnt[j].y, avaUJl:y(i) IN THE SAME;
    END;
END;

END interpolation;

MODEL laaat_squraaa REFINES Interpolation;
moment[lntagor] [intagarUlntagar] ISA ganarlc_raal;

FOR i:0..n_aval
CREATE
    FOR j:1..n_var
    CREATE
        approxdl.errorOl - aval[l].y(j) -
            approxUl.yUl;
    END;
END;

FOR 1:1..n_ordar-1
CREATE
    FOR j:1..n_var
    CREATE
        FOR k:0..n_aval
        CREATE
            moment[i][j][k] - <approx(kl.error(j))*
                Upprox[lc].x)^1;
        END;
        sum(moment(lHj)[0..n_aval]) - 0;
    END;
END;

FOR ltl..n_var
CREATE
    IUMUpprox[0..n_aval].srror[l]) - 0;
END;

INITIALIZE
PROCEDURE spacs;
    RUN avsl(0..n_aval).apaca;
    RUN approx(0..n_aval).fxn_apaca;
    approx[0..n_aval].errorCl..n_var|.fixed :- falsa;
    approx(0..n_aval).y(l..n_var|.fixed :- falsa;
    *omant(l..n_ordar-1H..n_varl[0..n_aval].fi«ad :- falsa;
    approx[0].cll..n_varM0..n_ordar-1l.fUad :- falsa;
    h.flxad :- falsaaj
    END spacs;

END l<<*t_squraaa!

MU>iL collocation RLK1NLS lntsrpulation;

points          ISA basepolyoai;
•val0..n_val| IS_REFINED_TO

```

mathematics.lib

5

61

```

appro*[0..n_eval)           derlivativejevaluations;
                           IS_REFINED_TO ~
                           derlivative_approxlsations;

n_order, n eval ARE_TH* _SAME;
FOR i:0..n~eval   **
CREATE
  FOR j:1..n_var
  CREATE
    eval[i].y[j], appro*U1.y(JI ARE_THE_SAME;
    appro*(1).error[j] - eval[1].dydx[j] -
    appro*U).dydx[j];
  END;
END;
FOR i:1..n_eval-1
CREATE
  eval[i].x - eval(0).x • (h/2.0)*
  (1 + points.r(n_order-1))UUf
END;

INITIALIZE
PROCEDURE Specs;
  RUN eval(O..n_eval).specs;
  RUN appro*[O..n eval).der_specs;
  eval[1..n_eval]y[1..n_var].fixed 2- false;
  eval[O].yll..n_var].fixed f true;
  ev»1(l..n_eval-1).*fixed 1- false;
  h.fixed :- false;
  appro*10..n_eval.errorll..n_varl 2- 0.0;
  approx(l..n_eval-1].errorll..n_varl.fixed :- true;
  approx10..n_eval].error(l..n_varl.fixed :- false;
  appro*[0].ell..n_var)[O..n^order-U.fixed 2- false;
END specs;

END collocation;

(* PROFILE MODELS
----- *)
MODEL multistep;
  n_step, nyar
  •tep(Integer)
  ^initial, final
  graph
    initial, stepll.eval(0) ARE THE SAME;
    n var, «tep(l..n_step).n_var ARE THE SAME;
    final, step[n_step].«val[tep[n_step].n_eval] ARE THE SAME;
    FOR l:1..n~tep-1
      CREATE
        stepli.evaUstepli).n_eval|, stepli+1).eva1[0] ARE THE SAME;
    END;

    ----- PLOTTING -----
    graph heuvvi, n v*r ARE THE SAME;
    graph curvvi 1 i . gi 4pr».nevuv«|.n^nii.
    st, Ap1 :MI >knt.
    ----- PLOTTING -----
    graph curvvi 1 i . gi 4pr».nevuv«|.n^nii.
    st, Ap1 :MI >knt.

step(j).eval[0].x ARE THE SAME;
graph.curve[1].pnt(j-1).y7
step[j].eval[0].y(1) ARE THE SAME;
END;
graph.curvelD.pnt(graph.curveli).npnt).x, final.x ARX_TN1_4I*
graph.curve(l).pnt [graph.curve(i).npnt).
y, final.y[1] ARE THE SAME;
END;

INITIALIZE
PROCEDURE specs;
  RUN step[1..nstep].specs;
  step[1..n_step].eval[0].x.fixed :- false;
  final.x.fixed :- false;
  stepll..n_step].h.fixed :- true;
  initial.x.fixed :- true;
  step[1..n_step].eval[0].y(l..n_var).fixed :- false;
  Initial.yll..n_var].fixed :- true;
END specs;

END Multi_step;

(* M U L T I - S T E P   M E T H O D S
----- *)
MODEL Ada*s_Bashforth_4 REFINES snilti_step;
  step[1..3]                      IS_REFINED_TO
                                   ~ Rungejutta_4;
  step[4..n_step]                  IS_REFINED_TO
                                   ~ OOC_Integration;
  step(4..natep).n_order          :- 1|
  step(4..n_step).n_eval           :- If

  FOR 1:4..n_step
  CREATE
    FOR j:1..step[1].eval(0).n_var
    CREATE
      stepm.evalU1.ylj) - step[ij.eva1 101.y[jl] *
      (step(l).h/24)*
      (SS*step(l).eval(01.dydxlj) -
      59*step[i-1].eval[0].dyd*[j] +
      37*step(l-21.eval(0).dydx(j) -
      t*step(l-3).eval(0).dydx(j));
    END;
  END;

INITIALIZE
PROCEDURE AB4_spacs;
  RUN step(1..3).specs;
  step[1..n_step].eval[0].x.fixed :- false;
  final.x.fixed :- false;
  step[1..n_step].h.fixed :- true;
  initial.x.fixed :- true;
  step[1..n_step].eval[0].y(l..n_var).fixed :- false;
  initial.y[l..n_var].fixed :- true;
END AB4_apecs;

END ka+m$ luahforth_4;

NOOL ISO4« RUINES Aulll «L<p;
IS_A derivative_evaluation*;
```

mathematics.lib

```

nstep          ISA Integer;
y(integer)(integer)   IS_A generic_real;
x(Integer)           IS_A generic_real;
rtol(Integer)         IS_A real;
atol[Integer]         IS_A real;
step(l..n_step)       IS REFINED_TO integration;
stepj1..n_stepj.o_eval  IS ~1;

natep, n_atep ARE_THE_SAME;
d.nvar, n_var ARE_TW_AM<
x(O.natep) # d.x AtE_AUEE;
FOR l:1..d.n_var
CREATE
    y(O..nstepUl|, d.ytl) AtE.AUKE;
END;
FOR l:1..n_atep
CREATE
    FOR j:1..tf.n_var
    CREATE
        step(l).eval(O).y(jl, y(l-1|(j) AAS.THE.SAME;
    END;
    atep(i|.eval(O).x, x(l-1) ARE_THE_tAM<;
END;
FOR k:1..d.n_var
CREATE
    atep[n_atep].eval[l].y[k], y(natep)(k) ARE_THE_SAME;
END;
8tep(n_atep).eval[l].x*.ji[natep] ARE_THE_SAME;

INITIALIZE
    PROCEDURE specs;
        d.x.fixed :- true;
        d.y(l..d.n_eq1.fixed :- true;
    END specs;
END lsode;

```

Figure A-3

componentsJib

```

C O M P O N E N T S . L I B
-----
ASCEND structure of component physical property
constants.

Joseph J. Zaher
07/91
-----)

```

```
llinclude "-/ascend/library/atoms.lib"
```

(GENERAL COMPONENT)

```

MODEL component_constants;
  name IS_A string;
  mw;
  Vliq, Tliq,
  anta, antb, ante,
  hara, harb, hare, hard,
  cpvapa, cpvapb, evpape, cpvapd,
  Tc, Tb, Pc, Vc, Zc, omega,
  TO, PO, HO, SO, Hv, Hf, Gf
  IS_A constant;

  INITIALIZE
    PROCEDURE reference;
    TO := 298.15IK;
    PO := 1.0(atm);
    HO := Hf;
    SO := (Hf - Gf)/TO;
    END reference;
  END component_constants;
-----)

```

(HYDROGEN)

```
UNIVERSAL MODEL Hydrogen REFINES component_constants;
```

```

  name := 'H2';
  INITIALIZE
    PROCEDURE values;
    mw := 2.016|g/gm_mole|;
    anta := 13.6333;
    antb := 164.90|K|;
    ante := 3.19{K};
    hara := 12.050;
    antD := -m .v*iK;
    harD := 0.00M;
    Tliq := 1.048e-17(mmol/g);
    Hv := 1.4411e-16(cal/gm_mole/K);
    cpvapa := 2.215e-16(cal/gm_mole/K^2);
    cpvapd := -1.298e-16(cal/gm_mole/K^4);

```

```

  cpvapd := 1.826e-9{cal/gm_mole/K^4};
  Tc := 33.3|K;
  Tb := 20.37(K);
  PC := 12.8(atm);
  Vc := 65.0(cm^3/gm_mole);
  Zc := 0.305;
  omega := -0.22;
  Hv := 904(J/gm_mole);
  Hf := 0.00(J/gm_mole);
  Gf := 0.00(J/gm_mole);
  Tliq := 20.0(K);
  Vliq := 28.3944(cm^3/9D_sole);
  RN reference;
  END values;
END Hydrogen;
-----)

```

(METHANE)

```
UNIVERSAL MODEL Methane REFINES component_constants;
```

```

  name := 'CH4';
  INITIALIZE
    PROCEDURE values;
    mw := 16.042|g/g»jsole);
    anta := 15.2243;
    antb := 897.841KI;
    ante := -7.16JKW
    hara := 30.715/
    harb := -1300.01(K);
    hare := -2.641;
    hard := 0.442(T2/mmHg);
    cpvapa := 4.59t(cal/gm_mole/K);
    cpvapb := 1.24S<-2(cal/gm_mole/K^2);
    evpape := 2.160e-6(cal/^i_mole/K^3);
    cpvapd := -2.703e-9|cal/gm_mole/K^4);
    Tc := 190.7(K»;
    Tb := 111.67(K»;
    Pc := 45.4{atm};
    Vc := 99.0cm^3/gm_mole);
    Ic := 0.288;
    omega := 0.008;
    Hv := 8180(J/gm_moleW
    Hf := -74840U/ga_moU);
    Gf := -5083S.6(J/gmjROle);
    Tliq := 111.7IK);
    Vliq := 37.7459(cm^3/gB_mole);
    RN reference;
  END values;
END Methane;
-----)

```

(* ETHANE)

```
UNIVERSAL MODEL Ethane REFINES componentj;onstants;
```

```

  name := 'C2H6';
  INITIALIZE
    PROCEDURE values;
    mw := 30.068|g/gm_mole|;

```

components.lib

```

anta :- 15.6637;
antb :- 1511.42UI;
ante :- -17.16|K|;
hara :- 30.759;
harb :- -2464.42(K);
hare :- -3.601;
hard :- 1.073{cm^3/g_mole};

cpvapa :- 1.292<cal/g>>ol</K|;
cpvapb :- 4.254ct-2|eal/gB MBI</K^2>;
epvape :- -1.65?<-S|eai/g>><U/K^3>;
cpvapd :- 2.001<-t|e<1/fBj>U/K^4>;
Tc :- 305.4(K);
Tb :- 104.531(K);
Pc :- 40.2(ata);
Vc :- 140.O(CB^3/ga_aou)f
Zc :- 0.205;
OMga :- 0.0M;
Hv :- 147201J/g<aol>|;
Hf :- -04667|J/g<_oU|;
Gf :- -329201J/ga <_oU|;
Tllq :- 1U.OIKI;""
Vllq :- 54.0606|cs^3/gB>ol*)f
RUM reference;
END valuta;
END Ethan*;
```

(* PROPANE *)
----- */

```

UNIVERSAL MODEL Propar* REFINES component_constants;

naa<                                     :<'C3H0'>;
INITIALIZE
PROCEDURE valua<s;
  w := 44.097(g/gm_mole);
  ant* := 15.7260;
  antb := 1062.46<K>;
  ante := -25.1t(K);
  hara := 43.492;
  harb := -3266.92(10);
  hare := -4.179;
  hard := 1.01{K^2/amHg};
  cpvapa := -1.009(cal/g)>iol</K|;
  cpvapb := 7.315<-2(cal/gM_<ol*/K^2>);
  cpvapo i := -3.709<-5(cal/ga_<U/K^3>);
  cpvapd := 7.670<-9(cal/gBjioU/ir4);
  Tc := 369.9(K);
  Tb := 229.99(KK);
  Pc := 41.9(ata);
  Vc := 203.0(c)>3/9i<_oia>;
  Zc := 0.201;
  oiega := 0.152;
  Hv := 10770(J/gn_mole);
  Hf := -103050|J/gn_>>oU|;
  Cf := -23472U/OB_aol<|;
  Tllq := 231.0IK);
  Vllq := ?5.7600(cm^3/gm_mole);
  RUM (*(*encaz
END valuta;
END HO_fi_Aum.
```

```

(* PROPYLENE *)
----- */
UNIVERSAL MODEL Propyl<n> REFINES coa^>n>nt_constants;
name                                         :- 'C3Hf';

INITIALIZE
PROCEDURE valuta;
  w := 42.001(g/g<_mole);
  anta := 15.7027;
  antb := 1007.531K;
  ante := -26.151K;
  hara := 44.704;
  harb := -3260.31(K);
  hare := -4.379;
  hard := 1.63{K^2/MHg};
  cpvapa := 0.006cal/gBjK>l</K|;
  cpvapb := 5.602<-2(cal/gnjM>l</K^2>);
  epvape := -2.771<-Scal/gm_b U / n i>;
  cpvapd := 5.26to-9(cal/g)>_w<ol*/K^4>));
  Tc := 365.11(K);
  Tb := 225.423(K);
  Pc := 48.6Uta);
  vc := 101.O(oa)>3/g<>ol);
  Zc := 0.275;
  oaaga := 0.140;
  Hv := 19653.92|j/gB>ol<|;
  Hf := 20414.0(J/g<_wolm|t
  Gf := 62710.O(J/e)>_Bol<|;
  Tllq := 223.0IKII
  Vllq := 60.75M(cm^3/gm_mole);
  RUM r<fr<<c<f
END valuaa;
END Propyl<A>;
```

(* i_BUTANE *)
----- */

```

UNIVEMAL MODEL i.ButM* MUIXNES oc<pon#nt_con<taftt>;
name                                         :- 'C4H10';

INITIALIZE
PROCEDURE valU<a>;
  w := 50.124(g/gft_<ol>);
  aota := 15.5301;
  antb := 2032.73IK);
  ante := -33.151K);
  hara := 46.141;
  harb := -3771.21|K|;
  hare t := -4.509;
  hard := 2.571 IT2/>Hg}>;
  cpvapa i := -0.332(cal/ga_aol</K|);
  cpvapb i := 9.109<-2(cal/g<>>U/K^2>);
  cpvapo i := -4.409<-5cal/g<>_ol</K^3>|;
  cpvapd i := 6.915<-9|ca1/g_>>Ou/K^4i;
  Tc := 400. UK);
  Tb := 241.424(K);
  Pc := 3t.o<<t<1;
  Vc := 343.0(cm^3/gm_mole);
  Ic := 0.213;
  omaga i := 0.176;
```

components.lib

```

Hv :- 21190{J/gm_mole};
Hf :- -134500tJ/gm_aoole;
Gf :- -20878.0(J/g<<_mole);
Tllq :- 293.0(K);
Vllq :- 104.3519(c<<3/g>>mcl0);  

RUN reference;  

END values;  

END i_Butane;

```

(* n_BUTANE *)

UNIVERSAL MODEL n_Butane REFINES component constants;

```
name          :- 'nC4H10';
```

INITIALIZE

PROCEDURE values;

```

mw :- 58.124(g/gm_mole);
anta :- 15.6782;
antb :- 2154.90(K);
ante :- -34.42(K1);
hara :- 48.334;
harb :- -4065.571K2;
hare :- -4.781;
hard :- 2.68(K^2/m^2Hg);
cpvapa :- 2.266(cal/g<<joolle/K>>);
cpvapb :- 7.913e-2(cal/gm_mole/K^2);
epvape :- -2.647e-5(cal/gm <<ole/K^3);
cpvapd :- -0.674e-9lcal/g*<<ole/ir4);

```

```

Tc :- 425.17(K);
Tb :- 272.665(K);
Pc :- 37.5(atm);
Vc :- 255.0(cm^3/gm_mole);
Zc :- 0.274;
omega :- 0.193;
Hv :- 22310(J/g<<_mole);
Hf :- -124730(J/gm_mole);
Gf :- -17154(J/gm_mole);
Tllq :- 293.0(K);
Vllq :- 1000.3869(cm^3/gmjnole);
RUN reference;
END values;
END n_Butane;

```

(* PENTANE *)

UNIVERSAL MODEL i_Pentane REFINES component^constants;

NAME

INITIALIZE

```

PROCEDURE values;
DM :- 72.1blig/gm_mole;
anta :- 1b.6338;
anib :- 2348.67JK1;
antc :- -40.0*1|K1;
antd :- >0.420;
ante :- -4.641K1;
ante :- 5.511;
hard :- 3.55(K^2/mmHg);

```

```

cpvapa :- -2.275(cal/gm_mole/K);
cpvapb :- 1.210e-1(cal/gm_mole/K^2);
epvape :- -6.519e-5(cal/gm_mole/K^3);
cpvapd :- 1.367e-8(cal/gm_mole/K^4);
Tc :- 461.0(K);
Tb :- 300.999(K);
Pc :- 33.4(atm);
Vc :- 306.0(cm^3/gm_mole);
Zc :- 0.271;
omega :- 0.227;
Hv :- 24452|J/gm_mole;
Hf :- -155185|J/gm_mole;
Gf :- -1481UJ/gm_mole;
Tllq :- 293.01K);
Vllq :- 116.3726(g/cm^3);
RUN reference;
END values;
END i_Pentane;

```

(* n_PENTANE *)

UNIVERSAL MODEL n_Pentane REFINES component constants;

```
name          :- 'nCSH12';
```

INITIALIZE

PROCEDURE values;

```

mw :- 72.151(g/gm_mole);
anta :- 15.8333;
antb :- 2477.07(K);
ante :- -39.94(k1);
hara :- 52.682;
harb :- -4827.081*;
hare :- -5.313;
hard :- 3.68[K^2/mmHg];
cpvapa :- -0.8*8icool/gm_mole/K);
cpvapb :- 1.1<4e-1(cal/gm_mole/K^2);
epvape :- -6.163e-5(cal/gm_mole/K^3);
cpvapd :- 1.217e-81cal/gm_mole/K^4);
Tc :- 489.8(K);
Tb :- 309.187IK);
Pc :- 33.3(atm);
Vc :- 304.0(cm^3/gm_mole);
Zc :- 0.262;
omega :- 0.251;
Hv :- 25770(J/gm_mole);
Hf :- -146400(J/g<< mole);
Gf :- -8368.0|J/ga_mole);
Tliq i* 293.0(K);
Vllq :- 115.2572{ca^3/gm_mole};  

RUN reference;
END values;
END n_Pentane;

```

(* o^HEXANT *)

UNIVERSAL MOOL n^Meine REFINES component constants;

```
name          :- 'nC6H14';
```

componeats.iib

```

INITIALIZE
  PROCEDURE values;
    » :- 86.178{g/ga >ole};
    anta :- 15.83**;
    antb :- 2*97.SS(K);
    ante :- -49.78(K)#
    hara :- 57.27*/;
    harb :- -5587.42|K)|;
    hare :- -5.885;
    hard :- 4.778|K^2/atfQ);
    cpvapa :- -1.054(cal/ga_Mole/K);
    cpvapb :- 1.390*-1{cal/gii M>1*7K^2};
    epvape :- -7.442e-5(cal/gi_ole/K^3);
    cpvpad :- 1.551e-8{cal/g>j>l</K^4}i
    Tc :- 507.9{K};
    Tb :- 341.887(K);
    Pc :- 29.3{at*};
    Vc :- 370.0(cal^3/g>_ole);
    Zc :- 0.2*0;
    onega :- 0.29*;
    Hv :- 28850EJ/g<>ole);
    Hf :- -1*7200|J/gB>ole);
    Gf :- -251.01J/gp_SK>le|;
    Tllq :- 293.0{K};
    Vllq :- 130.7709(g/cm^3);
    RUN reference;
  END values;
END i\ Hexane;

(* B E N Z E N E
----- *)
UNIVERSAL MODEL Benzene REFINES coapont_constants;
nam* :- 'C6H6';

INITIALIZE
  PROCEDURE values;
    nw :- 78.114(g/gm_nole);
    anta :- 15.9008;
    aneb :- 2788.51(K);
    ante :- -52.36(K);
    hara :- 52.1;
    harb :- -5557.*1{K};
    hare :- -5.072;
    hard :- 3.*1|K^2/m<Hg);
    cpvapa :- -8.101cal/g<jnol</K>;
    cpvapb :- 1.133e-Hcal/g>>ol</K^2|;
    epvape :- -7.20*~-5{cal/gajK>l</K^3};
    cpvpad :- 1.703*~-8{cal/g>>ol</K^4};
    Tc :- 562.2|K);
    Tb :- 353.2521K);
    Pc :- 48.3(atn);
    Vc :- 259.0(cm^3/gm_mole);
    Ic :- 0.271;
    omega :- 0.212;
    Hv :- 115.0(cal/gm_mole);
    II( :- 0.0101;
    II( :- W*fcJ.UtJ/ga_mole);
    Tllq :- 289.0(K);
    Vllq :- 88.3641e-3(g_mole);
    RUN reference;
  END values;

```

END Benzene;

(* T O L U E N E ----- *)

UNIVERSAL MODEL Toluene REFINES component_constant*;

name :- 'C7H8';

INITIALIZE

```

  PROCEDURE value*;
    »w :- 92.13{g/g<>ole};
    anta :- U.0137;
    antb :- 3O9*.S2(K);
    ante :- ~53.*7(K);
    hara :- 5*.785;
    harb :- -*283.50(K);
    hare :- -5.*81;
    hard :- 4.84|K^2/MHg);
    cpvapa :- -5.17|cal/g>>nole/K);
    cpvapb :- 1.224e-1cal/gn>>ole/K^2|;
    epvape I:- -*.*05e-5{cal/ga>>ole/K^3};
    cpvpad :- 1.173e-8(cal/gB_Mole/K^4);
    Tc :- S91.I(K);
    Tb :- 393.77*(K);
    Pc :- 40.6lat*;
    Vc :- 316.0(C^3/g>>ole);
    Zc :- 0.2*4;
    onega !:- 0.257;
    Hv :- 7930.0ical/g>>ole];
    Hf i:- S.003*4(J/ga_sole);
    Gf :- 122005.0(J/g>jsole);
    Tllq :- 293.0(K);
    Vllq :- 10*.2*30(CB^3/gBjK>le);
    RUN reference;
  END values;
END Toluene;
```

(* H A T E R ----- *)

UNIVERSAL MODEL Hater REPINE! component_constants;

name :- 'H2O';

INITIALIZE

```

  PROCEDUR values;
    mmi* 18.00(g/g>>Mill;
    anta t:- 18.303*i;
    aoth !:- 381*.44|M/
    anco :- -46.13|K),
    hara :- 55.33*;
    harb :- -68*9.50(KI/
    hare :- -5.115;
    hard :- 1.05(K^2/>>Mg);
    cpvapa :- 7.701(cal/g<jnole/K);
    cpvapb :- 4.595e-4(cal/gm_mole/K^2);
    epvapt :- 2.821e-4(cal/g>>ole/K^3);
    epvpad :- -0.85%e-9(cal/gm_mole/K^4);
    Tc i:- 447.4JK>;
    Tb :- 373.15|K|-
    fc :- 2U.*(at<i;
```

components.lib

```
Vc :- S6.0(ca^3/gR_<ole>);  
2c :- 0.229;  
omega :- 0.344;  
Hv :- 40650(J/gm_sole);  
Hf :- -241826|J<del>_no U|;  
Gl :- -228614.0(J<del>_noole);  
Tliq :- 293.0(K)f  
Vliq :- 18.0361(oi^3/g>jK>X<);  
RUN reference;  
END values;  
END Water;
```

Figure A-4

T H E R M O D Y N A M I C S . L I B

ASCEND structure of thermodynamic properties for single and multi-phase streams of pure* and mixed components.

Joseph J. Zaher
07/91

%Include *Wascend/library/components.lib*

ATOM specUs REFINES string;
END specie*;

ATOM states REFINES string;
END states;

(• P U R E - C O M P O N E N T M O D E L S

MODEL component_thermodynamics; (* Intensive *)

R	IS_A gas_constant;
data	IS_A component_constants;
T	IS_A temperature;
P	U_A pressure;
V	It_A molar_volume;
H	is_A molar_enthalpy;
S	IS_A molar_entropy;

END component_thermodynamics;

{ V A P O R P H A S E

2	IS_A factor;
HResidual	IS_A molar_enthalpy;
SResidual	IS_A molar_jentropy;

P*V - Z*R*T;
H - data.HO +
 data.cpvapa*(T - data.T0) •
 data.cpvapbT^A2 - data.T0^2)/2 •
 data.cpvapc*(T^3 - data.T0^3)/3 •
 dat.cpvapd*(T^4 - data.T0^4)/4 •
 HR<>sidu4l;

S - dat.e_SO •
 dat.e((P/d4t)^2*I*si(T/d4t *.TO) •
 dat.e((P/d4t)^2*IT - 4*I**e* * TO) •
 dat.e((P/d4t)^2*I/J - Osis.lQ'JII) •
 dat.e((P/d4t)^2*(T^2 J - Oaia.TO'D/1 •
 R^2)n(P/data.T0) •

```

tterraodynamicsiib

      SResidual;

      INITIALIZE
      PROCEDURE specs;
      T.fUed ::= true;
      P.fixed ::= true;
      V.fixed ::= false;
      H.fixed ::= false;
      S.fixed ::= false;
      I ::= 1.0;
      Z.fixed ::= true;
      HResidual ::= 0.0(cal/gm_mole);
      HResidual.fixed ::= true;
      SResidual ::= 0.0(cal/gm_mole/K);
      SResidual.fixed ::= true;
      END specs;

END vapor;

MODEL Redlich_Kwong_vapor REFINES vapor;

a, b           IS_A real;

Z = Z*R*T/(Z*R*T - b*p) - <a*p/R/T^1.5>/U*R*T * b*f;
HResidual/R/T - Z - 1.0 - <1.5*a/b/R/T^1.5>*lnU * o*p/*(a/T)|f
SResidual/R - ln(Z - b*p/R/T) * (HResidual/RVT f 1.0 - 1)/f

INITIALIZE
PROCEDURE RedlichKwong_specs;
a ::= 0.4274i*R^2*data.Tc^2.5/data.Pc;
b ::= 0.0<€*4*R*data.Tc/data.Pc;
t.low_bound ::= data.Zc;
Z.fixed ::= false;
HResidual.fixed ::= false;
SResidual.fixed ::= false;
END Redlich_Kwong_specs;

END RedlichKwong_vapor;

MODEL Pitzer_vapor REFINES vapor;

Z = 1.0 + <p*data.Tc/T/data.Pc)*
    ((0.0t) - 0.422*(data.Tc/T)^1.6) + data.omega*
    (0.139 - 0.172*(data.Tc/T)^4.2));
UResidual/R/T - Z - 1.0 - (P/data.Pc)*
    <(0.875*(data.Tc/T)^2.f) * data.omega*
    (0.722*(data.Tc/T)^5.2)>;
SAresidual/R - HResidual/R/T * (1.0-Z);

INITIALIZE
PROCEDURE Pitzer_specs;
Z.low_bound ::= data.Zc;
Z.fixed i* false;
HResidual.fixed ::= false;
SResidual.fixed ::= false;
END Pitzer.specs;

END Pitzer_vapor;

(* LIQUID PHASE *)
----- */

MODEL liquid REFINES a component th*rdyn*ics;

```

thermodynamicsJib

```

beta IS_A volumeexpansivity;
VP IS_A pressure;
Sat IS_A vapor;

T, Sat.T ARE THE_M9tj
VP, Sat.P ARE THE_JAM;
data, Sat.data" AUJrMSJMME;
VP - 1.0(mmHg)*exp(data.hara +
    data.harb/T +
    data.harc*ln(T/(KJ) +
    data.hard*VP/T^2)j

H - Sat.H -
    data.HV((data.Tc-T)/(data.Tc-data.Tb)**0.38 +
    VM1.0 - beta*T)*(*- VP);
S - Sat.S -
    (data.Hv/T)*((data.Tc-T)/(data.Tc-data.Tb))**0.38 -
    beta*V»(p - VP);

INITIALIZE
PROCEDURE specs;
    RUN Sat.specs;
    T.fixed :- true;
    P.fixed :- true;
    V :- data.Vllq;
    v.fixed :- true;
    H.fixed :- false;
    S.fixed :- false;
    beta :- 0.0U/K;
    beta.fixed :- true;
    VP :- 1.0(mmHg)*
        exp(data.antA - data.antB/(T * data.antC));
    VP.fixed :- false;
END specs;

END liquid;

MODEL Rackett_lliquid REFINES liquid;

ln(V/data.Vllq) -
    ln(0.29056-0.08775*data.omega)*
    ((1 - T/data.Tc)*0.2857 -
    (1 - data.Tliq/data.Tc)**0.2857);
betaM1 - T7data.TcrO.7143 - (0.2857/data.Tc) *
    ln(0.29056-0.08775*data.omega);

INITIALIZE
PROCEDURE Rackett_specs;
    V.fixed :- false;
    beta.fixed :- false;
END Rackett_specs;

END Rackett_liquid;

f M I X T U B t M O D E L S
----- *)
Model mixture_thermodynamics;

comp(specs|) IS_A component_constants;
IS_A component_thermodynamics;
IS_A molar_enthalpy;
IS_A molar_entropy;
IS_A gas_constant;
IS_A molar_volum;
IS_A molar_enthalpy;
IS_A mol<r_eiitropy;

```

```

IS_A fraction;
IS_A cocomponent_constants;
IS_A component_thermodynamics;
IS_A molar_volume;
IS_A molar_enthalpy;
IS_A molar_entropy;
IS_A gas_constant;
IS_A molar_volum;
IS_A molar_enthalpy;
IS_A mol<r_eiitropy;

T, comp(comps(1..nc)).T ARE THE_SAME;
P, comp(comps(1..nc)).P ARE THE_SAME;
comp(comps(1..nc)| ARE_ALIKE;
FOR 1:1..nc
CREATE
    comp(comps(i| j).data.name, comps(1) ARE THE_SAME;
END;
V - SUM(tV[comps(1..nc)]);
H - SUM(tH[comps(1..nc)]);
S - SUM(ts[comps(1..nc)]);
FOR 1:comps(1..no)
CREATE
    data(1), comp(i).data ARE THE_SAME;
    tv(1) - y(1)*pv(1);
    th(1) - y(1)*ph(1);
    ts(1) - y(1)*ps(1);
    pv(i) - comp(i).V * pVExcess(i);
    ph(i) - compdJ.H * pHExcess(i);
    ps(i) - comp(i).S - R*ln(y(U) * pSEExcess(i));
END;
SUM(y(comps(1..nc)J) - 1.0;

END mixture_thermodynamics;

MOEL vapor_mixture REFINES mixture_thermodynamics;

comp(comps(1..nc)| IS_REFINED_TO vapor;

INITIALIZE
PROCEDURE specs;
    RUN comp(comps(1..nc)).specs;
    T.fixed :- true;
    P.fixed :- true;
    y(compa(1..nc)).fixed :- true;
    y(comps(nc)).fixed :- false;
    V.fixed :- false;
    H.fixed :- false;
    S.fixed :- false;
    pV(comps(1..nc)).fixed :- false;
    pH(comps(1..nc)).fixed :- false;
    pf(comps[1..nc|j].fixed :- false;
    pVExcess[comps(1..nc)J] :- 0.0(ft^3/lb_moU);
    pVExcess(comps(1..nc)j).fixed :- true;
    pHExcess(comps(1..nc).noll :- 0.0(BTU/lb_mole);
    pHExcess[comps(1..nc|J)].fixed :- true;
    pSEExcess(comps(1..nc)) :- 0.0(BTU/lb_mole/R);
    pSEExcess(comps(1..nc)|).fixed :- true;
END specs;
END v*por_mixture;

MILL liquid_alliuit RLNLS mixture_thermodynamics;
co<plcomps(1..nc)| IS_REFINED_TO liquid;

```

(jbermodynamicsii)

```

INITIALIZE
  PROCEDURE specs;
    RUN comp(comps[1..nc]l.specs;
    T.fixed :- true;
    P.fixed :- tn»f
    y[coaps[1..no)].fixed :- true;
    y(comps(nc)).fixed x- falsef-
    v.fixed :- false;
    H.fixed :- false;
    s.fixed :- false;
    pv(comps(1..nc]).fixed :- false;
    pH[comps(1..nc)].fixed :- false;
    pS(comps(1..nc)).fixed :- false;
    pVExcess(comps[1..nc]II I- 0.0(ft^3/lb_mol»);
    pVExcess(comps[1..nc]).fixed :- true;**
    pHExcess(comps(1..nc)) :- 0.0(BTU/lb_mole);
    pHExcess(comps(1..nc).fixed :- true?
    pSExcess(comps(1..nc)) :- 0.0(BTU/lbjsole/R);
    pSExcess(compsII..ncj).fixed :- truef
  END specs;
END liquiddmixture;

(* HISCIBILITY MODEL

MODEL phasejilsclblly;
T IS_JI temperature;
P IS_JI pressure;
Mix(states) IS_JI mixture_thermodynamcs;
alpha(species)(states)[states] IS_JI factor;""
ave_alpha[states][states] IS_JI factor;
nc, np IS_JI integer;
comps[Integer] IS_JI species;
phases(Integer) IS_JI states;
data(species) IS_JI componentconstants;

T, Mix[phases(1..np)].T ARE THE SAME;
P, Mix[phases(1..np)].P ARE THE SAME;
nc, Mix[phases(1..np)].nc ARE THE SAME;
FOR 1; 1..nc
CREATE
  comps(i), Mix[phasesII..np]l.comps[i] ARE THE SAME;
END;
FOR 1: comps(1..nc)
CREATE
  data(1), Mix[phases[1..np]].data(1) ARC^THE.SAHE;
  FOR j: phases(2..np)
CREATE
  ave_alpha[phases(1)](j)*Mlx[alpha]as[1]).y(1) -
  alpha(1)(phases(1))(j)*Mix().yd;
  END;
END;
INITIALIZE
  PROCEDURE specs;
  H :: Mix[phases(1)].opt i MIXX «;
  alpha(i).fixed :- true;
  y(i).fixed :- true;
  v(i).fixed :- true;
  H fixed :- true;
  alpha(p)1..nc| II phases[1..np].fixed :- true;
  ave_alpha[phases(1)](j).fixed :- false;
END specs;

END phase jilsclblly;

(* PHASE EQUILIBRIUM
----- */

MODEL phase_equllbrum REFINES phase_mlsclblly;
FOR 1: comps(1..nc)
CREATE
  FOR j: phases(2..np)
CREATE
  Mix(phases[1]).pH(i) - T*Mix[phases(1J).pi(1) -
  Mlx(j).pH(l) - T*Mlx(j).pS(l);
END;
END;

INITIALIZE
  PROCEDURE equll_specs;
  T.fixed :- false;
  alpha(comps(1..nc)) (phases(1)) [phases(2..np)] .fU«d :- false;
  ave_alpha(phases(1)l[phases(2..np))] :- 1.0;
  ave_alpha(phases(1)l[phases(2..np)].fixed :- true;
END equll_specs;

END phase_equllbrum;

(* STREAM MODELS
-----

MODEL singlephase_strea«;
T IS_JI temperature;
IS_JI pressure;
IS_JI molar_flow;
IS_JI Integer;
IS_JI species;
is""ifc fraction;
IS_JI componentconstants;
IS_JI mixture_thermodynamics;
IS_JI molar_volume;
IS_JI molar_enthalpy;
IS_JI molar_entropy;

T, Mix.T ARE THE SAME;
P, Mix.P ARE THE SAME;
nc, Mix.no ARE THE SAME;
FOR 1:1..no
CREATE
  comps(l), Mix.comps(l) ARE THE SAME;
  data(oompa(l)), Mix.data(oomps(i)) ARE THE SAME;
  ylcompadl), Mix.y(compa(lJ) ARE THE SAME;
END;
V, Mix.V ARC THE SAME;
H, Mix.N ARC THE SAME;
J, Mix.S A[C THE SAME;

INITIALIZE
  PKOCCUM <pc>s;
  HUMMix.specs;
```

thermodynamicsJib

```

T.fixed :- true;
P.fixed :- true;
F.fixed :- true;
y(comp<[1..nc]).fixed :- true;
ylcomps(nc).fixed :- false;
V.fixed :- false;
H.fixed :- false;
S.fixed :- false;
END specs;
END single_phase_stream;

MODEL multlphasesstream;

T IS_Ji temperature;
P IS_Ji pressure;
F IS_Ji aolar_flow;
nc, np IS_Ji integer;
comps(Integer) IS_Ji species;
phases(Integer) IS_Ji states;
y[specie*], tylspecies)[states] IS_Ji species;
data(species) IS_Ji states;
Mlx(states) IS_Ji component-constants;
Hlsc .18*IS_Ji >xture_ther>odyna>lcs;
phi(states) IS_Ji phase-<>isclbility;
V, tV(states) IS_Ji fraction;
H, tH(states) IS_Ji <>olar_volumse;
S, tS(atates) IS_Ji solarentropy;

T, Mlsc.T ARE_THE_SAME;
?, Mlsc.P ARE_THE_SAME;
nc, Hlsc.nc ARE_THE_SAME;
np, Mlsc,np ARE_THE_SAME;
FOR l:1..nc
CREATE
    compsli), Misc.compsli) ARE_THE_SAME;
END;
FOR l:comps[1..nc)
CREATE
    data[l], Mlsc.datall) ARE_THE_SAME;
    y(i) - SUM(ty(i)[phases(l..np)]);
    FOR ) :phases(l..np)
    CREATE
        tyllljl - phl(jl*Mix[j].y(l);
    END;
END;
FOR j:1..np
CREATE
    phases(j), Misc.phases!j) ARE_THE_SAME;
END;
FOR j:phases[1..np)
CREATE
    Mix[j], Misc.Mlx(j) ARE THE SAME;
    (V|jl) - phl(j)*Mix[j].V;
    (H|jl) - phl(j)*Mixni.H;
    (S|jl) - phl(j)*Mixlj.S;
    INU;
    V - SUM (t V|jl) *Mix[j].V;
    M - SUM (t H|jl) *Mix[j].H;
    S - SUM (t S|jl) *Mixlj.S;
    H - SUM (t H|jl) *Mixlj.H;
    Mix - SUM (t Mix|jl) *Mixlj;
    END;
END;

```

```

T.fixed :- true;
P.fixed :- true;
F.fixed :- true;
y(coaps[1..ncl].fixed :- true;
y[co<ps[nc]).fixed :- false;
FOR l:co<ps[1..nc) 00
    Mlx[phases[1..np)].y[i] := ydl;
    Mix(phases[1..np).y(1).fixed :- false;
END;
phi(phases[1..np]).fixed :- false;
V.fixed :- false;
H.fixed :- false;
S.fixed :- false;
END specs;
END multl_phase_stream;

```

Figure A-5

I S O M . A S C

**ASEND structure for the modeling
of a pentane isoawrlatloo prooess.**

Joseph J. Zaher
07/91

```

include "--/ascend/library/mathematics.lib"
%Include "--/ascend/library/thermodynamics.lib"

(* RATE EQUATIONS *)
----- */

MODEL kinetics REFINES derivative_evaluations;

catalyst
s
t
p
n_var
s_nc
s_comps[1]
s_comps[2]
s_comps[3]
s_comps[4]
s_comps[5]
s_data('H2'1)
s_data('C2H6')
s_data('C3H8'1)
s_data('iC5H12')
s_data('nC5H12')
s_Mix
z

t, s.t ARE THE SAME;
p, s.p ARE THE SAME;
y(Dx ``.y('C2H6') ARE THE SAME;
y121, s.y['iC5H12'] ARE THE SAME;
y(3) ..y('nC5H12') ARE THE SAME;
s.y['C2H6M', a.y['C3Hr']] ARE THE SAME;

dydx111 = 1.0(cn^3/g/s)
*(catalyst/s.V/s.F)
M<>p(7.3 - 10000.0(R)/T)*s.y['iC5H12'] *
e<>(7.1 - 11000.01R/T)*s.y['nC5H12'] /
(1.0 + 1.0/l/p<>la)*p
*(0.63726-0.001045211/R)*t+4.2182e-7(l/R^2)*t^2)^2;
dydx121 = 1.0(cn^3/g/s)
*(catalyst/s.V/s.F)
*(exp(9.9 - 8500.0(R)/T)*s.y['iC5H12'] -
exp(7.1 - 11000.0(R)/T)*s.y['nC5H12']) *
/(1.0 + 1.0(l/p)<>la)*p
*(0.63726-0.001045211/R)*t+4.2182e-7(l/R^2)*t^2)^2;

```

isom.asc

```

dydx[3] = 1.0(coi^3/g/a)
*(catalyst/s.V/s.F)
M<>p(10.7 - 10100.0(R)/T)*s.y['iC5H12'] -
exp(9.9 - 8500.0(R)/T)*s.y['nC5H12'] -
exp(7.1 - 11000.0(R)/T)*s.y['nC5H12']U /
(1.0 + 1.0{l/p}<>la)*p
*(0.63726-0.001045211/R)*t+4.2182e-7(l/R^2)*t^2)^2;

```

INITIALIZE

```

PROCEDURE values;
RUN s.data('H2').values;
RUN s.data('C2H6').values;
RUN s.data['C3H8'].values;
RUN s.data['iC5H12'].values;
RUN s.data['nC5H12'].values;
catalyst := 10000.0/lbB;
s.t := 985.0(R);
s.p := 280.0(psia);
s.F := U25.0[lb_BK>le/hour];
s.y['H2'] := 0.50;
s.y['C2H6'] := 0.03;
s.y['C3H8'] := 0.03;
s.y['iC5H12'] := 0.11;
s.y['nC5H12'] := 0.33;
dyds(1).nominal := 0.001;
dydt(2).nominal := 0.01;
dydx(3).nominal := 0.01;

```

END values;

PROCEDURE specs;

```

RUN s.specs;
catalyst.fixed := true;
s.y['H2'].CiJied := false;
s.y['C2H6']*ClJid := true;
s.y['C3H8'].fU<>d := true;
s.y['iC5H12'].fU<>d := true;
s.y['nC5H12'].fU<>d := true;
x.fixed := true;
dydx[1..n_var].fixed := false;

```

END specs;

END kinetics;

(* REACTOR OPERATION *)

MODEL reactor;

```

in, out
profile
ratio
profile.n_step
profHe.step[1..profile.n_step].reorder
profile.stepd ..profile.n_atepl
profile.itipil..profile.n_step|.points
profile.initial

```

IS_A singlej>hase_strea<;

IS_A multl_step;

IS_A factor;

:= 5;

:= 3;

ISREFINED_TO collocation;

ISREFINEDTO Legendre;

IS_REFINED_TO kinetics;

```

in, profile.initial.s ARE THE SAME;
but, profile.final.s ARE THE SAME;

```

»O* is1..profile.step

CREATE

```

FOR j:0..profile.step(1).n_ord<r-1

```

isom.asc

```

CREATE
    profile.stepI.U.eval[j].s.H, profile.stepU.eval[j+1].s.K
    ARETHESAME;
    proflit.step(U.eval(j).s.F, profile.step[ll.eval[j+H.s.F
    ARE_THE_SAME;
    profile.stepI.li).eval[j].s.p, profile.stepI.U.eval[j+U.s.p
    ARE_THE_JAMB;
    END;
END;
ln.y['C2H6'l*(ratio + 1) - 0.05*ratio;
in.yC 1C5H12)* (ratio + 1) - 0.25;
in.yl'nC5H12'l*(ratio + 1) - 0.75j
ln.F - 500.01lb_M>l<hour)*(ratio + 1);

INITIALIZE
    PROCEDURE values;
        FOR 1:1..profile.nstap DO
            RUN profile.stepI.li.point*.value*;
            FOR }:0..profile.<step(1).n_ord<r DO
                RUN profile.<step111.availjl.value<;
            END;
        END;
        ln.T :- 985.0(R);
        ratio :- 1.25;
        profile.Initial.x :- 0.00;
        profile.8step[2].eval[0].K :- 0.20;
        profile.step[31].eval[0].x :- 0.40;
        profile.8step[4].eval[0].x :- 0.60;
        profile.<step(5).eval(0).a :- 0.80;
        profile.final.x :- 1.00;
    END values;

PROCEDURE specs;
    RUN In.specs;
    RUN out.specs;
    RUN profile.specs;
    FOR 1:1..profile.n_<step DO
        FOR j:0..profile.stepD].n_order DO
            profile.stepU.eval[j].s.t.fixed :- false;
            profile.step(U.eval[j].s.p.fixed :- false;
        END;
    END;
    ln.T.fixed :- true;
    In.P.fixed :- true;
    ln.F.fixed :- false;
    ln.y(ln.cosips[1..ln.nc]).fixed :- false;
    out.t.fixed :- false;
    out.y(out.co*ps[1..out.nc]).fixed :- false;
    ratio.fixed :- true;
    profile.initial.x.fixed :- true;
    profile.step[2].eval[0].x.fixed :- true;
    profile.step(3).eval[0].x.fixed :- true;
    profile.8step[4].eval[0].x.fixed :- true;
    profile.stepI5).eval(0).x.fixed :- true;
    profile.flinal.x.fixed :- true;
    profile.stepI1..5).h.fixed :- false;
    tIND specs;
LNU & exit();

```

References

- B. A^Finlayson. (1980). *Nonlinear Analysis in Chemical Engineering*. McGraw-Hill Book Company.
- P. C Piela, T. G. Eppeniy, K. M. Westeiberg, and A. W. Wcsterberg. (1991). ASCEND: An Object-Oriented Computer Environment for Modeling and Analysis: The Modeling Language. *Comput. Chem. Engng.*, 75(1), 53-72.
- R. C. Reid, J. M. Prausnitz, and T. K. Sherwood. (1977). *The Properties of Gases and liquids*. McGraw-Hill Book Company.
- J. M. Smith and H. C. Van Ness. (1987). *Introduction to Chemical Engineering Thermodynamics, 4th Edition*. McGraw-Hill Book Company.
- A. Vooihies and P. Bryant (1968). Hydroisomerizatkm of Normal Pentane over a Zeolite Catalyst *AIChEJournal*, 14(6), 852-856.
- A. W. Westeiberg, P. C. Piela, R. D. McKelvey, and T. G. Epperly. (July 1991). The ASCEND Modeling Environment and Its Implications . Paper presented at the 4th International Symposium meeting of the Process Systems Engineering.
- J. J. Zaher. (October 1990). Graphical Representations of the ASCEND Modeling Language . *Paibr* presented at the 12th Annual Chemical Engineering Symposium meeting of the Carnegie Mellon University ChEGSA.