Computer-Aided Thermodynamics Modeling of a Pure Substance

by

Jose L. Perez

B.S.M.E., University of Puerto Rico, Mayagüez (1992)

Submitted to the Department of Mechanical Engineering in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE in MECHANICAL ENGINEERING

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1995

© 1995 Massachusetts Institute of Technology All rights reserved.

Signature of Author

Department of Mechanical Engineering

May 12, 1995

Certified by

Professor Joseph L. Smith, Jr.
Thesis Supervisor

Accepted by

Professor Ain A. Sonin
Chairman, Graduate Committee
Department of Mechanical Engineering

AUG 31 1995

LIBRARIES
Computer-Aided Thermodynamics Modeling of a Pure Substance

by

Jose L. Perez

Submitted of the Department of Mechanical Engineering in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

Abstract

This thesis consists of three parts. First a Computer-Aided Thermodynamic (CAT) methodology is developed that enables modeling of a Pure Substance. The second part of the thesis develops the corresponding cT-based software for the modeling of steam as a Pure Substance. Finally, in part three the cT source code is presented, and some trial problems are analyzed using the software.

The foundation for the CAT methodology was developed here at the Massachusetts Institute of Technology (MIT) by Professor Joseph L. Smith, Jr. and Dr. Gilberto Russo in Russo's 1987 doctoral thesis A New Methodology of Computer-Aided Thermodynamics. The resultant CAT algorithm and software was later refined and incorporated into MIT’s Athena Project by Chun-On Cheng, graduate student at MIT. Mr. Cheng used C as programming language in the X-Windows environment. More recently, Oscar C. Yeh, working as an undergraduate research student at MIT translated the CAT source code into the cT language. The choice of cT as the source code language means that the CAT program is now portable across computer systems such as UNIX, Microsoft-DOS, and Apple Macintosh.

This thesis, built on the aforementioned work, presents the algorithm and corresponding source code for analysis of steam as a Pure Substance within the framework of the CAT methodology. The result is the addition of a new Steam Element to the array of choices previously available in the CAT program. The Steam Element is capable of modeling the behavior of steam at its liquid, liquid-vapor, and vapor states. Although the basic algorithm can be used for any substance that typically undergoes phase changes, the current code runs for steam, based on the equations of state for steam developed in Steam Tables, by J. Keenan, et al. (1969).

Thesis Supervisor: Dr. Joseph L. Smith, Jr.
Title: Ford Professor of Mechanical Engineering
To my wife, Carmen Enid, my mother, Alba Nydia and my soon-to-be-born daughter,

Laura Enid.
Acknowledgments

I would like to express my gratitude to the people who helped, directly or indirectly, to the fruition of this thesis. Among them, Professor Joseph L. Smith, Jr., for his insight and assistance in helping me understand the CAT methodology and its concepts. Also, Professor Adeboyje (Ade) Oni, for his enthusiasm in the initial stages of this project.

I would also like to thank the people who I call “the computer guys”, Tom Cheng and Oscar Yeh, who programmed CAT in the C and cT languages, respectively. They were always available when I required their technical assistance.

For the emotional assistance throughout my graduate school years I have to thank my wife, Carmen Enid. She celebrated my good moments and consoled me during the bad times. Te quiero mucho, Dindita. My mother, Alba Nydia, was also a source of emotional (and sometimes financial) support during my undergraduate years at Mayagüez. Gracias, Mami.

Finally, I wish to acknowledge the financial support provided by General Electric for my first year of studies at MIT; the support from the Naval Undersea Warfare Center (NAVUNSEAWARCCENTDIV) for my third semester; and the financial assistance arranged for my last semester at MIT by Margo Tyler, Assistant Dean at the Graduate School Office.
Table of Contents

Abstract ........................................................................................................................... 3
Dedication ....................................................................................................................... 5
Acknowledgments ........................................................................................................... 7
Table of Contents ............................................................................................................ 9
List of Figures ................................................................................................................. 11

Part I: Conceptualization and Formulation

1 Preamble
   1.1 Background ........................................................................................................... 14
   1.2 Basis for Method ......................................................................................... 15
   1.3 cT-Based Application Development ......................................................... 16

2 CAT Methodology
   2.1 Overview .......................................................................................................... 17
   2.2 Modeling Concept ...................................................................................... 17
   2.3 CAT Elements ........................................................................................... 18
   2.4 Topology Matrices .................................................................................. 19
   2.5 Derivation of System Equations .............................................................. 21
   2.6 Stiffness Matrix Elements ................................................................. 22
   2.7 Reversible Process ................................................................................. 24

3 Pure Substance Element Modeling
   3.1 Overview ........................................................................................................ 27
   3.2 Equations for Water Vapor and Liquid Regions ..................................... 27
   3.3 Equations for Water Liquid-Vapor Mixture (Saturation) ....................... 32
   3.4 Stiffness Matrix Expressions for Water Vapor and Liquid Regions ...... 34
   3.5 Stiffness Matrix Expressions for Liquid-Vapor Mixture (Saturation) .... 35
   3.6 Behavior of the Pressure Function .......................................................... 38
Part II: Software Development

4 Introduction
   4.1 cT as the Development Language ................................................. 44

5 The CAT Software
   5.1 Organization ................................................................................. 45
   5.2 Mesh Creation ................................................................................ 45
   5.3 Input Routines ............................................................................... 46
   5.4 Computational Algorithm ............................................................. 48
   5.5 Solution Presentation .................................................................... 51
   5.6 Relations Database ....................................................................... 52
   5.7 Graphical Interface Routines .......................................................... 52

6 Steam Element Software Development
   6.1 Input Routine ............................................................................... 53
      6.1.1 Known Quality ................................................................. 53
      6.1.2 Unknown Quality .............................................................. 54
   6.2 Steam Functions .......................................................................... 58

7 Conclusion
   7.1 Potential Extension ..................................................................... 63
   7.2 Closure ......................................................................................... 64

Part III: Appendices

Appendix A Running CAT
   A.1 Using cT and CAT.t ...................................................................... 66

Appendix B Sample CAT Runs
   B.1 Modeling a Problem in CAT ....................................................... 67
   B.2 Sample Problems Using the Steam Element ............................... 69

Appendix C Source Code
   C.1 Steam Element Computer Code ................................................. 78
      C.1.1 Modified Units ................................................................. 78
      C.1.2 New Units ......................................................................... 120

Appendix D Bibliography
   Bibliography .................................................................................... 139
   Biographical Note ................................................................. 140
List of Figures

2.1 Isolated system .................................................................................................... 18
2.2 Mesh of interconnected elements ........................................................................ 20
2.3 Isolated system with mechanical matching element ........................................... 24
3.1 \( P-v-T \) plot from equation (3.1) ........................................................................ 40
3.2 \( P-v-T \) plot from equation (3.1), first half of saturated region .................. 41
3.3 \( P-v-T \) plot from equation (3.1), second half of saturated region ............... 42
5.1 CAT initial screen ............................................................................................... 46
5.2 Flowchart, unit numerical ................................................................................... 50
5.3 CAT results screen ............................................................................................. 51
6.1 Input routine decision tree (known quality) .................................................. 56
6.2 Input routine decision tree (unknown quality) .............................................. 57
B.1 Sample thermodynamic problem, piston cylinder apparatus .................... 67
B.2 CAT mesh for sample problem ........................................................................ 68
B.3 CAT solution to sample problem ....................................................................... 68
B.4 CAT mesh for problem 1 ................................................................................... 71
B.5 CAT solution to problem 1 ................................................................................ 71
B.6 CAT mesh for problem 2 ................................................................................... 72
B.7 CAT solution to problem 2 ................................................................................ 72
B.8 CAT mesh for problem 3 ................................................................................... 73
B.9 CAT solution to problem 3 ................................................................................ 73
B.10 CAT mesh for problem 4 .................................................................................. 74
B.11 CAT solution to problem 4 .............................................................................. 74
B.12 \( P \) vs. \( V \) plot, problem 4 .............................................................................. 75
B.13 \( P \) vs. \( T \) plot, problem 4 ................................................................................ 75
B.14 CAT mesh for problem 5 .................................................................................. 76
B.15 CAT solution to problem 5 ................................................................................ 76
B.16 CAT mesh for problem 6 .................................................................................. 77
B.17 CAT solution to problem 6 .............................................................................. 77
Part I: Conceptualization and Formulation
1 Preamble

1.1 Background

The thermodynamic behavior of a system can be modeled as a set of mechanical work transfer interactions and thermal heat transfer interactions occurring within the system. These interactions occur among smaller subsystems with defined thermal and mechanical properties. The Computer-Aided Thermodynamic (CAT) algorithm is based on this approach. Instead of devising a particular solution procedure for analysis of any given thermodynamic system, the CAT methodology automates the problem setup in a manner which can easily be solved by a numerical computational algorithm. The automation of problem setup is done by decomposing the thermodynamic system into elements that interact among them by means of mechanical and/or thermal interconnections. Each element has its own set of properties, and its behavior is defined by an equation of state (also referred to here as a constitutive relation). The interconnections define the thermodynamic interaction between elements.

The foundation for the CAT methodology was developed here at the Massachusetts Institute of Technology (MIT) by Professor Joseph L. Smith, Jr., and Dr. Gilberto C. Russo. The CAT algorithm, and corresponding computer software was formally presented in Dr. Russo’s 1987 doctoral thesis “A New Methodology for Computer-Aided Thermodynamics”. The original CAT version was a 1.5-megabytes, FORTRAN-77 computer program, running under the UNIX/ULTRIX operational system. More recently, MIT graduate student Chun-On Cheng updated the computer code to a C computer program, and MIT undergraduate student Oscar C. Yeh translated the C code into cT, a portable language across computer operating systems (UNIX, MS-DOS, Apple Macintosh). Currently, the C version of CAT is being used by MIT undergraduate students as part of the 2.40 Thermodynamics course.

This thesis builds on the past work to produce an algorithm, and corresponding software code, for modeling and analysis of steam as a pure substance. Prior to this work, the CAT algorithm was only able to model substances as single-phase ideal gases. With the addition of the steam element, CAT is now able to model a pure substance, such as water, and its behavior in the liquid, liquid-vapor and vapor states. The new steam element can connect with all the existing elements in CAT, and can be used in final equilib-
rium problems as well as reversible process problems. Although the algorithm can be used with any substance that typically undergoes phase changes, such as Refrigerant-12 for instance, the current version of the CAT runs with water, using the equations developed by J. Keenan, et al., in *Steam Tables* (1969).

1.2 Basis for Method

The CAT software consists of a graphical interface that allows the user to model a given thermodynamic system as an assembly or *mesh* of interconnected elements. The interconnections determine the type of interactions between elements: whether two elements are thermally connected (i.e., equal temperature at equilibrium) or mechanically connected (i.e., equal pressure at equilibrium). After modeling the given thermodynamic system, the computer program performs the necessary steps to generate a solution, namely, setup of equations, solving of equations by numerical methods, and presentation of results.

The equations are established depending on the type of process involved. When modeling final equilibrium problems, we seek to determine the final state of a system given certain initial conditions. This is achieved by generating force-balance and energy-balance equations. The force balance equation establishes that at equilibrium, all mechanically connected elements will have a resultant zero force at each *node*. A node is the point of connection between two elements, for example, it can be thought of as an adiabatic moveable wall separating two gases. At equilibrium, the net force on this wall is zero, meaning the two gases have come to the same pressure. The energy balance equation establishes that all mechanical or thermally connected elements will have the same energy level, mechanical or thermal. For the example of the two gases separated by a wall, if connected by a thermal interconnection only, it would mean having a fixed, thermally conductive wall. At equilibrium these two gases will have the same temperature. When modeling reversible process problems, the energy balance equation is replaced by an entropy balance equation. A *mechanical matching element* is introduced to ensure mechanical equilibrium at every step of the process (quasi-static process), so that the net entropy change is zero.

These equations are solved for the independent variables, namely temperature and nodal displacement at each element. CAT uses a Newton-Raphson algorithm to solve the
equations. The Newton-Raphson method requires that derivatives for all equations, with respect to each independent variable be calculated. These are programmed in the CAT source code. At each iteration CAT performs several checks to ensure the iteration process is headed in the right direction. If any of the variables drift from the expected solution (e.g., negative temperatures or volumes) CAT takes corrective action in order to produce a solution that will not violate the laws of thermodynamics.

Finally, the results are displayed to the user in a tabular form, listing the properties for each element along with the initial and final states of the major properties. In the reversible process case, there is even a function that will plot the property values as they changed from initial to final conditions.

The new steam element was added to couple with the CAT program without any major changes to the basic algorithm. As such, it equips the CAT program with the necessary functions and routines to effectively model the behavior of steam as a pure substance, taking into account changes of the substance phase which are typical of steam applications.

1.3 cT-Based Application Development

The choice of cT as the development language satisfied the following functional requirements:

- modular programming
- portability across computer systems
- dynamic memory allocation
- capability for graphical interface programming.

cT is an integrated programming environment developed in the Center for Design of Educational Computing at Carnegie Mellon University. It is designed to help develop interactive applications for modern computers, and the programs can run without change on Macintosh, MS-DOS, and UNIX with X11. For these reasons, cT was found suitable for the development of CAT and the steam element algorithm extension.
2 CAT Methodology

2.1 Overview

CAT was established in an attempt to standardize problem solving in thermodynamics to a larger extent than it is currently possible. Solutions to thermodynamic problems have typically been ad hoc, equations devised for a specific problem were valid for that type of problem only. Even though solutions to thermodynamic problems are often derived from energy balance equations, and the application of the First and Second Laws of Thermodynamics, once these governing equations are simplified to suit a particular problem, their generality is lost. CAT preserves this generality at the expense of a few extra steps in calculations, but in a manner that can easily be handled by a computer.

2.2 Modeling Concept

A thermodynamic system is represented as an isolated system composed of interconnected subsystems, as shown in Figure 2.1. Each subsystem is described by a specified list of equilibrium properties and equations of state (constitutive relations). Each of the subsystems is modeled as being capable of one or more energy transfer interactions. Currently, CAT support two types of energy transfer between subsystems (elements). They are: heat transfer, due to a temperature difference (represented by a "T" symbol, for thermal interaction) and work transfer, due to boundary forces and related displacements (represented by a "M" symbol, for mechanical interaction).

In addition to allowing energy transfer between elements, the interconnections imply that at equilibrium, certain conditions will exist. Specifically, a thermal interconnection implies temperature equality between the two interconnected elements at equilibrium. In Figure 2.1, the temperature of element 1 will be the same as the temperature 2, upon reaching system equilibrium. A mechanical interconnection means force or pressure equality between the two interconnected elements at equilibrium. In Figure 2.1, the pressure of element 2 will be the same as the pressure of element 3, upon reaching system equilibrium. The resultant force at the boundary (represented by the "M") will be zero.
2.3 CAT Elements

Elements are used to model physical entities. An element is defined by the following attributes:
- a set of constitutive relations
- a set of independent properties
- a set of dependent properties.

The elements modeled by CAT (not including the Pure Substance/Steam element, which is the focus of this thesis and is treated separately in Section 3) and their attributes are tabulated below.

<table>
<thead>
<tr>
<th>Element</th>
<th>Constitutive Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thermal reservoir</td>
<td>Any $U$ to keep $T$ constant</td>
</tr>
<tr>
<td>Thermal capacity</td>
<td>$U = mc_v T$</td>
</tr>
<tr>
<td>Pressure reservoir</td>
<td>$F = -PA$</td>
</tr>
<tr>
<td>Ideal Spring</td>
<td>$F = k(L - L_0)$</td>
</tr>
<tr>
<td>Ideal gas</td>
<td>$P = mRT/V$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Pressure/Force</th>
<th>Energy</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thermal reservoir</td>
<td>-</td>
<td>Any $U$ to keep $T$ constant</td>
<td>-</td>
</tr>
<tr>
<td>Thermal capacity</td>
<td>-</td>
<td>$U = mc_v T$</td>
<td>$S = mc_v \ln T$</td>
</tr>
<tr>
<td>Pressure reservoir</td>
<td>$F = -PA$</td>
<td>$U = \int Fdl$</td>
<td>-</td>
</tr>
<tr>
<td>Ideal Spring</td>
<td>$F = k(L - L_0)$</td>
<td>$U = \frac{1}{2}k(L - L_0)^2$</td>
<td>-</td>
</tr>
<tr>
<td>Ideal gas</td>
<td>$P = mRT/V$</td>
<td>$U = mc_v T$</td>
<td>$S = mR \ln V + mc_v \ln T$</td>
</tr>
</tbody>
</table>

Table 2.1: Elements and their constitutive relations
<table>
<thead>
<tr>
<th>Element</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Independent</td>
</tr>
<tr>
<td>Thermal reservoir</td>
<td>$T$</td>
</tr>
<tr>
<td>Thermal capacity</td>
<td>$T$</td>
</tr>
<tr>
<td>Pressure reservoir</td>
<td>$L$</td>
</tr>
<tr>
<td>Ideal Spring</td>
<td>$L$</td>
</tr>
<tr>
<td>Ideal gas</td>
<td>$V, T$</td>
</tr>
</tbody>
</table>

Table 2.2: Elements and their properties

2.4 Topology Matrices

In order to systematically describe the arrangement of elements and their interconnections for any given problem, CAT generates topology matrices, one for each type of interconnection involved. Consider a mesh of interconnected elements as shown in Figure 2.2. When the mesh has been completely initialized, all the physical and initial state properties are specified for each element. Moreover, the interconnections are inactive or “frozen” prior to the simulation and equation solving. The mechanical topology matrix is defined as follows:

$$T_M = \begin{bmatrix} \alpha_i^1 & \cdots & \alpha_i^n \\ \vdots & \ddots & \vdots \\ \alpha_m^1 & \cdots & \alpha_m^n \end{bmatrix}$$

(2.1)

where

- $m = \text{number of mechanical interconnections in the mesh}$
- $n = \text{number of elements in the mesh}$
- $\alpha_i^j = \begin{cases} -1 & \text{if element } j \text{ is connected to mechanical interconnection } i \\ 0 & \text{otherwise.} \end{cases}$
Similarly, the thermal topology matrix is defined as

\[
T_t = \begin{bmatrix}
\beta_1^t & \cdots & \beta_n^t \\
\vdots & \ddots & \vdots \\
\beta_1^n & \cdots & \beta_n^n
\end{bmatrix}
\]

(2.2)

where

\[
t = \text{number of thermal interconnections in the mesh}
\]

\[
n = \text{number of elements in the mesh}
\]

\[
\beta_i^j = \begin{cases} 
1 & \text{if element } j \text{ is connected to thermal interconnection } i \\
0 & \text{otherwise.}
\end{cases}
\]

We use the following notation:

- Parameters with superscripts are for element or local state parameters; the superscript is the element number. For instance, \( V^1 \) is the volume of element one.
- Parameters with subscripts are for system or global state parameters; the subscript is the interconnection number. For example, \( x_1 \) is the boundary (nodal) displacement associated with interconnection one.
- Parameters with superscripts and subscripts are for element nodal parameters. Here the superscript and the subscript are the element and the interconnection that share that node. \( F_1^2 \) is the force exerted by element number two on mechanical interconnection number one.

![Figure 2.2: Mesh of interconnected elements](image_url)

The continuity of mechanically interconnected elements relate element nodal displacements to the volume at each element. For the mesh in Figure 2.2
\[ V^i = V^i_{\text{initial}} + A x_i \]
\[ V^i = V^i_{\text{initial}} + A(x_i - x_{i-1}) \quad (i = 2, \ldots, n-1) \quad (2.3) \]
\[ V^n = V^n_{\text{initial}} + A x_{n-1}. \]

There is no continuity requirement, however, for thermal interconnections.

### 2.5 Derivation of System Equations

Once all elements have been initialized and the topology matrices generated, CAT proceeds to establish the main system equations. These equations describe the behavior of the entire, isolated system by summing their respective force and energy constitutive relations. The solution to these equations constitute the final equilibrium state of the system. For the case of reversible process, the energy equation is replaced by an entropy summation equation. The final equilibrium case is discussed first. For simplicity of presentation, let us consider the case of a system with a single thermal domain. A thermal domain is a portion of the system that consist of thermally interconnected elements. These elements may have different initial temperatures, but at equilibrium they all have the same temperature.

Let \( F_i^j \) be the resultant nodal force residual at the \( i^{th} \) mechanical interconnection and \( E_j^j \) be the energy at the \( j^{th} \) element. Then the global system equations are

\[ R_i = \sum_{j=1}^{n} F_i^j \quad ; \quad i = 1, 2, \ldots, n - 1 \quad (2.4) \]
\[ R_E = \sum_{j=1}^{n} E_i^j - E_0^j. \quad (2.5) \]

There is a total of \( n \) nonlinear, simultaneous equations. When the residuals \( R_i \) and \( R_E \) are zero, mechanical and thermal equilibrium is reached. Our goal is to find the values of \( x_i \) and \( T \), such that all residuals are zero. This is accomplished by using the Newton-Raphson method for nonlinear equations. We linearize equations (2.4) and (2.5) by

\[ \Delta R_i = R_i - 0 = \sum_{j=1}^{n} \frac{\partial R_i}{\partial x_j} + \frac{\partial R_i}{\partial T}; \quad i = 1, 2, \ldots, n - 1 \quad (2.6) \]
\[ \Delta R_k = R_k - 0 = \sum_{j=1}^{n} \frac{\partial E}{\partial x_j} + \frac{\partial E}{\partial T} \] (2.7)

These set of linear equations can be rewritten as the matrix form

\[
\begin{bmatrix}
\frac{\partial R_1}{\partial x_1} & \cdots & \frac{\partial R_1}{\partial x_{n-1}} & \frac{\partial R_1}{\partial T} \\
\vdots & \ddots & \vdots & \vdots \\
\frac{\partial R_{n-1}}{\partial x_1} & \cdots & \frac{\partial R_{n-1}}{\partial x_{n-1}} & \frac{\partial R_{n-1}}{\partial T} \\
\frac{\partial R_n}{\partial x_1} & \cdots & \frac{\partial R_n}{\partial x_{n-1}} & \frac{\partial R_n}{\partial T}
\end{bmatrix}
\begin{bmatrix}
\Delta x_1 \\
\vdots \\
\Delta x_{n-1} \\
\Delta T
\end{bmatrix}
= \begin{bmatrix}
\Delta R_1 \\
\vdots \\
\Delta R_{n-1} \\
\Delta R_n
\end{bmatrix}. 
\] (2.8)

The square matrix \((n \times n)\) on the left hand side of equation (2.8) is known as the stiffness matrix. The column vector on the right hand side contains the force and energy residuals. By finding the inverse of the stiffness matrix (by means of a Gauss elimination algorithm) we can solve the equation for the nodal displacements and the system temperature. This is done in successive iterations, where the variables of interest, \(x_i\) and \(T\), are updated as

\[ x_{i}^{k+1} = x_{i}^{k} - \Delta x_{i}^{k} \; ; \; (i = 1, 2, \ldots, n-1) \]
\[ T^{k+1} = T^{k} - \Delta T^{k} . \] (2.9)

When the system reaches equilibrium state, all nodal forces are zero and all thermally connected elements reach a common temperature; the residuals vector is zero.

### 2.6 Stiffness Matrix Elements

The entries of the stiffness matrix in equation (2.8) are given by the following expressions

\[ \frac{\partial R_i}{\partial x_j} = \sum_{k=1}^{n} \frac{\partial F_i^k}{\partial x_j} = \sum_{k=1}^{n} \frac{\partial F_i^k}{\partial p^k} \frac{\partial p^k}{\partial x_j} = \sum_{k=1}^{n} (A_i \alpha_i^k) \frac{\partial p^k}{\partial V^k} (A_i \alpha_i^k) \] (2.10)
\[ \frac{\partial R_i}{\partial T} = \sum_{k=1}^{n} \frac{\partial F_i^k}{\partial T} = \sum_{k=1}^{n} \frac{\partial F_i^k}{\partial p^k} \frac{\partial p^k}{\partial T} = \sum_{k=1}^{n} (A_i \beta_i^k) \frac{\partial p^k}{\partial T} \] (2.11)
\[
\frac{\partial R_E}{\partial x_j} = \sum_{k=1}^{n} \frac{\partial E^k}{\partial x_j} = \sum_{k=1}^{n} \frac{\partial E^k}{\partial x_j} \frac{\partial V^k}{\partial x_j} = \sum_{k=1}^{n} \frac{\partial E^k}{\partial x_j} (A_k \alpha_j^k) 
\]
(2.12)

\[
\frac{\partial R_E}{\partial T} = \sum_{k=1}^{n} \frac{\partial E^k}{\partial T}.
\]
(2.13)

The terms \(\frac{\partial E^k}{\partial x_j}\), \(\frac{\partial E^k}{\partial x_j}\), \(\frac{\partial E^k}{\partial x_j}\), and \(\frac{\partial E^k}{\partial x_j}\) in equations (2.10) through (2.13) can be derived for each element from the element constitutive relations (Table 2.1). Notice that the partial derivatives in the stiffness matrix are also dependent on the mesh topological information. Typically, many of the terms in the above summations are zero from the mesh topology alone. Table 2.3 lists the terms \(\frac{\partial E^k}{\partial x_j}\), \(\frac{\partial E^k}{\partial x_j}\), \(\frac{\partial E^k}{\partial x_j}\), and \(\frac{\partial E^k}{\partial x_j}\) for each element in CAT.

<table>
<thead>
<tr>
<th>Element</th>
<th>(\frac{\partial E^k}{\partial x_j})</th>
<th>(\frac{\partial E^k}{\partial x_j})</th>
<th>(\frac{\partial E^k}{\partial x_j})</th>
<th>(\frac{\partial E^k}{\partial x_j})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thermal reservoir</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Thermal capacity</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>(mc_v)</td>
</tr>
<tr>
<td>Pressure reservoir</td>
<td>0</td>
<td>0</td>
<td>(-P)</td>
<td>0</td>
</tr>
<tr>
<td>Ideal Spring</td>
<td>(\frac{\partial E^k}{\partial x_j} = -k)</td>
<td>0</td>
<td>(\frac{\partial E^k}{\partial x_j} = kx)</td>
<td>0</td>
</tr>
<tr>
<td>Ideal gas</td>
<td>(-\frac{\partial E^k}{\partial x_j} = \frac{\psi R}{\beta})</td>
<td>(\frac{\partial E^k}{\partial x_j} = \frac{\psi R}{\beta})</td>
<td>0</td>
<td>(mc_v)</td>
</tr>
</tbody>
</table>

Table 2.3: Expressions for Stiffness Matrix

A note about the thermal reservoir element is in order. When there is a thermal reservoir in the mesh, the system temperature is fixed by the thermal reservoir temperature. Since the thermal reservoir acts as an energy source (or sink), energy in the mesh is no longer conserved and the energy balance equation can be deleted from our system of equations. Temperature is no longer an independent variable, and the stiffness matrix loses its last row and column. The stiffness matrix becomes a \(n-1\) square matrix and the matrix equation (2.8) reduces to

\[
\begin{bmatrix}
\frac{\partial R_1}{\partial x_1} & \cdots & \frac{\partial R_n}{\partial x_1} \\
\vdots & \ddots & \vdots \\
\frac{\partial R_1}{\partial x_{n-1}} & \cdots & \frac{\partial R_n}{\partial x_{n-1}}
\end{bmatrix}
\begin{bmatrix}
\Delta x_1 \\
\vdots \\
\Delta x_{n-1}
\end{bmatrix}
= 
\begin{bmatrix}
\Delta R_1 \\
\vdots \\
\Delta R_{n-1}
\end{bmatrix}.
\]
(2.14)
2.7 Reversible Process

The solution of a reversible problem is built upon the final equilibrium solution methodology. For an isolated system to undergo a reversible process, the system must go from an initial state to a final state through a continuum of equilibrium states, i.e., at each intermediate state, the system must be in neutral equilibrium. In CAT, this is accomplished by introducing a new element that ensures mechanical equilibrium of the system during the reversible process. We call this element the mechanical matching element since its job is to match the residual force of the system, thereby attaining mechanical equilibrium at each step of the process.

![Figure 2.3: Isolated system with mechanical matching element](image)

In contrast to a final equilibrium problem, where energy is conserved, in a reversible process problem, entropy remains constant. Computationally, this means replacing the energy conservation equation by a conservation of entropy equation. It is also necessary to specify a final state property for one of the elements in the mesh, in order to dictate the final state of the system. This state property is termed the handle, $h$. The goal of the process is go from $h_i$ to $h_f$ while keeping the system entropy constant. This results in an extra residual equation in the system. The new set of equations are of the form

$$R_s = \sum_{j=1}^{n} F_i^j ; i = 1, 2, ..., n - 2$$  \hspace{1cm} (2.15)

$$R_s = \sum_{j=1}^{n} S^j - S_0^j$$  \hspace{1cm} (2.16)

$$R_s = h - h_f.$$  \hspace{1cm} (2.17)

These equations are solved using the Newton-Raphson method. The linearized equations in matrix form are given by
The partial derivatives in the stiffness matrix of equation (2.18) are generally derived the same way as the final equilibrium case. The partial derivatives for $R_h$ and $R$ are calculated below.

$$\frac{\partial R_h}{\partial x_j} = \frac{\partial h}{\partial V} \frac{\partial V}{\partial x_j} = \frac{\partial h}{\partial V} (A_j \alpha_j)$$  \hspace{1cm} (2.19)$$

$$\frac{\partial R_h}{\partial T} = \frac{\partial h}{\partial T}$$ \hspace{1cm} (2.20)$$

$$\frac{\partial R_s}{\partial x_j} = \sum_{k=1}^{n} \frac{\partial s^k}{\partial x_j} = \sum_{k=1}^{n} \frac{\partial s^k}{\partial V^k} \frac{\partial V^k}{\partial x_j} = \sum_{k=1}^{n} \frac{\partial s^k}{\partial V^k} (A \alpha^k_j)$$ \hspace{1cm} (2.21)$$

$$\frac{\partial R_s}{\partial T} = \sum_{k=1}^{n} \frac{\partial s^k}{\partial T}.$$ \hspace{1cm} (2.22)$$

The expressions $\frac{\partial h}{\partial T}$, $\frac{\partial h}{\partial V}$, $\frac{\partial s^k}{\partial x_j}$ and $\frac{\partial s^k}{\partial T}$ are tabulated below.

<table>
<thead>
<tr>
<th>$h$</th>
<th>$\frac{\partial h}{\partial T}$</th>
<th>$\frac{\partial h}{\partial V}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$P$</td>
<td>$-\frac{m \beta T}{\rho}$</td>
<td>$\frac{m \beta}{\rho}$</td>
</tr>
<tr>
<td>$T$</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2.4: Handle derivatives for Stiffness Matrix
<table>
<thead>
<tr>
<th>Element</th>
<th>( \frac{\partial S}{\partial V} )</th>
<th>( \frac{\partial S}{\partial T} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thermal reservoir</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Thermal capacity</td>
<td>0</td>
<td>( \frac{mc_p}{V} )</td>
</tr>
<tr>
<td>Pressure reservoir</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Ideal Spring</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Ideal gas</td>
<td>( \frac{mR}{V} )</td>
<td>( \frac{mc_p}{T} )</td>
</tr>
</tbody>
</table>

Table 2.5: Entropy derivatives for Stiffness Matrix
3 Pure Substance Element Modeling

3.1 Overview

The modeling of the pure substance element requires the use of an explicit analytical function of sufficient accuracy. This function is the $P - \rho - T$ constitutive relation, also known as the equation of state for the substance. Ideally, this equation would be valid for the solid, liquid and vapor phases of the substance. However, the behavior of the properties is so different for the various phases that a common approach is to develop $P - \rho - T$ functions valid for each phase or combination of phases of the substance. This model is very general and represents a system without chemical reactions. The pure substance model is very useful in an ample range of physical situations and thermodynamic applications. It requires additional attention when applying the first and second laws of thermodynamics than is necessary for the ideal gas element. In particular, the possibility of a phase change that accompanies any property change in the pure substance model requires evaluation of the substance from charts or tables. For the purpose of developing and implementing an algorithm within the CAT program framework, we consider the case of water as a pure substance. Its phases most frequently found in thermodynamic applications are vapor, liquid-vapor mixture, and liquid.

3.2 Equations for Water Vapor and Liquid Regions

The expressions we will need for this region, as required by the CAT algorithm, are $P = P(\rho,T)$, $U = U(\rho,T)$, and $S = S(\rho,T)$. In this model of water as a pure substance we will adopt the basic constitutive relations from the steam equations as presented in Reynolds, W., Thermodynamic Properties in SI, Stanford University, 1979, which are a slightly modified version of the ones used in Keenan, J., et al., Steam Tables, John Wiley and Sons, 1969. The $P - \rho - T$ equation is given by:

$$P = \rho RT \left[ 1 + \rho Q + \rho^2 \left( \frac{\partial Q}{\partial \rho} \right)_T \right]$$

(3.1)
where,

\[
Q = (\tau - \tau_c) \sum_{j=1}^{7} (\tau - \tau_{a_j})^{i-2} \left[ \sum_{i=1}^{8} A_i (\rho - \rho_{a_i})^{i-1} + e^{-E} \sum_{i=9}^{10} A_i \rho_i^{i-9} \right]
\]  

(3.2)

\[
\tau = \frac{1000}{T}
\]  

(3.3)

\[
\tau_{a_1} = 1.5449121; \quad \tau_{a_j} = 2.5, \quad j > 1
\]

\[
\rho_{a_1} = 634; \quad \rho_{a_j} = 1000, \quad j > 1
\]

\[
E = 4.8 \times 10^{-3}; \quad R = 461.51 \text{ J/kg \cdot K}
\]

and \( A_i \) is as shown in Table 3.1. Constants for \( T \) in K, \( \rho \) in kg/m\(^3\) and \( P \) in Pa.

In order to find the term \( \left( \frac{\partial^2 \rho}{\partial \rho^2} \right)_T \), let us first define

\[
H_j(\rho) = \sum_{i=1}^{8} A_i (\rho - \rho_{a_i})^{i-1} + e^{-E} \sum_{i=9}^{10} A_i \rho_i^{i-9}
\]

(3.4)

\[
H'_{j}(\rho) = \frac{dH_j(\rho)}{d\rho} = \sum_{i=2}^{8} (i-1)A_i (\rho - \rho_{a_i})^{i-2} - e^{-E} \sum_{i=9}^{10} A_i \rho_i^{i-9} + e^{-E} A_{10j}
\]

(3.5)

\[
H''_{j}(\rho) = \frac{d^2 H_j(\rho)}{d\rho^2} = \sum_{i=3}^{8} (i-1)(i-2)A_i (\rho - \rho_{a_i})^{i-3} + E^2 e^{-E} \sum_{i=9}^{10} A_i \rho_i^{i-9} - 2E e^{-E} A_{10j}
\]

(3.6)

we then find \( \left( \frac{\partial^2 \rho}{\partial \rho^2} \right)_T \) as follows:

\[
\left( \frac{\partial^2 \rho}{\partial \rho^2} \right)_T = (\tau - \tau_c) \sum_{j=1}^{7} (\tau - \tau_{a_j})^{i-2} H'_{j}(\rho).
\]  

(3.7)

The energy equation can be written in terms of the specific energy,

\[
U(\rho, T) = m u(\rho, T)
\]  

(3.8)

where \( m \) is the mass and

\[
u = \int_{T_0}^{T} c_v^0(T) \rho \frac{1}{\rho^2} \left[ P - T \frac{\partial P}{\partial T} \rho \right] d\rho + u_0
\]

(3.9)
<table>
<thead>
<tr>
<th>$i$</th>
<th>$j$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>2.9492937 × 10^{-2}</td>
<td>-5.1985860 × 10^{-3}</td>
<td>6.8335354 × 10^{-3}</td>
<td>-1.5641040 × 10^{-4}</td>
<td>-6.3972405 × 10^{-3}</td>
<td>-3.9661401 × 10^{-3}</td>
<td>-6.9048554 × 10^{-4}</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>-1.3213917 × 10^{-4}</td>
<td>7.7779182 × 10^{-6}</td>
<td>-2.6149751 × 10^{-5}</td>
<td>-7.2546108 × 10^{-7}</td>
<td>2.6409282 × 10^{-5}</td>
<td>1.5453061 × 10^{-5}</td>
<td>2.7407416 × 10^{-6}</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>2.7464632 × 10^{-7}</td>
<td>-3.3301902 × 10^{-8}</td>
<td>6.5326396 × 10^{-8}</td>
<td>-9.2734289 × 10^{-9}</td>
<td>-4.7740374 × 10^{-8}</td>
<td>-2.9142470 × 10^{-8}</td>
<td>-5.1028070 × 10^{-9}</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>-3.6093828 × 10^{-10}</td>
<td>-1.6254622 × 10^{-11}</td>
<td>-2.6181978 × 10^{-11}</td>
<td>4.3125840 × 10^{-12}</td>
<td>5.6323130 × 10^{-11}</td>
<td>2.9568796 × 10^{-11}</td>
<td>3.9636085 × 10^{-12}</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>3.4218431 × 10^{-13}</td>
<td>-1.7731074 × 10^{-13}</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>-2.4450042 × 10^{-16}</td>
<td>1.2748742 × 10^{-16}</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>1.5518535 × 10^{-19}</td>
<td>1.3746153 × 10^{-19}</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>5.9728487 × 10^{-24}</td>
<td>1.5597836 × 10^{-22}</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>-4.1030848 × 10^{-1}</td>
<td>3.3731180 × 10^{-1}</td>
<td>-1.3746618 × 10^{-1}</td>
<td>6.7874983 × 10^{-3}</td>
<td>1.3687317 × 10^{-1}</td>
<td>7.9847970 × 10^{-2}</td>
<td>1.3041253 × 10^{-2}</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>-4.1605860 × 10^{-4}</td>
<td>-2.0988666 × 10^{-4}</td>
<td>-7.3396848 × 10^{-4}</td>
<td>1.0401717 × 10^{-5}</td>
<td>6.4581880 × 10^{-4}</td>
<td>3.9917570 × 10^{-4}</td>
<td>7.1531353 × 10^{-5}</td>
</tr>
</tbody>
</table>

Table 3.1: The Coefficients $A_y$ in Equation (3.2)
where,

\[
\left( \frac{\partial P}{\partial T} \right)_P = \rho R \left[ 1 + \rho Q + \rho T \left( \frac{\partial Q}{\partial \tau} \right)_P \frac{d\tau}{dT} + \rho^2 T \left( \frac{\partial Q}{\partial \rho} \right)_T \frac{d\tau}{dT} + \rho^3 \left( \frac{\partial Q}{\partial \rho} \right)_T \right]
\]  \hspace{1cm} (3.10)

\[
\frac{d\tau}{dT} = -\frac{1000}{T^2} = -\frac{\tau}{T}
\]  \hspace{1cm} (3.11)

\[
\int_0^\rho \frac{1}{\rho^2} \left[ P - T \left( \frac{\partial P}{\partial T} \right)_P \right] d\rho = \int_0^T \left[ \frac{RT}{\rho} + RTQ + \rho RT \left( \frac{\partial Q}{\partial \rho} \right)_T - \frac{T}{\rho^2} \left( \frac{\partial P}{\partial T} \right)_\rho \right] d\rho
\]

\[
= -RT^2 \frac{d\tau}{dT} \int_0^\rho \left[ \left( \frac{\partial Q}{\partial \tau} \right)_P + \rho \frac{\partial}{\partial \rho} \left( \frac{\partial Q}{\partial \tau} \right)_P \right] d\rho
\]

\[
= -RT^2 \frac{d\tau}{dT} \int_0^\rho \frac{\partial}{\partial \rho} \left( \rho \left( \frac{\partial Q}{\partial \tau} \right)_P \right) d\rho
\]

\[
= -RT^2 \frac{d\tau}{dT} \rho \left( \frac{\partial Q}{\partial \tau} \right)_P
\]

\[
= 1000 \rho R \left( \frac{\partial Q}{\partial \tau} \right)_P
\]  \hspace{1cm} (3.12)

\[
\frac{\partial}{\partial \tau} \left( \frac{\partial Q}{\partial \rho} \right)_T = \left[ \sum_{j=1}^7 (\tau - \tau_{aj})^{-2} + (\tau - \tau_c)(j - 2)(\tau - \tau_{aj})^{-2} \right] H_j(\rho).
\]  \hspace{1cm} (3.13)

Simplifying, we obtain

\[
u = \int_{T_0}^T c_v(T) dT + 1000 \rho R \left( \frac{\partial Q}{\partial \tau} \right)_P + u_0
\]  \hspace{1cm} (3.14)

where,

\[
\left( \frac{\partial Q}{\partial \tau} \right)_P = \sum_{j=1}^7 \left[ (j - 2)(\tau - \tau_c)(\tau - \tau_{aj})^{-2} + (\tau - \tau_c)(j - 2)(\tau - \tau_{aj})^{-3} \right] H_j(\rho)
\]  \hspace{1cm} (3.15)

\[
T_0 = 273.16; \ u_0 = 2375020.7 \ \text{J/kg}
\]

\[
c_v(T) = \sum_{i=1}^6 G_i T^{i-2}
\]  \hspace{1cm} (3.16)

\[
G_1 = 46000 \ \ \ G_2 = -2.19989 \times 10^{-4}
\]

\[
G_2 = 1011.249 \ \ \ G_3 = 2.46619 \times 10^{-7}
\]

\[
G_3 = 0.83893 \ \ \ G_6 = -9.7047 \times 10^{-11}
\]
\[ \int_{T_0}^{T} c_v^0(T) dT = G_1 \ln T + \sum_{i=2}^{6} \frac{G_i}{T} T^{i-1} - G_i \ln T_0 - \sum_{i=2}^{6} \frac{G_i}{T} T_0^{i-1} \]

\[ = G_1 \ln T + \sum_{i=2}^{6} \frac{G_i}{T} T_0^{i-1} - 564413.52 \]  

(3.17)

\( u \) is in J/kg, \( c_v \) in J/kg·K. The equation for specific entropy is given by

\[ s = \int_{T_0}^{T} \frac{c_v^0}{T} dT - R \ln \rho + \int_{0}^{\rho} \frac{1}{\rho^2} \left[ \rho R - \frac{1}{\rho} \left( \frac{\partial P}{\partial T} \right)_\rho \right] d\rho + s_0 \]  

(3.18)

where,

\[ s_0 = 6696.5776 \]

\[ \int_{T_0}^{T} \frac{c_v^0}{T} dT = -\frac{G_1}{T} + G_2 \ln T + \sum_{i=3}^{6} \frac{G_i}{T_{i-2}} T_{i-1} - G_i \ln T_0 - \sum_{i=3}^{6} \frac{G_i}{T_{i-2}} T_0^{i-2} \]

\[ = -\frac{G_1}{T} + G_2 \ln T + \sum_{i=3}^{6} \frac{G_i}{T_{i-2}} T_0^{i-2} - 5727.2610 \]  

(3.19)

\[ \int_{0}^{\rho} \frac{1}{\rho^2} \left[ \rho R - \frac{1}{\rho} \left( \frac{\partial P}{\partial T} \right)_\rho \right] d\rho = \int_{0}^{\rho} \left[ \frac{R}{\rho} - \frac{1}{\rho^2} \left( \frac{\partial P}{\partial T} \right)_\rho \right] d\rho \]

\[ = \int_{0}^{\rho} \left[ \frac{R}{\rho} - \frac{R}{\rho} - RQ - \rho R \left( \frac{\partial Q}{\partial \tau} \right)_T \right. \]

\[ -R \left( \frac{\partial Q}{\partial \tau} \right)_T \frac{d\tau}{d\rho} \]  

(3.20)

\[ \int_{0}^{\rho} \left[ \frac{R}{\rho} - \rho Q + \tau \left( \frac{\partial Q}{\partial \tau} \right)_T \right] d\rho \]

\[ = \rho R \left[ \frac{R}{\rho} - \rho Q + \tau \left( \frac{\partial Q}{\partial \tau} \right)_T \right] \]

\[ = \rho R \left[ \frac{R}{\rho} - \rho Q + \tau \left( \frac{\partial Q}{\partial \tau} \right)_T \right] \]

\[ = s \] is in J/kg·K.

The preceding equations will allow us to determine pressure, specific energy and specific entropy as a function of density and temperature. They are valid for the vapor and liquid phases of water.
3.3 Equations for Water Liquid-Vapor Mixture (Saturation)

The expressions needed for this region, as required by the CAT algorithm, are again, \( P = P(T), U = U(\rho, T), \) and \( S = S(\rho, T) \). The \( P-T \) equation is given by:

\[
P = P_c \exp \left\{ \left( \frac{T_c}{T - 1} \right)^{i-1} \sum_{i=1}^{8} F_i \left( a(T - T_p) \right)^i \right\}
\]  

(3.21)

where,

\[
P_c = 22.089 \text{ MPa}; \quad T_c = 647.286 \text{ K}
\]

\[
F_i = \begin{align*}
F_1 &= -7.4192420 \\
F_2 &= 2.9721000 \times 10^4 \\
F_3 &= -1.1552860 \times 10^3 \\
F_4 &= 8.6856350 \times 10^-3 \\
F_5 &= 1.0940980 \times 10^3 \\
F_6 &= -4.3999300 \times 10^3 \\
F_7 &= 2.5206580 \times 10^3 \\
F_8 &= -5.2186840 \times 10^4 \\
a &= 0.01; \quad T_p = 338.15 \text{ K}
\end{align*}
\]

The energy equation is given by

\[
U(\rho, T) = m u(\rho, T) = m (u_f + \chi_u u_g)
\]  

(3.22)

where,

\[
u_f = u(\rho_f, T)
\]  

(3.23)

in which the function \( u(\rho, T) \) is as given in equation (3.14), and

\[
\rho_f = \rho_c \left[ 1 + \sum_{i=1}^{8} D_i (1 - T / T_c)^{i/3} \right]
\]  

(3.24)

where,

\[
\rho_c = 317.0 \text{ kg/m}^3
\]

\[
D_i = \begin{align*}
D_1 &= 3.6711257 \\
D_2 &= -2.8512396 \times 10^4 \\
D_3 &= -8.8243852 \times 10^2 \\
D_4 &= 2.2265240 \times 10^2 \\
D_5 &= 1.8297674 \times 10^3 \\
D_6 &= -2.6122557 \times 10^3 \\
D_7 &= -5.3350520 \times 10^2 \\
D_8 &= -5.3350520 \times 10^2
\end{align*}
\]
\[ x = \frac{v - v_f}{v_f} = \frac{v - v_f}{v_f} = \frac{1}{\rho - 1 / \rho_f} \]

\[ \rho_g = \rho(P_{\text{sat}}, T_{\text{sat}}). \]

We have not defined an explicit function for \( \rho = \rho(P, T) \) as expressed in equation (3.26) and there was not one available. However, \( \rho_g \) as a function of \( T \) can be computed by using the following methodology:

1) Given the temperature, \( T \), we use equation (3.21) to calculate \( P_{\text{sat}} \)
2) Using these two values \((T_{\text{sat}}, P_{\text{sat}})\) we iterate in equation (3.1) to obtain \( \rho_g \).

An iteration scheme such as Newton-Raphson can be used, where

\[ \rho_g^{k+1} = \rho_g^k - \frac{P(\rho_g^k, T_{\text{sat}}) - P_{\text{sat}}}{\frac{\partial P(\rho_g^k, T_{\text{sat}})}{\partial \rho_g^k}}. \]

The Newton-Raphson algorithm requires an initial guess, \( \rho_g^0 \).

The equation for specific energy in the saturated region is

\[ u_g = u(\rho_g, T) \]

where, again, the function \( u(\rho, T) \) is as given in equation (3.14). Finally we have

\[ u_f = u_g - u_f. \]

The entropy equation is given by

\[ S(\rho, T) = m s(\rho, T) = m (s_f + x s_{fg}) \]

where,

\[ s_f = s(\rho_f, T) \]

in which the function \( s = s(\rho, T) \) is as given in equation (3.18). Similarly,

\[ s_g = s(\rho_g, T) \]

where \( \rho_g \) is calculated as previously explained.
3.4 Stiffness Matrix Expressions for Water Vapor and Liquid Regions

The derivatives required for the Newton-Raphson stiffness matrix, namely \( \frac{\partial P}{\partial V} \), \( \frac{\partial u}{\partial V} \), \( \frac{\partial u}{\partial T} \), \( \frac{\partial \xi}{\partial V} \), and \( \frac{\partial \xi}{\partial T} \), are calculated below, from the equations in section 3.2:

\[
\left( \frac{\partial P}{\partial V} \right)_T = \left( \frac{\partial P}{\partial \rho} \right)_T \frac{d\rho}{dV} = -\frac{m}{V^2} \left( \frac{\partial P}{\partial \rho} \right)_T \tag{3.32}
\]

\[
\left( \frac{\partial P}{\partial \rho} \right)_T = RT \left[ 1 + 2\rho Q + \rho^2 \left( \frac{\partial Q}{\partial \rho} \right)_T + 3\rho^2 \left( \frac{\partial Q}{\partial \rho} \right)_T + \rho^3 \left( \frac{\partial^2 Q}{\partial \rho^2} \right)_T \right] \tag{3.33}
\]

where,

\[
\left( \frac{\partial^2 Q}{\partial \rho^2} \right)_T = (\tau - \tau_i) \sum_{j=1}^{7} (\tau - \tau_j)^{j-2} H_j(\rho). \tag{3.34}
\]

The other pressure derivative we seek is \( \frac{\partial \xi}{\partial \rho} \), which was already calculated as equation (3.10) (Section 3.2).

The first energy derivative, \( \frac{\partial U}{\partial V} \), is found as

\[
\left( \frac{\partial U}{\partial V} \right)_T = m \left( \frac{\partial u}{\partial V} \right)_T = m \left( \frac{\partial u}{\partial \rho} \right)_T \frac{d\rho}{dV} = -\frac{m^2}{V^2} \left( \frac{\partial u}{\partial \rho} \right)_T \tag{3.35}
\]

\[
\left( \frac{\partial u}{\partial \rho} \right)_T = 1000 \left[ \frac{\partial Q}{\partial \tau} \right]_\rho + \rho \frac{\partial}{\partial \rho} \left( \frac{\partial Q}{\partial \tau} \right)_\rho \tag{3.36}
\]

We had already calculated \( \frac{\partial \xi}{\partial \rho} \) in equation (3.13).

The second energy derivative is

\[
\left( \frac{\partial U}{\partial T} \right)_V = m \left( \frac{\partial u}{\partial T} \right)_\rho \tag{3.37}
\]
\[
\left( \frac{\partial u}{\partial T} \right)_p = c^0_e(T) + 1000 \rho R \left( \frac{\partial^2 Q}{\partial T^2} \right)_\rho \frac{dT}{d\tau} \\
= c^0_e(T) - \rho R T^2 \left( \frac{\partial^2 Q}{\partial T^2} \right)_\rho
\]

(3.38)

where

\[
\left( \frac{\partial^2 Q}{\partial T^2} \right)_\rho = \sum_{j=1}^{\ell} [(j-2)(j-3)(\tau - \tau_c)(\tau - \tau_q)^{j-4} + 2(j-2)(\tau - \tau_q)^{j-3}] H_j(\rho)
\]

(3.39)

The first entropy derivative, \( \left( \frac{\partial S}{\partial V}\right)_T \), is found as

\[
\left( \frac{\partial S}{\partial V} \right)_T = m \left( \frac{\partial s}{\partial V} \right)_T = m \left( \frac{\partial s}{\partial \rho} \right)_T \frac{dV}{d\rho} = -m^2 \left( \frac{\partial s}{\partial \rho} \right)_T
\]

(3.40)

\[
\left( \frac{\partial s}{\partial \rho} \right)_T = -\frac{R}{\rho} + \frac{1}{\rho^2} \left[ \rho R - \left( \frac{\partial P}{\partial T} \right)_\rho \right] = -\frac{1}{\rho^2} \left( \frac{\partial P}{\partial T} \right)_\rho
\]

(3.41)

where \( \left( \frac{\partial s}{\partial \rho} \right)_\rho \) was calculated in equation (3.10).

The second entropy derivative is

\[
\left( \frac{\partial S}{\partial T} \right)_V = m \left( \frac{\partial s}{\partial T} \right)_V
\]

(3.42)

\[
\left( \frac{\partial s}{\partial T} \right)_V = c^0_e \frac{T}{T} \left( \frac{\partial^2 Q}{\partial T^2} \right)_\rho.
\]

(3.43)

### 3.5 Stiffness Matrix Expressions for Water Liquid-Vapor Mixture (Saturation)

The \( \left( \frac{\partial^2 s}{\partial V^2} \right)_T \) derivative is found to be equal to zero, since, in this region, \( P \) is not a function of \( V \). The second pressure derivative is found as

\[
\left( \frac{dP}{dT} \right)_{\text{sat}} = P \exp \left\{ T \left( \frac{T_c}{T} - 1 \right) \sum_{i=1}^{8} F_i [a(T - T_p)]^{-1} \right\}
\]

\[
\times \left\{ -\frac{T_c}{T^2} \sum_{i=1}^{8} F_i [a(T - T_p)]^{-1} + a \left( \frac{T_c}{T} - 1 \right) \sum_{i=2}^{8} (i-1) F_i [a(T - T_p)]^{-2} \right\}
\]

(3.44)
The first energy derivative, \( \left( \frac{\partial U}{\partial V} \right)_T \), is calculated as follows.

\[
\left( \frac{\partial U}{\partial V} \right)_T = \left( \frac{\partial u}{\partial V} \right)_T = \frac{\partial}{\partial V}(u_f + xu_{fs})_T = u_{fs} \left( \frac{\partial x}{\partial V} \right)_T
\]  

(3.45)

where,

\[
x = \frac{v - v_f}{v_{fs}}
\]  

(3.46)

\[
\left( \frac{\partial x}{\partial V} \right)_T = \frac{1}{v_{fs}}
\]  

(3.47)

and so,

\[
\left( \frac{\partial U}{\partial V} \right)_T = \frac{u_{fs}}{v_{fs}} = \frac{u_{fs}}{1/\rho_s - 1/\rho_f}.
\]  

(3.48)

The second energy derivative, \( \left( \frac{\partial u}{\partial V} \right)_v \), is found as shown:

\[
\left( \frac{\partial U}{\partial T} \right)_v = m \left( \frac{\partial u}{\partial T} \right)_v = m \frac{\partial}{\partial T}(u_f + xu_{fs})_v
\]

\[
= m \left[ \frac{du_f}{dT} + x \frac{du_{fs}}{dT} + \left( \frac{\partial x}{\partial T} \right)_v u_{fs} \right]
\]  

(3.49)

where,

\[
\left( \frac{\partial x}{\partial T} \right)_v = (v - v_f) \frac{d}{dT} \left( \frac{1}{v_{fs}} \right) - \frac{dv_f}{dT} \left( \frac{1}{v_{fs}} \right)
\]

\[
= (v_f + xv_{fs} - v_f) \left[ \frac{1}{v_{fs}} \right] \frac{dv_{fs}}{dT} - \frac{dv_f}{dT} \left( \frac{1}{v_{fs}} \right)
\]

\[
= - \frac{x}{v_{fs}} \frac{dv_{fs}}{dT} - \frac{dv_f}{dT} \left( \frac{1}{v_{fs}} \right)
\]

(3.50)

\[
= - \left( \frac{dv_f}{dT} + x \frac{dv_{fs}}{dT} \right) \frac{1}{v_{fs}}
\]  

(3.51)
\frac{dv_f}{dT} = \frac{d}{dT}\left(\frac{1}{\rho_f}\right) = -\frac{1}{\rho_f^2} \frac{d\rho_f}{dT} \quad (3.52)

\frac{d\rho_f}{dT} = \rho_c \sum_{i=1}^8 \left(-\frac{iD_i}{3T_c}\right)(1 - T / T_c)^{ij-1}. \quad (3.53)

Since we do not have an explicit function for \( v_g \), we need to find a numerical approximation to \( \frac{dv_f}{dT} \). This is given by

\frac{dv_{f,k}}{dT} = \frac{dv_{f,k}}{dT} - \frac{dv_f}{dT} \quad (3.54)

where,

\frac{dv_{f,k}}{dT} = \frac{v_g(T + \Delta T) - v_g(T)}{\Delta T}. \quad (3.55)

We can calculate \( \frac{du_f}{dT} \) as,

\frac{du_f}{dT} = \frac{u_f(T + \Delta T) - u_f(T)}{\Delta T} \quad (3.56)

where \( \frac{du_f}{dT} \) is as given in equation (3.38). Similarly, for \( \frac{dv_f}{dT} \), we find that

\frac{dv_{f,k}}{dT} = \frac{u_g(T + \Delta T) - u_g(T)}{\Delta T} \quad (3.57)

\frac{du_{f,k}}{dT} = \frac{du_{f,k}}{dT} - \frac{du_f}{dT}. \quad (3.58)

The entropy derivatives are calculated in similar fashion,

\left(\frac{\partial s}{\partial v}\right)_T = \left(\frac{\partial s}{\partial v}\right)_T = \left(\frac{\partial}{\partial v} \left(s_f + x_{f,k}\right)\right)_T = s_{f,k} \left(\frac{\partial x}{\partial v}\right)_T \quad (3.59)
\[
\left( \frac{\partial S}{\partial T} \right)_v = m \left( \frac{\partial s}{\partial T} \right)_v = m \left[ s_f + x s_{fs} \right]_{s_f} \\
= m \left[ \frac{ds_f}{dT} + x \frac{ds_{fs}}{dT} + \left( \frac{\partial x}{\partial T} \right)_{s_f} s_{fs} \right] 
\] (3.60)

where,

\[
\frac{ds_f}{dT} = \frac{s_f(T + \Delta T) - s_f(T)}{\Delta T} \\
\frac{ds_s}{dT} = \frac{s_s(T + \Delta T) - s_s(T)}{\Delta T} \\
\frac{ds_{fs}}{dT} = \frac{ds_s}{dT} - \frac{ds_f}{dT}. 
\] (3.61, 3.62, 3.63)

### 3.6 Behavior of the Pressure Function

The behavior of the \( P - \rho - T \) function (Eq. (3.1)) was studied for the working range of steam (liquid, liquid-vapor and vapor regions). The function is able to describe both liquid and vapor phases, however we are also interested in looking at the behavior of the equation in the saturated steam region for the purposes of developing the CAT algorithm equations.

Equation (3.1) was plotted as a continuous function from the liquid phase up to the vapor phase. The intention was to obtain an idea of the accuracy of the function compared to the numbers in the Steam Tables, and to get a sense of the sensitivity of the function, especially in the liquid phase. We were also interested in the behavior of the function in the vicinity of the critical state.

Figure 3.1 gives an idea of the proportions of the \( P - \rho - T \) surface by showing the relative sizes of the liquid, liquid-vapor, vapor, and gas regions. Figure 3.2 zooms in the \( P - \rho - T \) plot to show the liquid region and part of the liquid-vapor region. Notice that some isotherms actually cross the specific volume axis to go into the negative pressure region before coming back and fall in place right at the line of saturated vapor states. Other isotherms cross the liquid region twice, going down and then coming back up.
This behavior has a negative effect in the performance of the Newton-Raphson iteration algorithm.

Figure 3.3 shows the behavior of the function in the vapor, gas and part of the liquid-vapor regions. The isotherms are smoother and less sensitive to changes in specific volume than the isotherms in the liquid region. This part of the $P - \rho - T$ surface is more amicable to the Newton-Raphson algorithm.
Figure 3.1: $P$-$v$-$T$ plot from equation (3.1)
Figure 3.2: P-v-T plot from equation (3.1), first half of saturated region
Figure 3.3: $P\cdot v\cdot T$ plot from equation (3.1), second half of saturation region
Part II: Software Development
4 Introduction

4.1 cT as the Development Language

The cT programming language provides end-user programmability for creating modern graphics-oriented and multimedia programs, with portability across the most popular computer operating systems; Macintosh, Microsoft Windows (DOS), and UNIX's X-Windows. cT has the mathematical capabilities of algorithmic languages such as C but also offers easy control of modern user-interface features such as graphics, mouse clicks and drags, pull-down menus, rich text, color images, and digital video.

Like C, cT offers the advantages of a modular programming language. cT programs are divided into subdivisions called “units”. Each unit has a unique name by which it is referenced. A unit may use variables which are local to that unit. Local variables are defined and are valid within each unit. If no local variables are used, all global variables are automatically available to the unit. The unit can also use both local and global variables. Modularity of the CAT program is useful to separate main algorithmic units, where the computational part of CAT is handled, from other support units, such as graphic input/output routines.

cT also offers dynamic memory allocation, in the form of dynamic arrays. In certain software applications the size of an array of variables may not be known a priori. This is the case in CAT where the size of some arrays, especially those related to the number of elements in a mesh, is not known initially. cT provides computer commands that lengthen or shorten dynamic arrays, for an efficient memory management.

The cT programming environment was developed in the Center for Design of Educational Computing at Carnegie Mellon University, in 1989. The CAT source code for steam modeling presented in this thesis was written in cT version 2.0. Currently, cT is available on MIT's Project Athena by typing:

```
add at; cT
```

at the athena% prompt.
5 The CAT Software

5.1 Organization

The CAT algorithm was programmed as an interactive computer program using the cT language. The resulting computer software is designed to help the user assemble a given thermodynamic problem as a mesh of elements, using the approach presented in Part 1, and perform a simulation to obtain a numerical solution. All of this is done through a user-friendly graphical interface consisting of windows, icons, and pull-down menus which simplifies the use of the program. The graphical interface brings the software up to the modern computer standard, which users have now come to expect.

The cT CAT source code (CAT.t) is included in magnetic media (3.5-inch floppy disk) with this thesis. The units dealing with the steam element, product of this research, are presented in Part 3 of this thesis. The CAT.t source code is divided in six major areas, these are:

1) mesh creation
2) element property input routines
3) computational algorithm
4) solution presentation
5) element constitutive relations and derivatives database
6) graphical interface support routines.
Their functionality is described in the following sections.

5.2 Mesh Creation

The initial CAT screen (refer to Figure 5.1) is divided in several areas. The left hand side of the screen contains the full library of modeling elements available in CAT, displayed as icons. The user chooses from any of them by simply clicking and dragging the desired element to the work area. The work area, is the empty middle section of the screen. To connect the elements with mechanical or thermal interconnections, the user clicks on the “Interconnections” pull-down menu from the menu bar to choose the desired type of connection. The user then clicks on the first element and drags the cursor to the second element to connect the two. The menu bar at the top of the screen offers various
other choices, such as saving or retrieving a mesh file, changing values, deleting icons, and other useful functions for generating and simulating the mesh. The status line is located at the bottom of the screen and is used to prompt and aid the user to enter data, and display error messages. Finally, the area in the right hand side of the screen is the data entry window. This is where the user initializes the mesh by entering the initial properties of the elements and interconnections.

Figure 5.1: CAT initial screen

Creation of the mesh consists of placing the desired element icons on the work area, and connect them with mechanical or thermal interconnections. While this is happening, the program checks to prevent the user from creating an unsolvable mesh. For instance, a mesh can have no more than one mechanical matching element. The program also prevents the user from connecting incompatible elements, such as thermally connecting an ideal spring to a thermal reservoir.
5.3 Input Routines

At the beginning of each run of CAT, the global variables (valid for all units) are defined and initialized in the Initial Entry Unit (IEU). The IEU refers to those commands which appear before the first unit command. After the IEU is finished, execution proceeds to the first unit in the program. The IEU is the appropriate place to define all the variables that are going to be used in all or most of the program units.

The handling and storage of the element properties is performed in arrays that are defined for each element type available in CAT. The element array variables used in the program and their storage organization are shown in Table 5.1.

<table>
<thead>
<tr>
<th>Element</th>
<th>Array variable</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thermal reservoir</td>
<td>thermresarray(i,j)</td>
<td>x_c</td>
<td>y_c</td>
<td>T</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Thermal capacity</td>
<td>caparray(i,j)</td>
<td>x_c</td>
<td>y_c</td>
<td>c_v</td>
<td>m</td>
<td>T</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Pressure reservoir</td>
<td>pressarray(i,j)</td>
<td>x_c</td>
<td>y_c</td>
<td>P</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ideal spring</td>
<td>sprarray(i,j)</td>
<td>x_c</td>
<td>y_c</td>
<td>k</td>
<td>L</td>
<td>ΔL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ideal gas</td>
<td>garray(i,j)</td>
<td>x_c</td>
<td>y_c</td>
<td>c_v</td>
<td>R</td>
<td>c_p</td>
<td>m</td>
<td>P</td>
<td>v</td>
<td>T</td>
<td>-</td>
</tr>
<tr>
<td>Steam</td>
<td>stearray(i,j)</td>
<td>x_c</td>
<td>y_c</td>
<td>x</td>
<td>ρ_f</td>
<td>ρ_g</td>
<td>m</td>
<td>P</td>
<td>v</td>
<td>T</td>
<td>x_0</td>
</tr>
</tbody>
</table>

Table 5.1: Element array organization

The first index in the array, i, denotes the element sequence number in the mesh. The second index, j, denotes the local state parameter. For example, thermresarray(2,3) is the temperature of thermal reservoir number two. Note that for all elements, the first two array values are the (x,y) screen position coordinates of the element in the mesh. These arrays house the initial values of the properties, and remain constant during the algorithm execution. The only exception is the steam element array, stearray(i,j). During execution of the algorithm, it is necessary to know the steam quality, so as to determine the steam state (liquid, liquid-vapor mixture, or vapor). Therefore, at every step of the iteration algorithm, the values of quality, x, saturated liquid density, ρ_f, and saturated vapor density ρ_g, are updated and checked for possible phase change. The initial value of quality is stored in stearray(i,10).
Once the mesh has been created, the elements and interconnections must be initialized. To do this the user clicks on the icon to open up the data entry window. The user then enters the properties required to define the element’s initial state. These properties are listed in Table 5.1. The only interconnection type that requires initializing is the mechanical interconnection. The boundary area associated with the mechanical interconnection must be entered to initialize it.

The program also verifies whether negative values for properties such as pressure or temperature were entered. In that case the user is prompted to re-enter new values.

5.4 Computational Algorithm

The mesh is solved with the Newton-Raphson algorithm as described in Section 2.5. This is carried out in the unit numerical of the CAT source code. The flowchart shown in Figure 5.2 describes the solution process used by CAT. Subroutines used by the unit numerical are included in parentheses at some of the flowchart steps. At the start of the unit, there is a first check to determine whether all elements and interconnections have been initialized. The program checks in the variable track(ij) which among other functions, keeps record of which elements have been initialized. In the unit makemats, the matrices used to setup the Newton-Raphson system equations are setup. This unit also sets the initial guess for the displacement and temperature vector, xvector(ij). The initial guess for the displacements is zero, and the initial guess for temperature is taken from the initial Ideal Gas or Steam Element temperature. If there is a thermal reservoir in the mesh, the initial guess is set to the reservoir temperature.

The residuals are calculated for the initial conditions, and then checked to see whether they are still larger than the specified tolerance (0.00003, arbitrarily set). Once the residuals are less than the tolerance, equilibrium is reached and the problem is solved. Otherwise, the program uses the Newton-Raphson algorithm to estimate a better solution set. At every step of the iteration, the program performs two checks to make sure it is converging. In the first check the program verifies the updated properties; if the volume or temperature of any element becomes negative, the program will reduce the change vector (cvector) in half and calculate a new solution vector. This is done in case the algorithm overshoots the solution vector. In the second check, the program calculates the normal of the vector containing the force and energy balance equations (vector function).
At equilibrium this vector should be a column of zeros. The program compares the new normal with the old normal from the previous iteration. If the new normal is smaller than the old normal, then iteration is proceeding in the right direction and the program resumes the iteration loop. Otherwise, the change vector is cut in half, and a new solution vector is calculated. If the change vector is cut in half for 13 times (arbitrary number), it is assumed that the equation system is just not converging, and execution of the program halts with an error message.

As a way of diagnosing the progress of the Newton-Raphson iteration, the program displays the word “Calculating” in the status line at the beginning of the numerical simulation. Then, for every successful iteration a period (“.”) is printed in the status line. For each time the change vector has to be halved because a negative temperature or volume was calculated, the program prints a question mark (“?”). Finally, every time the change vector is adjusted because the new norm is larger than the old norm, an exclamation point (“!”) is printed in the status line. The symbols get printed next to each other.
Yes
Calculate initial residuals (unit makeinitial)

No

quilibrium reached

old norm (kenorm)

j=0

Figure 5.2: Flowchart, unit numerical

Yes
Set initial guess (unit makemats)

Yes

Equilibrium reached

Residuals < 3E-5?

No

Reduce step size by half
Calculate new values

j=j+1

Yes

vol < 0
T < 0?

No

Changes are OK.
Calculate new residuals (unit makefunction)

Calculate new norm (unit makenorm)

Stop and print error

Yes

newnorm > oldnorm?

No

Calculate Stiffness (Kt) matrix (unit ktmatrix)

Solve system equations (unit ludecmp unit makeidentity unit lubksb)

Save current properties

Update element properties

Calculate old norm (unit makenorm) Set j=0

j=13?

Yes

No
5.5 Solution Presentation

After a successful run, the solution is presented in tabular form. A pop-up window appears on the screen listing all elements present in the mesh, along with their initial and final state properties. The properties change from initial to final is also calculated. A typical results window is presented in Figure 5.3. The properties listed for each element are pressure, volume, temperature, energy and entropy. S.I. units are used throughout the program.

For the case of reversible problems, the user has the option of plotting the reversible process in terms of the main properties. Prior to starting the numerical simulation, the user is asked to specify the number of intermediate states that will be used to create the plots. After a successful run, the user has the choice of plotting pressure vs. volume, pressure vs. temperature, and volume vs. temperature for any of the gases (Ideal Gas or Steam Element) in the mesh.
5.6 Relations Database

All constitutive relations listed in Table 2.1, and derivatives listed in Tables 2.3, 2.4, and 2.5 are programmed in CAT as functions. These functions are called as needed by the main numerical algorithm. The constitutive relations from Table 2.1 of pressure, energy, and entropy are programmed in the unit `makefunction` of the CAT source code. This unit contains the constitutive relations of all elements available in CAT. However, the constitutive relations of the steam element are programmed separately, since they are referenced from other units as well.

The derivatives required for the Stiffness matrix in the Newton-Raphson algorithm are also programmed in the CAT source code. The units `drdx`, `drdt`, `drEdx`, `drEdt`, `dhdx`, `dhdt`, `drSdx`, and `drSdt` correspond to the derivatives $\frac{R}{R}$, $\frac{R}{t}$, $\frac{R_x}{x}$, $\frac{R_y}{y}$, $\frac{R_z}{z}$, $\frac{d}{d}$, $\frac{S}{s}$, and $\frac{S}{t}$, respectively. Within each unit the derivative is calculated for each of CAT’s elements.

5.7 Graphical Interface Routines

One of the key features of the cT language is its ease of programming graphic tools for user-friendly displays. CAT makes use of cT’s graphic commands in setting up the screen display and handling the icons for mesh creation. Various units in the CAT program take care of the graphical management of the screen. The main units are:

- **Unit display** creates screen display and menus
- **Unit buttons** draws “SHOW” and “EXIT” screen buttons
- **Unit boxes** draws palette of element icons
- **Unit drag** allows dragging of element icons around the screen
- **Unit clean** clears status line from text
- **Unit finder** tracks mouse movement of screen cursor
- **Unit connect** handles the drawing and placement of interconnections.
6 Steam Element Software Development

6.1 Input Routine

When initializing any element in CAT, the user should be able to enter the minimum number of properties that will precisely define the state of the element. This is true for the Ideal gas element, where the user needs to enter just three of the four properties from the equation of state \((P, m, V, \text{ or } T)\) to completely define the state of the gas. The program takes care of calculating the fourth property from the other three. This feature is also desirable in the Steam element. However, steam has an additional property that the Ideal element lacks; quality. Since quality tells use whether we are dealing with a multi-phase substance or a single phase substance, it is helpful to ask the user for the value of this property first. The steam quality dictates what set of constitutive relations is appropriate to use. Recall that the common approach in modeling a pure substance is to define constitutive relations valid for a specific phase (or combination of phases) of the substance. The user may or may not know the initial steam quality, and the input algorithm should be able to handle both instances.

6.1.1 Known Quality

When the quality of steam is known, the applicable set of equations is as given in section 3.3, valid for the steam saturated region. The user can then specify either the saturation temperature or saturation pressure for the steam. If the saturation temperature is specified, the value is simply plugged in equation (3.21) to obtain the saturation pressure. If the saturation pressure is specified, the program iterates in equation (3.21) to obtain the corresponding temperature. This is done using a Newton-Raphson algorithm,

\[
T_{\text{sat}}^{k+1} = T_{\text{sat}}^k - \frac{P_{\text{sat}}(T_{\text{sat}}^k) - P_{\text{sat}}}{dP_{\text{sat}}(T_{\text{sat}}^k)/dT}.
\]  

(6.1)

The initial guess, \(T_{\text{sat}}^0\), required in the Newton-Raphson iteration is supplied by a function that approximates the unavailable function \(T_{\text{sat}} = T(P)\). Choosing several equally-spaced \((T_{\text{sat}}, P_{\text{sat}})\) points we came up with the function

\[
T_{\text{sat}}^0 = 1.33844 \times 10^{-5} P + 380.398574
\]  

(6.2)
which is simply used to obtain a better initial guess for the Newton-Raphson iteration, thereby minimizing computation time.

Once the saturation temperature and pressure have been specified, the user is given the option of entering either the steam mass or volume. At this point, the density of the steam has already been calculated as

\[ \rho = \frac{1}{\frac{1}{\rho_f} + x(\frac{1}{\rho_v} - \frac{1}{\rho_f})} \]  \hspace{1cm} (6.3)

The mass or volume can then be calculated from \( m = \rho V \). The properties that the user does not enter are calculated and displayed by the program.

**6.1.2 Unknown Quality**

The user may skip entering a value for the initial quality. When this happens there are no assumptions made about the state of the steam. Only after the user has entered some of the known properties, the program can determine the specific state of the steam. After skipping entering the initial quality value, the user is then asked to enter three of the four steam properties, \( m, P, V, \) and \( T \). All possible combinations are studied and discussed below.

• *Known \( P, V, m \)*

To calculate the missing property, temperature, the program first needs to know whether to use the \( P - \rho - T \) equation (Eq. (3.1)) valid for liquid and vapor states, or to use the \( P - T \) equation (Eq. (3.21)) for saturated steam. From the given pressure, the program calculates the saturation temperature, \( T_{sat} \), using equation (6.1). The saturated liquid density and saturated vapor density are calculated for this temperature. The density of the fluid is calculated as \( \rho = m/V \). If the density is less than \( \rho_f \) and larger than \( \rho_g \), then the steam is saturated and the calculated \( T_{sat} \) is the temperature we are seeking. Otherwise, the temperature is calculated from

\[ T^{k+1} = T^k - \frac{P(\rho, T^k) - P}{\frac{dP(\rho, T^k)}{dT}} \]  \hspace{1cm} (6.4)
which is another Newton-Raphson iteration algorithm, this one on the pressure equation 3.1.

* Known $T, V, m$
To calculate the missing property, pressure, the program first calculates the saturated liquid density and saturated vapor density for the given temperature. The density of the fluid is calculated as $\rho = m/V$. If the density is less than $\rho_g$ and larger than $\rho_f$, then the steam is saturated and pressure is calculated using equation (3.21). Otherwise, we have liquid or vapor water and the pressure is calculated from equation (3.1).

* Known $P, T, V$
To calculate the mass, the program first calculates the saturated pressure, $P_{\text{sat}}$, corresponding to the given temperature and compares it to the given pressure, $P$. If $P < P_{\text{sat}}$, the fluid is water vapor and the vapor density is calculated from equation (3.1), as

$$
\rho^{k+1} = \rho^k - \frac{P(\rho^k, T) - P}{\frac{\partial P(\rho^k, T)}{\partial \rho}}.
$$

(6.5)

If $P > P_{\text{sat}}$ the fluid is liquid water and its density is calculated the same way. The mass is calculated as $m = \rho V$. In the highly unlikely case that $P = P_{\text{sat}}$, which can only happen if the user enters the exact combination ($P_{\text{sat}}, T_{\text{sat}}$), the information is insufficient to specify the state of the fluid. The program would default to using equation (6.5) in this case, thereby calculating density as $\rho_g$.

* Known $P, T, m$
This case is similar to the previous one. After calculating the steam density, the volume comes from $V = m/\rho$.

After all properties of the steam element are known, and they place it in the saturated region, quality is calculated as

$$
x = \frac{1/\rho - 1/\rho_f}{1/\rho_g - 1/\rho_f}.
$$

(6.6)

and displayed on the computer screen. Figures 6.1 and 6.2 show the input routine decision tree.
Figure 6.1: Input routine decision tree (known quality)
Figure 6.2: Input routine decision tree (unknown quality)
6.2 Steam Functions

A library of functions was developed for calculations of steam properties. The idea was to make the whole \( P - \rho - T \) surface look continuous to the CAT algorithm, by combining equation (3.1) (for liquid and vapor) and equation (3.21) (for saturated steam). Both equations are programmed in CAT, (unit \( P \) and unit \( Psat \), respectively) and they are managed by a "central" \( P - \rho - T \) function (unit \( cP \)). This central function ascertains the steam state by verifying the steam quality and then decides which \( P - \rho - T \) function is the correct one to use. The other constitutive relations for energy and entropy are handled the same way, as well as the derivative functions for the stiffness matrix.

In order to make this technique more efficient, the quality is not calculated at each of the central units, just evaluated. Quality as a property is updated every time the temperature or volume of the steam element is updated as part of the Newton-Raphson iteration. This is done once every iteration. The information is then stored in the steam properties array (stearray) for later access by the central units. At the central units the quality of the steam is converted to an integer number: 0 for the saturated region and 1 for liquid or vapor. Quality is converted from a floating-point number to an integer number since integers are faster to evaluate in conditional commands, thereby minimizing execution time.

Following is a description of all cT CAT program functions developed to aid in the modeling of the steam element.

Central Units

unit \( cP \) is the pressure function for the steam element. From this function, the determination is made to call unit \( Psat \) for the saturated region or unit \( P \) for liquid and vapor.

unit \( cu \) is the central function for energy. This function actually calculates specific energy; the CAT algorithm takes care of later multiplying mass. The unit branches off to unit \( usat \) for the saturation conditions, or to unit \( u \) for liquid and vapor (single phase conditions).
unit cs is the central function for specific entropy. The unit branches off to unit ssat for the saturation conditions, or to unit s for liquid and vapor.

unit cdPdr is the function for \( \left( \frac{\partial e}{\partial p} \right)_T \) as required for the stiffness matrix. For saturated conditions, \( \left( \frac{\partial e}{\partial p} \right)_T \) is simply zero. For single phase conditions, the unit directs the program to unit dpdr.

unit cdPdT is the function for \( \left( \frac{\partial e}{\partial T} \right)_v \) required by the stiffness matrix. For saturated conditions the unit calls on unit dPsatdt, and for single phase conditions, unit dPdt.

unit cdudr is the function for \( \left( \frac{\partial u}{\partial T} \right)_T \). For saturated conditions the unit calculates

\[
\left( \frac{\partial u}{\partial T} \right)_T = \frac{u_g - u_f}{1/\rho_g - 1/\rho_f} \left( -\frac{1}{\rho^2} \right).
\] (6.7)

Otherwise, the program is directed to unit dudr for liquid and vapor.

unit cdudt calculates \( \left( \frac{\partial u}{\partial T} \right)_v \). For saturated steam this is computed numerically as

\[
\left( \frac{\partial u}{\partial T} \right)_v \approx \frac{u_{sat}(\rho, T + \text{TOL}) - u_{sat}(\rho, T)}{\text{TOL}}.
\] (6.8)

For liquid and vapor unit dudt is called.

unit cdsdr is the function for \( \left( \frac{\partial s}{\partial p} \right)_T \). For saturated conditions the unit calculates

\[
\left( \frac{\partial s}{\partial p} \right)_T = \frac{s_g - s_f}{1/\rho_g - 1/\rho_f} \left( -\frac{1}{\rho^2} \right).
\] (6.9)

For liquid and water the unit goes to unit dsdr.
**unit c_\text{dsdt}** calculates \( \frac{\partial s}{\partial \rho} \). For saturated steam this is computed numerically as

\[
\frac{\partial s}{\partial T} \bigg|_v \approx \frac{s_{\text{sat}}(\rho, T + \text{TOL}) - s_{\text{sat}}(\rho, T)}{\text{TOL}}.
\]  

For liquid and vapor **unit d_\text{dsdt}** is called.

---

**Saturated Steam Units**

**unit T_\text{sat}** calculates the steam saturated temperature as a function of pressure. It uses a Newton-Raphson algorithm (Eq. (6.1)) to iterate in equation (3.1). The initial guess for the iteration is supplied by equation (6.2).

**unit rho_\text{g}** calculates the saturated vapor density as a function of temperature by iterating in equation (3.1). It is equivalent to equation (6.5). The initial guess comes from the numerical approximation

\[
\rho_s = 3 \times 10^{-33} T^{12.459}
\]  

which was derived by taking several evenly-spaced \((\rho_s, T)\) points from the *Steam Tables* and using a computer-generated curve fit. The software used for this is *Microsoft Excel*.

Additional saturation functions were programmed as units, as shown in Table 6.1.

---

**Liquid and Vapor Units**

**unit T** calculates the steam temperature as a function of pressure and density. It uses a Newton-Raphson algorithm (Eq. (6.4)) to iterate in equation (3.1).

**unit Rho** calculates the steam density as a function of temperature and pressure. It also iterates in equation (3.1) using a Newton-Raphson algorithm as per equation (6.5).

Additional steam functions were programmed as units, as shown in Table 6.2.
<table>
<thead>
<tr>
<th>Function</th>
<th>Equation</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{\text{sat}}(T)$</td>
<td>Eq. (3.21)</td>
<td>unit $P_{\text{sat}}$</td>
</tr>
<tr>
<td>$dP_{\text{sat}}(T)/dT$</td>
<td>Eq. (3.44)</td>
<td>unit $dP_{\text{sat}}dT$</td>
</tr>
<tr>
<td>$u_{\text{sat}}(\rho,T)$</td>
<td>$u_{\text{sat}} = u_f + xu_{fg}$</td>
<td>unit $u_{\text{sat}}$</td>
</tr>
<tr>
<td>$u_f(T)$</td>
<td>Eq. (3.23)</td>
<td>unit $u_f$</td>
</tr>
<tr>
<td>$u_g(T)$</td>
<td>Eq. (3.28)</td>
<td>unit $u_g$</td>
</tr>
<tr>
<td>$s_{\text{sat}}(\rho,T)$</td>
<td>$s_{\text{sat}} = s_f + xs_{fg}$</td>
<td>unit $s_{\text{sat}}$</td>
</tr>
<tr>
<td>$s_f(T)$</td>
<td>Eq. (3.30)</td>
<td>unit $s_f$</td>
</tr>
<tr>
<td>$s_g(T)$</td>
<td>Eq. (3.31)</td>
<td>unit $s_g$</td>
</tr>
<tr>
<td>$\rho(x,T)$</td>
<td>$\rho = 1/[1/\rho_f + x(1/\rho_g - 1/\rho_f)]$</td>
<td>unit $\rho$</td>
</tr>
<tr>
<td>$\rho_f(T)$</td>
<td>Eq. (3.24)</td>
<td>unit $\rho_f$</td>
</tr>
<tr>
<td>$x(\rho,T)$</td>
<td>Eq. (3.25)</td>
<td>unit $x$</td>
</tr>
<tr>
<td>$d\rho_f(T)/dT$</td>
<td>Eq. (3.25)</td>
<td>unit $d\rho_fdT$</td>
</tr>
<tr>
<td>$d\rho_g(T)/dT$</td>
<td>$\frac{d\rho_g}{dT} = (\rho_g(T + \text{TOL}) - \rho_g(T))/\text{TOL}$</td>
<td>unit $d\rho_gdT$</td>
</tr>
</tbody>
</table>

Table 6.1: Saturated Steam Units
<table>
<thead>
<tr>
<th>Function</th>
<th>Equation</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(p, T)$</td>
<td>Eq. (3.1)</td>
<td>unit $P$</td>
</tr>
<tr>
<td>$u(p, T)$</td>
<td>Eq. (3.14)</td>
<td>unit $u$</td>
</tr>
<tr>
<td>$s(p, T)$</td>
<td>Eq. (3.18)</td>
<td>unit $s$</td>
</tr>
<tr>
<td>$\partial P(p, T) / \partial \rho$</td>
<td>Eq. (3.33)</td>
<td>unit $dPdr$</td>
</tr>
<tr>
<td>$\partial P(p, T) / \partial T$</td>
<td>Eq. (3.10)</td>
<td>unit $dPdT$</td>
</tr>
<tr>
<td>$\partial u(p, T) / \partial \rho$</td>
<td>Eq. (3.36)</td>
<td>unit $dudr$</td>
</tr>
<tr>
<td>$\partial u(p, T) / \partial T$</td>
<td>Eq. (3.38)</td>
<td>unit $dudT$</td>
</tr>
<tr>
<td>$\partial s(p, T) / \partial \rho$</td>
<td>Eq. (3.41)</td>
<td>unit $dsdr$</td>
</tr>
<tr>
<td>$\partial s(p, T) / \partial T$</td>
<td>Eq. (3.43)</td>
<td>unit $dsdT$</td>
</tr>
<tr>
<td>$Q(p, T)$</td>
<td>Eq. (3.2)</td>
<td>unit $Q$</td>
</tr>
<tr>
<td>$\partial Q(p, T) / \partial \rho$</td>
<td>Eq. (3.7)</td>
<td>unit $dQdr$</td>
</tr>
<tr>
<td>$\partial Q(p, T) / \partial T$</td>
<td>Eq. (3.15)</td>
<td>unit $dQdT$</td>
</tr>
<tr>
<td>$\partial^2 Q(p, T) / \partial \rho^2$</td>
<td>Eq. (3.34)</td>
<td>unit $dQdr^2$</td>
</tr>
<tr>
<td>$\partial^2 Q(p, T) / \partial T^2$</td>
<td>Eq. (3.39)</td>
<td>unit $dQdT^2$</td>
</tr>
<tr>
<td>$\partial^2 Q(p, T) / \partial \rho \partial T$</td>
<td>Eq. (3.13)</td>
<td>unit $dQd\rho dT$</td>
</tr>
<tr>
<td>$c_v(T)$</td>
<td>Eq. (3.16)</td>
<td>unit $c_v$</td>
</tr>
</tbody>
</table>

Table 6.2: Liquid and Vapor Steam Units
7 Conclusion

7.1 Potential Extension

There are many possible extensions to the work presented in this thesis. The same algorithms developed here for CAT modeling of steam can be duplicated for other substances such as refrigerants (R-12, R-13, etc.), or any other substance that typically undergoes phase changes in its application.

The computational portion of the CAT steam element computer code can also be refined. The behavior of the steam constitutive relations as given by the state equations can be studied further to better understand the Newton-Raphson iteration process in the boundaries that delimit the $P - \rho - T$ space. This way, the algorithm can be made more efficient in the case of phase changes in the substance. Further analysis of the steam element and its numerical behavior can also provide the insight required to determine a good initial guess for the main Newton-Raphson algorithm. This would reduce execution time and increase the chances of converging to a solution.

As far as the CAT program as a whole, there have been many advances and there are also many potential extensions to its capabilities. The program’s graphical interface has just recently been modernized to its present condition. Also, two new elements have been added to model finite work and finite heat transfers. And with the work presented in this thesis, CAT has been expanded to handle multi-phase pure substances. A possible addition to the present code is to model gas mixtures and chemical reactions. This will require new elements to model mass diffusion and equilibrium of chemical potentials. Another desirable feature in CAT is modeling of processes that involve time as a variable. A new heat rate interconnection will relate the energy flux between storage elements to their temperature differences. This will be a thermal interconnection with a controllable rate. Development of the thermal matching element is also in the horizon. A thermal matching element will be used to model non-adiabatic isentropic process problems in the same way the mechanical matching element is used to model adiabatic reversible problems.
7.2 Closure

This thesis presented the development of the equations, algorithm, and corresponding computer code for modeling of steam as a pure substance. The algorithm was developed to fit in the current methodology used by the Computer-Aided Thermodynamics (CAT) program.

In Part I some background information on CAT is presented, and its methodology is discussed. Following this methodology, the necessary equations were derived from the steam constitutive relations as presented by Keenan, et al., in “Steam Tables” for the liquid, vapor and saturated regions of the $P - \rho - T$ surface. In Part II, the computer algorithm is developed to enable the initialization and modeling of steam for final equilibrium and reversible process problems. Potential extensions to the steam element code, as well as the entire CAT code are also suggested. Finally in Part III, several cases are presented and solved using the software, and the corresponding computer code is presented.
Part III: Appendices
Appendix A Running CAT

A.1 Using cT and CAT.t

The CAT program featuring the steam element (CAT.t) was written in cT, a programming language created in the Center for Design of Educational Computing at Carnegie Mellon University. The CAT.t source code can be compiled and run without change in a Macintosh, MS-DOS or UNIX computer. The cT platform software for each of these machines is commercially available from Falcon Software, Inc., Box 200, Wentworth, NH 03282. The UNIX platform software is currently accessible in MIT’s Project Athena, by typing:

```
add ct; cT
```

at the `athena%` prompt. This will bring two windows to the computer screen, one for the program editor, and the second for the program execution. The CAT.t source code is publicly available in athena’s locker `/mit/jlperez/Public`, and is also included in magnetic media form (3.5-inch floppy disk) with this thesis. When the program is executed the code is compiled, producing a binary file if one does not already exist. The binary file contains instructions which the computer understands and is unique for each machine.

When testing individual units in the programming environment, there will rarely be long delays due to compilation because units are compiled one at a time as needed. However, if all units have to be compiled, compilation time can be rather long. This occurs if when new units are added or old ones deleted, or if no up-to-date binary already exists.

When a program is compiled, the conventional “.t” extension is dropped and the extension “.ctb” is added. Other extensions are preserved. On UNIX workstations, the extension becomes “.machinename.ctb”, as “sun3.ctb”.

When the program is finished, a binary can be prepared and distributed along with the freely-distributable cT Executor (the run-time package). An UNIX binary version of CAT is also available in the `/mit/jlperez/Public` locker.
Appendix B  Sample CAT Runs

B.1 Modeling a Problem in CAT

When using CAT, it is important to understand that CAT is basically a tool for number crunching. The user is expected to use his insight in thermodynamics to devise a CAT model for the physical problem at hand, and to interpret the results from the program.

To demonstrate how the user can model a thermodynamic problem using CAT and interpret the results from the program, consider the following problem:

A high temperature gas (air) is contained in a piston cylinder apparatus at a lower temperature. The cylinder is applied a force of 150 kN. We are interested in the final equilibrium states of the air and cylinder. Assume that the air can be modeled as an ideal gas with \( R = 287 \text{ J/kgK} \) and \( c_v = 716 \text{ J/kgK} \). The air is initially at a temperature of 500 K and a pressure of \( 10^5 \text{ Pa} \). The piston cylinder has a specific heat of 400 J/kgK and a mass of 3 kg and is initially at 300 K. The mass of the air in the cylinder is 1 kg, and the piston area is 1 m\(^2\). Refer to Figure B.1 for the system's physical arrangement.

![Frictionless Piston Cylinder Apparatus](image)

Figure B.1: Sample thermodynamic problem, piston-cylinder apparatus

Figure B.2 shows the sample problem modeled as a CAT mesh. After entering the initial data and running the simulation, a results window appears as in Figure B.3.
Figure B.2: CAT mesh for sample problem

<table>
<thead>
<tr>
<th>Options</th>
<th>Mesh Tools</th>
<th>Interconnections</th>
<th>Graphing</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.40 Course Software</td>
<td>Thermal Elements</td>
<td>Mechanical Elements</td>
<td>Coupled Elements</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure B.3: CAT solution to sample problem

Press your left mouse button when you are done reviewing the results.
Notice that the force on the piston was modeled as a pressure reservoir. The equivalent to a 150 kN force is a 150 kPa pressure applied over a 1 m² area. As expected, the gas and the thermal capacity (cylinder container) reach a common temperature at equilibrium. The system reaches mechanical equilibrium when the gas is compressed to the reservoir pressure.

Note that the computer screen displays an “R12” element, however it is not available for use since its equations have not yet been programmed.

B.2 Sample Problems Using the Steam Element

The following pages present a variety of problems involving the use of the steam element. Sample problems are included to show some features of the steam element applied to final equilibrium and reversible process problems. Each problem is modeled as a mesh, and the corresponding CAT results are presented.

**Problem 1: Steam Element with a Pressure Reservoir/Vapor State/Final Equilibrium**

In this problem vapor steam is expanded irreversibly to the reservoir pressure. This results in a temperature increase in the steam and decrease in its volume. The steam remains in its single phase state at equilibrium. Refer to Figures B.4 and B.5.

**Problem 2: Steam Element with a Thermal Reservoir/Saturated State/Final Equilibrium**

Saturated steam is cooled to the thermal reservoir temperature. This is a constant volume process, resulting in the decrease of the steam quality. The steam remains in its two-phase state at equilibrium. Refer to Figures B.6 and B.7.

**Problem 3: Steam Element with a Thermal Capacity/Liquid State/Final Equilibrium**

Liquid water is put in contact with a colder thermal capacity. This is also a constant volume process, so there is both a decrease in pressure and temperature. The water stays in a single phase. Refer to Figures B.8 and B.9.

**Problem 4: Steam Element with an Ideal Gas/Saturated State/Reversible Process**

Saturated steam is thermally and mechanically connected to an ideal gas element. A mechanical matching element was used on the steam to change its
temperature from 500 K to 448.15 K, in a reversible manner. The steam under-
goes the process as a two-phase mixture. Notice that the system entropy re-
 mains constant. Two plots are also included, Pressure vs. Volume and Pressure
vs. Temperature, both for steam. Refer to Figures B.10 to B.13.

**Problem 5: Steam Element /Saturated State to Vapor State/Reversible Process**

Saturated steam is compressed from 0.93 MPa to 6 MPa. The steam element is
connected to a mechanical matching element to indicate the process is a re-
versible one. During this process, the saturated steam increases its temperature
so that a phase change occurs. The final state is in the vapor region. Refer to
Figures B.14 and B.15.

**Problem 6 Steam Element with an Ideal Gas and Thermal Reservoir/Liquid State to
Saturated State/Final Equilibrium**

Liquid water is thermally and mechanically connected to an ideal gas. Both
gases are in contact with a thermal reservoir. At equilibrium both the gas and
the steam are at equal pressure and temperature. Energy is not conserved in this
problem due to the presence of the thermal reservoir. This process results in a
temperature increase for the steam, bringing the liquid into the saturated region.
Refer to Figures B.16 and B.17.
Figure B.4: CAT mesh for problem 1

Figure B.5: CAT solution to problem 1
Data for item is complete. Continue with mesh creation.

Figure B.6: CAT mesh for problem 2

Press your left mouse button when you are done reviewing the results.

Figure B.7: CAT solution to problem 2

<table>
<thead>
<tr>
<th>Element</th>
<th>Pressure(Pa)</th>
<th>Volume(m³)</th>
<th>Temperature(K)</th>
<th>Energy(J)</th>
<th>Entropy(J/K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEAM #1 (x=3, y=0.5, z=0.18)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Final</td>
<td>1170890.038</td>
<td>0.003</td>
<td>460.000</td>
<td>3345464.400</td>
<td>8551.547</td>
</tr>
<tr>
<td>Initial</td>
<td>2327074.579</td>
<td>0.003</td>
<td>500.000</td>
<td>4672480.051</td>
<td>12158.882</td>
</tr>
<tr>
<td>Final - Initial</td>
<td>-146693.441</td>
<td>0.000</td>
<td>-40.000</td>
<td>-1527015.561</td>
<td>-3175.335</td>
</tr>
<tr>
<td>THERMAL RESERVOIR #1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Final</td>
<td>-</td>
<td>-</td>
<td>460.000</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Initial</td>
<td>-</td>
<td>-</td>
<td>460.000</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Final - Initial</td>
<td>-</td>
<td>-</td>
<td>0.000</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Figure B.8: CAT mesh for problem 3

Figure B.9: CAT solution to problem 3

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Pressure (Pa)</th>
<th>Volume (m³)</th>
<th>Temperature (K)</th>
<th>Energy (J)</th>
<th>Entropy (JK)</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEAM 1 (m=10.02)</td>
<td>10236233.558</td>
<td>0.010</td>
<td>992370.291</td>
<td>3491.200</td>
<td></td>
</tr>
<tr>
<td>Initial</td>
<td>1216870.972</td>
<td>0.010</td>
<td>1113857.555</td>
<td>3898.202</td>
<td></td>
</tr>
<tr>
<td>Final - Initial</td>
<td>-1833637.415</td>
<td>0.000</td>
<td>-121497.363</td>
<td>-407.003</td>
<td></td>
</tr>
<tr>
<td>THERMAL CAPACITY 1 (c=410, m=8)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Final</td>
<td>-</td>
<td>-</td>
<td>297.042</td>
<td>974287.363</td>
<td>18575.504</td>
</tr>
<tr>
<td>Initial</td>
<td>-</td>
<td>-</td>
<td>260.000</td>
<td>652800.000</td>
<td>19239.036</td>
</tr>
<tr>
<td>Final - Initial</td>
<td>-</td>
<td>-</td>
<td>37.042</td>
<td>121497.363</td>
<td>436.068</td>
</tr>
</tbody>
</table>
**Figure B.10: CAT mesh for problem 4**

**Figure B.11: CAT solution to problem 4**

<table>
<thead>
<tr>
<th>Element</th>
<th>Pressure(Pa)</th>
<th>Volume(m$^3$)</th>
<th>Temperature(K)</th>
<th>Energy(J)</th>
<th>Entropy(J/K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAS #1</td>
<td>(cv=716, R=287, cp=1003, m=1.2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Final</td>
<td>562200.000</td>
<td>0.173</td>
<td>448.150</td>
<td>395050.168</td>
<td>4641.348</td>
</tr>
<tr>
<td>Initial</td>
<td>172200.000</td>
<td>1.000</td>
<td>500.000</td>
<td>429600.000</td>
<td>5339.591</td>
</tr>
<tr>
<td>Final - Initial</td>
<td>719800.000</td>
<td>-0.827</td>
<td>-51.850</td>
<td>-44549.831</td>
<td>-698.244</td>
</tr>
</tbody>
</table>

| STEAM #1 | (m=3.992, x=0.3, yf=0.32) |              |                |           |              |
| Final    | 862000.000  | 0.281         | 448.150         | 5310761.320| 14154.620    |
| Initial  | 569880.131  | 0.400         | 430.000         | 4919641.320| 13456.376    |
| Final - Initial | 322119.969  | -0.119 | 18.150 | 391120.000 | 688.244     |
Figure B.12: $P$ vs. $V$ plot, problem 4

Figure B.13: $P$ vs. $T$ plot, problem 4
Figure B.14: CAT mesh for problem 5

Figure B.15: CAT solution to problem 5
2.40 Course Software

Thermal Elements

Mechanical Elements

Coupled Elements

Continue with mesh

Press your left mouse button when you are done reviewing the results.

Figure B.16: CAT mesh for problem 6

<table>
<thead>
<tr>
<th>Element</th>
<th>Pressure(Pa)</th>
<th>Volume(m³)</th>
<th>Temperature(K)</th>
<th>Energy(J)</th>
<th>Entropy(JK)</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEAM #1 (m-10, x=0.111)</td>
<td>Final 6112988.143</td>
<td>0.047</td>
<td>550,000</td>
<td>136756940.11</td>
<td>33510.377</td>
</tr>
<tr>
<td>Initial</td>
<td>12169870.972</td>
<td>0.010</td>
<td>500,000</td>
<td>1113867.655</td>
<td>3898.202</td>
</tr>
<tr>
<td>Final – Initial</td>
<td>-6057472.829</td>
<td>0.037</td>
<td>250,000</td>
<td>12561825.385</td>
<td>29712.175</td>
</tr>
</tbody>
</table>

| THERMAL RESERVOIR #1 | Final | - | - | 550,000 | - |
| Initial | - | - | 550,000 | - |
| Final – Initial | - | - | 0.000 | - |

| GAS #2 (cv=718, R=287, cp=1003, m-3) | Final 6112988.143 | 0.077 | 550,000 | 1181400.000 | 11351.424 |
| Initial | 3000000.000 | 0.115 | 400,000 | 859200.000 | 11005.976 |
| Final - Initial | -0.037 | 150,000 | 322200.000 | 3455448 |

Press your left mouse button when you are done reviewing the results.

Figure B.17: CAT solution to problem 6
Appendix C  Source Code

C.1 Steam Element Computer Code

The following computer code supplies the necessary modifications and additions to the main CAT program to enable modeling of steam as a pure substance. This Appendix only list those units that were modified or added to the original code in order to program the steam element. CAT.t is a file over 150 kilobytes long, a hardcopy of the program would take in excess of 200 pages to print. Consequently, only the units that were the direct result of this research are included here. The complete CAT.t source code is included with this thesis in a 3.5-inch floppy disk.

C.1.1 Modified Units

The following are the CAT units that were modified to support steam element modeling.

Initial Entry Unit (IEU)

$syntaxlevel 2

* variable definition and initialization

define i: screenheight, screenwidth, n, no, row, a, b, p,f, flag, a3, b3, ff3, q3,x,y
     ga, sp, pr, cp, th, st, f11, f12, f13, f1, f2, f3, f4, g1, g2,
     g3,1,
     gal, spl, pr1, cp1, th1, st1, tcon, mcon, tpl, tptcon, tpmcon, mm,
     tt, doone, domnum,
     mmee, spectnum, it, gascount, fh1, fw1, fh, fw, heatee, workee, sefl, j2
     button: exit, done, spec
     screen: save, refr, uconn, current1, current2, current3, c1, c2,
     c3, under, complete, mmscr,
     mmbank, graph, grapht
     f: garray(*,*) , sprarray(*,*) , stearray(*,*) , pressarray(*,*) ,
     caparray(*,*) , thermresarray(*,*) , track(*,*) ,
     tconnections(*,*) , mconnections(*,*) , tx, ty, temptrack(*,*) ,
     temptconn(*,*) , tempmconn(*,*) ,
     mechmat(*,*) , thermat(*,*) , kmat(*,*) , indx(*,) , identity(*,*) ,
     ident(*,) , xvector(*,) , cvector(*,)
     function(*,) , initial,
     tvector(*,) , hf(3) , itvalues(*,*) , heat(*,*) , work(*,*) ,
     A(10,7) , F(8) , D(8) , G(6) , a1, Tp, Pc, Tc, rhoc, m,
calc TOL:=1E-6 $$ Tolerance for Newton-Raphson iterations
screenheight:=800
screenwidth:=1230
flag:=0
sefl:=0
l:=0
gal:=0
sp1:=0
pr1:=0
cpl:=0
th1:=0
st1:=0
tcon:=0
mcon:=0
tpl:=0
tptcon:=0
tpmcon:=0
mmee:=0
fh:=0
fw:=0
E:=4.8E-3 $$ Constants for Steam Equations
R:=461.51
al:=0.01
Tp:=338.15
Pc:=22.089E+6
Tc:=647.286
rhoc:=317

zero hf
set F:= -7.419242, 2.9721E-1, -1.155286E-1, 8.685635E-3,
    1.094098E-3, -4.39993E-3, 2.520658E-3, -5.218684E-4
set D:= 3.6711257, -28.512396, 222.6524, -882.43852,
    2000.2765, -2612.2557, 1829.7674, -533.5052
set A:= 2.9492937E-2, -5.1985860E-3, 6.8335354E-3, -1.5641040E-4,
    -6.3972405E-3, -3.9661401E-3, -6.9048554E-4,
    -1.3213917E-4, 7.779182E-6, -2.6149751E-5, -7.2546108E-7,
    2.6409282E-5, 1.5453061E-5, 2.7407416E-6,
    2.746432E-7, -3.3301902E-8, 6.5326396E-8, -9.27342989E-9,
    -4.7740374E-8, -2.9142470E-8, -5.1028070E-9,
    -3.6093828E-10, -1.6254622E-11, -2.6181978E-11, 4.3125840E-12,
    5.6323130E-11, 2.9568796E-11, 3.9636085E-12,
    3.4218431E-13, -1.7731074E-13, 0, 0, 0, 0,
    -2.4450042E-16, 1.2748742E-16, 0, 0, 0, 0,
    1.5518558E-19, 1.3746153E-19, 0, 0, 0, 0,
    5.9728487E-24, 1.5597836E-22, 0, 0, 0, 0,
    -4.1030846E-1, 3.3731180E-1, -1.3746618E-1, 6.7874983E-3,
    1.3687317E-1, 7.9847970E-2, 1.3041253E-2,
    -4.1605860E-4, -2.0988666E-4, -7.3396848E-4, 1.0401717E-5,
    6.458180E-4, 3.9917570E-4, 7.1531353E-5
set G:= 46000, 1011.249, 0.83893, -2.19989E-4, 2.46619E-7, -9.7047E-11
fine screenwidth, screenheight
rescale TRUE, TRUE,TRUE, TRUE
do main

unit drag(a,b,f) $$ Manipulation of element icons
merge, global:
i: lastx, lasty,dum,w
calc lastx:=a
  lasty:=b
dum:=0
mode xor
loop
  pause keys=touch(left: move, up)
  case zkey
    zk(left: move)
    do chooser(lastx, lasty, f)
    zk(left: up)
  outloop
  endcase
  do chooser(ztouchx, ztouchy, f)
calc lastx:=ztouchx
  lasty:=ztouchy
endloop
mode write
if (lastx<300 & lasty<350) $or$ lastx>925 $or$ lasty>640
  mode xor
  do chooser (lastx, lasty,f)
  do clean
  at 350, 720
  write Please reselect icon and place it in an appropriate area
  calc dum:=1
endif
loop w:=1,1
  if (lastx < (55+track(w,1))) & (lastx >(-55+track(w,1))) &
  (lasty < (55+track(w,2))) & (lasty>(-55+track(w,2)))
    mode xor
    do chooser (lastx, lasty,f)
    do clean
    at 350, 720
    write Please reselect icon and place it in an appropriate area
    calc dum:=1
  outloop
endif
if f=51 & dum=0 & mmee=1
  mode xor
  do chooser(lastx,lasty,f)
  do clean
  at 350,720
  write Sorry...only one MME per mesh is allowed.
  pause 3
  do clean
  write Continue with mesh.
  calc dum:=1
endif
if (f=61 $or$ f=42 $or$ f=43 $or$ f=44 $or$ f=51 $or$ f=52 $or$ f=53 $or$ f=54 $or$ f=62) & dum=0
  calc flag:=flag+1
endif
if dum=0
  calc l:=l+1
  alloc track(l,6)
calc track(l,1):=lastx
  track(l,2):=lasty
track(1,3):=f
track(1,4):=0
track(1,5):=0
if f=51
calc track(1,5):=1
  mme:=1
endif
if f!=51
do  highlight(lastx, lasty)
endif
endif
pause .001

unit row2(p)  $$ Routine to coordinate the graphical
  * functions: delete connection, delete icon,
  * move icon, freeze connection, unfreeze
  * connection
merge, global:
i: numtrack, tracker,o, kind, dx, yes, yess

case p
1  do  clean
    write Not ready yet.
2  calc yes:=0
  loop o:=1, tcon
    if tconnections(o,1) !=0
      calc yes:=1
      outloop
  endif
endloop
if yes!=1
  loop o:=1, mcon
    if mconnections(o,1)!=0
      calc yes:=1
      outloop
  endif
endloop
if yes!=1
  loop o:=1, tptcon
    if temptconn(o,1)!=0
      calc yes:=1
      outloop
  endif
endloop
if yes!=1
  loop o:=1, tmpmcon
    if tempmconn(o,1)!=0
      calc yes:=1
      outloop
  endif
endloop
if yes!=1
  do  twotwoa(100,720)
do  twotwoa(100,720)
do  clean
write Click on either the "T" or the "M" of the connection you wish to delete.

loop
  pause keys=touch(left:down)
  do findconn(ztouchx, ztouchy; kind, dx)
  if kind !=0 & dx!=0
    case kind
      1  
        zero tconnections(dx,1),16
      2  
        zero mconnections(dx, 1),17
    endcase
    outloop
  endif
  do findtempconn(ztouchx, ztouchy; kind, dx)
  if kind!=0 & dx!=0
    case kind
      1  
        zero temptconn(dx,1),17
      2  
        zero tempmconn(dx,1),18
    endcase
    outloop
  endif
endloop
do fresh
  do rebuild
  do clean
  write Connection deleted. Continue with mesh.
end
else
  do clean
  write You must first connect two icons.
endif
3
if flag>0
  calc numtrack:=0
  yes:=0
  do twotwoa(100, 720)
  do clean
  write Click on the icon you wish to delete.
  do finder(;numtrack)
  if track(numtrack, 5)=1
    case track(numtrack,3)
      42  
        zero thermresarray(track(numtrack,4),1),3
      43  
        zero caparray(track(numtrack,4), 1),5
      52  
        zero pressarray(track(numtrack,4), 1),3
      53  
        zero sprarray(track(numtrack,4), 1),5
      61  
        zero garray(track(numtrack,4),1),9
  endif
endif
62
zero stearray(track(numtrack,4), 1),9
dif
do findconn1(numtrack,1)
do findconn2(numtrack,4)
do fresh
loop tracker:=numtrack, l-1
  loop o:=1,6
    calc track(tracker,o):=track(tracker+1,o)
  endloop
del loop
  calc l:=l-1
  flag:=flag-1
  sefl:=0
  alloc track(l,6)
  loop o:=1,1
    if track(o,3)=62
      calc sefl:=1
    endif
  endloop
do rebuild
mode xor
do twon(100,720)
mode write
do clean
write Continue with mesh.
do unhigh(100,720)
else
  do clean
  write You must first place an icon.
endif
4
calc tx:=0
ty:=0
if flag>0
do clean
write Click on the icon you wish to move and drag it to the desired position.
do twon(100, 720)
loop
  pause keys=touch(left: down)
do trackfinder(ztouchx, ztouchy; yes, numtrack)
  if yes=1
    outloop
  endif
endloop
mode xor
do chooser(track(numtrack,1), track(numtrack, 2),
track(numtrack, 3))
if track(numtrack,5)=1 & track(numtrack, 3)!=51
  font zsans, 20
  at track(numtrack,1)-12, track(numtrack, 2)+5
  show track(numtrack, 4)
endif
mode write
do unhigh(track(numtrack, 1), track(numtrack,2))
mode xor
loop
    pause keys=touch(left: move, up)
    case zkey
        zk(left: move)
        if tx!0 & ty!0
            if track(numtrack,5)=1 &
                track(numtrack,3)!=51
                font zsans, 20
                at tx-12, ty+5
                show track(numtrack,4)
            endif
            do chooser(tx, ty, track(numtrack,3))
        endif
        zk(left: up)
        if track(numtrack, 5)=0
            do highlight(tx, ty)
        endif
        outloop
    endcase
    do chooser(ztouchx, ztouchy, track(numtrack, 3))
    if track(numtrack, 5)=1 & track(numtrack, 3)!=51
        font zsans, 20
        at ztouchx-12, ztouchy+5
        show track(numtrack,4)
    endif
    calc tx:=ztouchx
    ty:=ztouchy
endloop
mode write
    do findconn2(numtrack,1)
    do findconn1(numtrack,3)
    do findconn2(numtrack,2)
    do findconn1(numtrack,2)
    calc track(numtrack, 1):=tx
    track(numtrack, 2):=ty
    do fresh
    do rebuild
    do clean
    write Continue with mesh.
    mode xor
    do twotwo(100, 720)
    mode write
    do unhigh(100, 720)
else
    do clean
    write You must first place an icon.
endif
5
calc yess:=0
loop o:=1, tcon
    if tconnections(o, 1) !=0
        calc yess:=1
    outloop
endif
endloop

if yess!=1
loop o:=1, mcon
  if mconnections(o,1)!=0
    calc yess:=l
  outloop
endif
endloop
endif

if yess=l

  do clean1
    write Click on the letter of the connection that you wish to freeze.
    do clean2nd
      at 335, 750
      write (Clicking on an icon will freeze all connections to that icon.)
    do twotwob(100,720)
      loop
        pause keys=touch(left: down)
        do trackfinder(ztouchx, ztouchy;yes, numtrack)
          if yes=l
            do findconnl(numtrack,4)
          outloop
          do findconn(ztouchx, ztouchy; kind,dx)
            if kind!=0 & dx!=0
              case kind
                1
                  at tconnections(dx,7), tconnections(dx,8)
                  pattern zpatterns,3
                  disk 15
                  pattern
                  calc tptcon:=tptcon+1
                  alloc temptconn(tptcon, 17)
                  calc temptconn(tptcon, 1):=dx
                  loop o:=2,17
                    calc temptconn(tptcon,
                      o):=tconnections(dx,o-1)
                  endloop
                  zero tconnections(dx,1),16
                  outloop
                2
                  if mconnections(dx, 17)=0
                    do smallunhi(mconnections(dx,7),
                      mconnections(dx,8))
                    do conbox(mconnections(dx,7),
                      mconnections(dx,8), 2)
                    mode xor
                    box mconnections(dx,7)-12.5,
                    mconnections(dx,8)-12.5;
                    mconnections(dx,7)+12.5,
                    mconnections(dx,8)+12.5
                    mode write
                    at mconnections(dx, 7),
            endif
  endif

mconnections(dx,8)

85
pattern zpatterns, 3
disk 15
pattern
calc tpmcon:=tpmcon+1
alloc tempmconn(tpmcon, 18)
calc tempmconn(tpmcon,1):=dx
loop o:=2, 18
calc tempmconn(tpmcon,
o):=mconnections(dx, o-1)
endloop
zero mconnections(dx, 1),17
outloop
endcase
endloop
do clean1
write Connections frozen. Continue with mesh.
do clean2nd
mode xor
do twotwob(100,720)
mode write
do unhigh(100,720)
else
do clean
write You must first connect two icons.
endif
6
calc yess:=0
loop o:=1, tptcon
if temptconn(o,1)!=0
calc yess:=l
outloop
endif
endloop
if yess!=l
loop o:=1, tpmcon
if tempmconn(o,1)!=0
calc yess:=l
outloop
endif
endloop
if yess=1
do clean1
write Click on the letter of the connection you wish to
unfreeze.
do clean2nd
at 315, 750
write (Clicking on an icon will unfreeze all connections to
that icon.)
do twotwoc(100, 720)
loop
pause keys=touch(left:down)
do trackfinder(ztouchx, ztouchy;yes, numtrack)
if yes=1
do  findconn2(numtrack, 3)
  outloop
endif

do  findtempconn(ztouchx, ztouchy; kind, dx)
if  kind! =0 & dx! =0
  case    kind
  1
    loop  o:=2, 17
      calc
tconnections(temptconn(dx,1),o-1):=temptconn(dx,o)
    endloop
    zero  temptconn(dx, 1),17
    outloop
  2
    loop  o:=2,18
      calc  mconnections(tempmconn(dx,1),
o-1):=tempmconn(dx,o)
    endloop
    zero tempmconn(dx, 1),18
    outloop
  endcase
endloop

do  fresh
  do   rebuild
  do   clean2nd
  do   clean1
    write Connections have been unfrozen. Continue with mesh.
  put under; 100, 720
else
  do   clean
    write No connection is frozen. Continue with mesh.
endif
endcase

unit row6(p)$$ Routine to graphically select the Ideal
  * Gas Element, Steam Element and R12.
if  p>=20 & p<=75
  do   clean
    write Ideal Gas Element
  do  highlight(20, 280)
  do  drag(20, 280, 61)
  do  unhigh(20, 280)
  do  sixone(20, 280)
endif
if  p>=90 & p<=145
  do   clean
    write Steam Element
  do  highlight(90, 280)
  do  drag(90, 280, 62)
  do  unhigh(90, 280)
  do  sixtwo(90, 280)
endif
if  p>=160 & p<=215
  do   clean
    write Not implemented yet.
endif
unit fin
merge, global:
i: q,o
f: vol, hee, hei, entr, entropy, temp
m: text, element, tadd
edit: output
if doone=1
font zsans, 20
string text

Element Pressure(Pa) Volume(m^3) Temperature(K)
Energy(J) Entropy(J/K)

\ loop q:=1,1
if track(q,6)=1
  calc o:=track(q,4)
case track(q,3)
  51
    string element
  
  54
    string element

FINITE WORK ENERGY SOURCE #<s,o> (<s,work(o,3)> J)
\  
  44
    string element

FINITE HEAT ENERGY SOURCE #<s,o> (<s,heat(o,3)> J)
\  
  61
do ktgas(q,o;vol)
calc
  hee:=garray(o,4)*garray(o,6)*xvector(mm+1)/vol
  he:=garray(o,3)*garray(o,6)*xvector(mm+1)
  hei:=he*garray(o,9)/xvector(mm+1)
  entr:=garray(o,6)*garray(o,4)
  entro:=garray(o,6)*garray(o,3)
  string element

GAS #<s,o> (cv=<s,garray(o,3)|>, R=<s,garray(o,4)|>,
cp=<s,garray(o,5)|>, m=<s,garray(o,6)>)
Final <t,hee,6,3> <t,vol,6,3>
  <t,xvector(mm+1),6,3> <t,he,8,3>
  <t,(entr*ln(vol))+(entro*ln(xvector(mm+1))),7,3>
Initial <t,garray(o,7),6,3> <t,garray(o,8),6,3>
  <t,garray(o,9),6,3> <t,hei,6,3>
  <t,(entr*ln(garray(o,8)))+entro*ln(garray(o,9))),7,3>
Final - Initial <t,hee-garray(o,7),6,3> <t,vol-
garray(o,8),6,3> <t,xvector(mm+1)-garray(o,9),6,3>
  <t,he-hei,6,3>
\  
62
do ksteam(q,o;vol)
do cp(starray(o,6)/vol,xvector(mm+1),o;hee)
do cu(starray(o,6)/vol,xvector(mm+1),o;he)
calc he:=he*starray(o,6)
do cs(starray(o,6)/vol,xvector(mm+1),o;entr)
calc entr:=entr*starray(o,6)
tempx:=stearray(o,3)
stearray(o,3):=stearray(o,10)
do
cu(stearray(o,6)/stearray(o,8),stearray(o,9),o;hei)
calc hei:=hei*stearray(o,6)
do
cs(stearray(o,6)/stearray(o,8),stearray(o,9),o;entri)
calc entri:=entri*stearray(o,6)
stearray(o,3):=tempx
string element

STEAM #<s,0> (m=<s,stearray(o,6)>)
|, xi=<s,stearray(o,10)>
if stearray(o,10)!=-1
string tadd
append element, tadd
endif

, xf=<s,stearray(o,3)>
append element, tadd
string tadd

Final <t,hee,6,3> <t,vol,6,3>
|<t,xvector(mm+1),6,3|> <t,he,8,3> <t,entri,7,3>
Initial <t,stearray(o,7),6,3>
|<t,stearray(o,8),6,3|> <t,stearray(o,9),6,3>
|<t,hei,6,3|> <t,entri,7,3>
Final - Initial <t,hee-stearray(o,7),6,3> <t,vol-
|stearray(o,8),6,3|> <t,xvector(mm+1)-stearray(o,9),6,3|>
|<t,he-hei,6,3|> <t,entri-entri,7,3>
|3
append element, tadd

53
do ktspring(q,o;vol)
calc hee:=abs(sprarray(o,3)*vol)
hee:=.5*sprarray(o,3)*vol*vol
he:=sprarray(o,5)-sprarray(o,4)
string element

IDEAL SPRING #<s,0> (k=<s,sprarray(o,3)>, Lo=<s,sprarray(o,4)>)
Final <t,hee,6,3> <t,vol,6,3>
|<t,he,6,3|>
Initial <t,abs(sprarray(o,3)*hei),6,3|>
|<t,hei,6,3|>
|<t,.5*sprarray(o,3)*hei*hei,6,3|>
Final - Initial <t,hee-abs(sprarray(o,3)*hei),6,3|>
|<t,vol-hei,6,3|>
|<t,he-.5*sprarray(o,3)*hei*hei,6,3|>
NOTE: For the ideal spring, the pressure actually represents the absolute value of the force in newtons. Divide by the area of the particular interconnection to find the actual pressure. The volume actually represents the length of either the elongation or compression of the spring. Multiply by the area of the particular interconnection to find the actual volume.
do ktpress(q,o;vol)
calc hei:=-1000*pressarray(o,3)
he:=-vol*pressarray(o,3)

string element
PRESSURE RESERVOIR #<ls,ol>
Final  |t,pressarray(o,3),6,3|<t,vol,6,3> -
-     |t,he,6,3| -
Initial |t,pressarray(o,3),6,3|<t,1000,6,3> -
-     |t,he,6,3| -
Final - Initial |t,0,6,3|<t,vol-1000,6,3> -
-     |t,he-hei,6,3| -
\ 
42

string element
THERMAL RESERVOIR #<ls,ol>
Final  |t,thermresarray(o,3),6,3| -
-     |t, -
Initial |t,thermresarray(o,3),6,3| -
-     |t, -
Final - Initial |t,0,6,3| -
-     |t,hei,6,3| -
\ 
43
calc
he:=caparray(o,3)*caparray(o,4)*xvector(mm+1)
hei:=he*caparray(o,5)/xvector(mm+1)
entro:=caparray(o,4)*caparray(o,3)

string element
THERMAL CAPACITY #<ls,oI> (c=<ls,caparray(o,3)>, m=<js,caparray(o,4)I>)
Final |t,he,6,3|<t,entro*ln(xvector(mm+1)),6,3> -
-     |t,xvector(mm+1),6,3| -
Initial |t,hei,6,3|<t,entro*ln(caparray(o,5)),6,3> -
-     |t,xvector(mm+1)-caparray(o,5),6,3| -
|t,(entro*ln(xvector(mm+1)))-(entro*ln(caparray(o,5))),6,3| -
\ 
endcase
append text,element
endif
do clean
write Press your left mouse button when you are done reviewing the results.
pause keys=touch(left: down)
jump main
else
do clean
write You haven't solved a mesh yet.
endif
unit chooser2(a,b,f,trnum) $$ Coordinates selection of element for initialization
* 
merge, global:
i: trnum
case f
do prethermres(a, b, trnum)
do prethermcap(a, b, trnum)
do preq(a, b, trnum)
do prepress(a, b, trnum)
do prespring(a, b, trnum)
do prew(a, b, trnum)
do pregasses(a, b, trnum)
do presteam(a, b, trnum)
endcase

unit chooser3(a3, b3, ff3, numsp, tnum) $\$$ Coordinates selection of element for review and change of properties
*
* merge, global:
i:  tnum, numsp
case ff3
42  do  thermresr(a3, b3, numsp, tnum)
43  do  thermcapr(a3, b3, numsp, tnum)
44  do  heatr(a3, b3, numsp, tnum)
52  do  pressr(a3, b3, numsp, tnum)
53  do  springr(a3, b3, numsp, tnum)
54  do  workr(a3, b3, numsp, tnum)
61  do  gassesr(a3, b3, numsp, tnum)
62  do  specsteam(a3, b3, numsp, tnum)
endcase

unit numerical $\$$ The main computational algorithm for assembling the system’s equations and solving the mesh using Newton-Raphson.
*
* merge, global:
i:  q, yes,
yess,qq, o, j, stop, therms, r, sit, tres, mme, its, itindex, gasdex, aahh, violate
f:  onorm, nnorm, range, sum, vol, origf
calc yes:=1
range:=3.0E-5
tres:=0
therms:=0
mme:=0
gasdex:=0
violate:=0
loop q:=1, l
if  track(q, 5)=0
calc yes:=0
outloop
endif
endloop
if yes=1
loop q:=1,mcon
  if mconnections(q,17)=0 & mconnections(q,1)!=0
    calc yes:=0
  outloop
endif
endloop
endif
if yes=1
loop q:=1,1
  calc track(q,6):=0
endloop
do mechmatrix
do thermatrix
do checkconn
loop q:=1,1
  if track(q,3)=51 & track(q,6)=1
    calc mme:=1
    spectnum:=q
  endif
  if track(q,3)=44 & track(q,6)=1
    calc heatee:=1
  endif
  if track(q,3)=54 & track(q,6)=1
    calc workee:=1
  endif
endloop
if mme=1
do mmeh
endif
loop q:=1,1
  if track(q,3)=42 & track(q,6)=1
    calc tres:=tres+1
  endif
  if (track(q,3)=42 $or$ track(q,3)=43 $or$ track(q,3)=61 $or$ track(q,3)=62) & track(q,6)=1
    calc therms:=therms+1
  endif
endloop
if tres=0 & domnum=1 & mme=0
  calc sit:=1
  r:=mm+1
elseif tres=1 & domnum=1 & mme=0
  calc sit:=2
  r:=mm
elseif tres=0 & domnum=1 & mme=1
  calc sit:=3
  r:=mm+1
elseif tres=1 & domnum=1 & mme=1
  calc sit:=4
  r:=mm
endif
case sit
1
calc its:=1
3 calc its:=1
4 calc its:=1+it
endcase
if sit=3 $or$ sit=4
calc orighf:=hf(1)
endif
loop itindex:=1, its
if (sit=3 $or$ sit=4) & it>0
  case track(hf(2), 3)
    42 calc hf(1):=-(((caparray(track(hf(2), 4), 5)-orighf)*(itindex-1))/it) + caparray(track(hf(2), 4), 5)
    53 calc hf(1):=-(((sprarray(track(hf(2), 4), 5)-orighf)*(itindex-1))/it) + sprarray(track(hf(2), 4), 5)
    61 case hf(3)
      1 calc hf(1):=-(((garray(track(hf(2), 4), 8)-orighf)*(itindex-1))/it) + garray(track(hf(2), 4), 8)
      2 calc hf(1):=-(((garray(track(hf(2), 4), 7)-orighf)*(itindex-1))/it) + garray(track(hf(2), 4), 7)
      3 calc hf(1):=-(((garray(track(hf(2), 4), 9)-orighf)*(itindex-1))/it) + garray(track(hf(2), 4), 9)
  endcase
  62 case hf(3)
    1 calc hf(1):=-(((starray(track(hf(2), 4), 8)-orighf)*(itindex-1))/it) + starray(track(hf(2), 4), 8)
    2 calc hf(1):=-(((starray(track(hf(2), 4), 7)-orighf)*(itindex-1))/it) + starray(track(hf(2), 4), 7)
    3 calc hf(1):=-(((starray(track(hf(2), 4), 9)-orighf)*(itindex-1))/it) + starray(track(hf(2), 4), 9)
  endcase
endif
if (sit=3 $or$ sit=4) & it=0
calc hf(1):=orighf
endif
do makeinitial(sit)
do makemats(sit)
do makefunction(sit)
do clean
write Calculating
loop
if self=1
    do satcheck
endif
calc stop:=0
loop q:=1,r
    if abs(function(q))>range
        calc stop:=1
        outloop
    endif
endloop
if stop=0
    outloop
endif
do ktmatrix(sit)
do ludcmp(r)
do makeidentity(sit)
loop q:=1,r
    loop qq:=1,r
        calc ident(qq):=identity(qq)
    endloop
endoop
endoop
loop q:=1,r
calc sum:=0
loop qq:=1,r
    calc sum:=sum+identity(qq)*function(qq)
endloop
endoop
calc cvector(q):=sum
endoop
loop q:=1,r
calc tvector(q):=xvector(q)
xvector(q):=xvector(q)-cvector(q)
endoop
do makenorm(r;onorm)
calc j:=0
loop
    calc yess:=0
    loop q:=1,1
        calc o:=track(q,4)
        vol:=0
        case track(q,3)
61        do ktgas(q,o;vol)
62        do ktsteam(q,o;vol)
52        do ktpress(q,o;vol)
endcase
        if vol<0
            calc yess:=1
        endif
    endif
endloop
if yess!=1
    if xvector(mm+1)<=0
4

calc yess:=1
endif
if yess=1
loop q:=1,r
calc xvector(q):=tvector(q)-.5*cvector(q)
cvector(q):=.5*cvector(q)
endloop
write ?
calc j:=j+1
else
if sefl=1
do satcheck
endif
do makefunction(sit)
do makenorm(r;nnorm)
if nnorm>onorm
loop q:=1,r
calc xvector(q):=tvector(q)-.5*cvector(q)
cvector(q):=.5*cvector(q)
endloop
calc j:=j+1
write
else
outloop
endif
if j=13
calc yess:=-1
outloop
endif
endloop
if yess=-1
outloop
endif
write .
endloop
if yess=-1
do clean
write ERROR of the j type
else
if sit=1 $or$ sit=2 $or$ itindex=its
do clean
write Done!
if sit=3 $or$ sit=4
loop aahh:=1,l
if (track(aahh,3)=61 $or$
track(aahh,3)=62) & track(aahh,6)=1
calc gasdex:=gasdex+1
calc itvalues(gasdex,1):=aahh
case track(aahh,3)
61
do
ktgas(aahh,track(aahh,4);vol)
62
ktsteam(aahh,track(aahh,4);vol) do endcase
calc itvalues(gasdex,2):=vol
calc
itvalues(gasdex,3):=xvector(mm+1)
endif
dendloop
if workee=1 $or$ heatee=1
do secondcheck(;violate)
endif
if violate=0
calc doone:=1
do fin
else
do clean
write This system violates the second law.
Please recheck numbers.
endif
else
loop aahh:=1,1
if (track(aahh,3)=61 $or$
track(aahh,3)=62) & track(aahh,6)=1
calc gasdex:=gasdex+1
calc itvalues(gasdex,1):=aahh
case track(aahh,3)
61 do
kfgas(aahh,track(aahh,4);vol)
62 do
ktsteam(aahh,track(aahh,4);vol)
endcase
calc itvalues(gasdex,2):=vol
calc
itvalues(gasdex,3):=xvector(mm+1)
endif
dendloop
dendif
else
do clean
write Please initialize all elements.
endif

unit \( \frac{dR}{dx} \text{(dice1,dice2,there)} \) $$ Calculates \( \frac{dR}{dx} \) for each element
merge, global:
i: dice1,dice2, q, o
f: here, there, fax,vol
calc here:=0
there:=0
loop q:=1, 1
if mechmat(q,dice1)!=0 & mechmat(q,dice2)!=0
if (track(q,3)=61 $or$ track(q,3)=53 $or$ track(q,3)=62)
& track(q,6)=1
case track(q,3)
53 calc fax:=-sprarray(track(q,4),3)

61 calc o:=track(q,4)
do ktgas(q,o;vol)
calc fax:=
garray(o,4)*garray(o,6)*xvector(mm+1)/vol^2

do ktsteam(q,o;vol)
do
cdpdT(stearray(o,6)/vol,xvector(mm+1),o;fax)
calc fax:=-stearray(o,6)*fax/vol^2
endcase
cal
here:=here+(mconnections(mechmat(l+l,dice),17)*mconnections(mechmat(l+1,dice2),17)*fax*mechmat(q,dice)*mechmat(q,dice2))
endif
dfindif
dendloop
calc there:=here

unit drdt(dice;there) $$ Calculates dR/dT for each element
merge, global:
i: dice,o,q
f: here,there,fax,vol
calc here:=0
there:=0
loop q:=l,1
if mechmat(q,dice)!=0 & track(q,6)=1
calc o:=track(q,4)
case track(q,3)
61 do ktgas(q,o;vol)
calc fax:=(garray(o,4)*garray(o,6)/vol)
62 do ktsteam(q,o;vol)
do
cdpdT(stearray(o,6)/vol,xvector(mm+1),o;fax)
endcase
calc here:=here+mconnections(mechmat(l+1,dice),17)*fax*mechmat(q,dice)
endif
dendloop
calc there:=here

unit drEdx(dice;there) $$ Calculates dE/dx for each element
merge, global:
i: o,q, dice,qq
f: here,there,fax, len,vol
calc here:=0
there:=0
loop q:=l,1
if mechmat(q,dice)!=0 & (track(q,3)=53 $or$ track(q,3)=52 $or$
track(q,3)=62) & track(q,6)=1
case track(q,3)
52
loop qq:=l,mm
if mechmat(q,qq)!=0
outloop
endif
endloop
calc fax:=-pressarray(track(q,4),3)*mechmat(q,qq)
loop qq:=1,mm
    if mechmat(q,qq)!=0
        outloop
    endif
endloop
calc o:=track(q,4)
do ktspring(q,o;len)
calc fax:=sprarray(o,3)*len*mechmat(q,qq)
endcase
calc fax:=-stearray(o,6)^2*fax*mechmat(q,qq)/vol^2
endcase
calc here:=here+(mconnections(mechmat(l+l,dice),17)*fax)
endif
endloop
 calc there:=here
unit drEdt(;there) $$ Calculates dE/dT for each element
merge, global:
i: o,q
f: here, there,fax,vol
calc here:=0
    there:=0
endcase
loop q:=1,1
    if (track(q,3)=61 $or$ track(q,3)=43 $or$ track(q,3)=62) &
        track(q,6)=1
        calc o:=track(q,4)
        case track(q,3)
            43 calc fax:=caparray(o,3)*caparray(o,4)
            61 calc fax:=garray(o,3)*garray(o,6)
            62 do ktsteam(q,o;vol)
                do cdudT(stearray(o,6)/vol,xvector(mm+l),o;fax)
                calc fax:=stearray(o,6)*fax
            endcase
        endif
    endif
endloop
calc there:=here
unit dhdx(dice;there) $$ Calculates dh/dx for each element
merge, global:
i: dice, q, o
f: here, there, fax, vol
\begin{verbatim}
calc here:=0
  there:=0
if mechmat(hf(2),dice)!=0
  case track(hf(2),3)
  53  calc here:=1/mconnections(mechmat(l+l,dice),17)
  61  case hf(3)
  1    calc here:=1*mechmat(hf(2),dice)
  2    calc o:=track(hf(2),4)
do ktgas(hf(2),o;vol)
calc here:=-garray(o,4)*garray(o,6)*xvector(mm+1)/(vol*vol)*mechmat(hf(2),dice)
endcase
  62  case hf(3)
  1    calc here:=1*mechmat(hf(2),dice)
  2    calc o:=track(hf(2),4)
do ktsteam(hf(2),o;vol)
do cdPdr(stearray(o,6)/vol,xvector(mm+1),o;fax)
calc here:=-stearray(o,6)*fax/vol^2*mechmat(hf(2),dice)
endcase
  endif
  calc there:=here*mconnections(mechmat(l+l,dice),17)
endcase
unit dhdt(dice;there) $$ Calculates dh/dT for each element
merge, global:
i:dice,q,o
f:here,there,fax,vol
calc here:=0
  there:=0
case track(hf(2),3)
  43  calc here:=1
  61  case hf(3)
  2    calc o:=track(hf(2),4)
do ktgas(hf(2),o;vol)
calc here:=garray(o,4)*garray(o,6)/vol
  3    calc here:=1
endcase
  62  case hf(3)
  2    calc o:=track(hf(2),4)
do ktsteam(hf(2),o;vol)
do cdPdT(stearray(o,6)/vol,xvector(mm+1),o;here)
  3    calc here:=1
endcase
\end{verbatim}
endcase
calc there:=there

unit \text{drSdx}(\text{dice};\text{there}) \quad \text{Calculates \(dS/dx\) for each element}
merge,\text{global}:
i: o,q,dice,qq
f:here, there, vol, dsdr
calc here:=0
there:=0
loop q:=1,1
  if mechmat(q,dice)!=0 & track(q,6)=1 & (track(q,3)=61 \text{ or } track(q,3)=62)
    calc o:=track(q,4)
    loop qq:=1,mm
      if mechmat(q,qq)!=0
        outloop
      endif
    endloop
    case track(q,3)
      61
do ktgas(q,o;vol)
calc here:=here+((garray(o,4)*garray(o,6)/vol)*mechmat(q,qq)*mconnections(mechmat(1+1,dice),17))
      62
do ktsteam(q,o;vol)
do cdsdr(stearray(o,6)/vol,xvector(mm+1),o;dsdr)
calc here:=here+(dsdr*(stearray(o,6)/vol)^2*mechmat(q,qq)*mconnections(mechmat(1+1,dice),17))
      endcase
  endif
endloop
calc there:=there

unit \text{drSdt}(\text{there}) \quad \text{Calculates \(dS/dT\) for each element}
merge,\text{global}:
i: o,q
f:here, there, fax, vol
calc here:=0
there:=0
fax:=0
loop q:=1,1
  if (track(q,3)=61 \text{ or } track(q,3)=43 \text{ or } track(q,3)=62) \& track(q,6)=1
    calc o:=track(q,4)
    case track(q,3)
      43
calc fax:=caparray(o,3)*caparray(o,4)/xvector(mm+1)
      61
calc fax:=garray(o,6)*garray(o,3)/xvector(mm+1)
      62
do ktsteam(q,o;vol)
do cdsdT(stearray(o,6)/vol,xvector(mm+1),o;fax)
calc fax:=fax*stearray(o,6)
    endcase
  endif
calc here:=here+fax
endif
**unit makemats(sit)** $\$$ Indicates vectors prior to main iteration loop

```
merge, global:
i: sit, q
case sit
1,3
alloc xvector(mm+l)
alloc cvector(mm+1)
alloc tvector(mm+1)
alloc function(mm+1)
zero xvector
zero cvector
zero tvector
loop q:=1, l
  if track(q,3)=61
    calc xvector(mm+l):=garray(track(q,4), 9)
  endif
endloop
loop q:=1, l
  if track(q,3)=62
    calc xvector(mm+l):=starray(track(q,4), 9)
  endif
endloop
2,4
alloc xvector(mm+1)
alloc cvector(mm)
alloc tvector(mm)
alloc function(mm)
zero xvector
zero cvector
zero tvector
zero function
loop q:=1,l
  if track(q,3)=42
    outloop
  endif
endloop
```

Calc xvector(mm+l):=thermresarray(track(q,4),3)
endcase

**unit makefunction(sit)** $\$$ Calculates the vector function

```
merge, global:
i: sit, q,o,special
f: sum, total, len, vol, press, u, s
calc special:=0
if sit=3 $or$ sit=4
loop q:=1,mm
  if mechmat(spectnum,q)!=0
    outloop
  endif
endloop
```
calc special:=q
endif
loop q:=l,mm
calc total:=0
sum:=0
if q!=special
loop o:=1,1
if mechmat(o,q)!=0 & track(o,6)=l
  case track(o,3)
  52  
calc
  sum:=pressarray(track(o,4),3)*mechmat(o,q)
  53
  do ktspring(o,track(o,4);len)
calc sum:=-
  sprarray(track(o,4),3)*len*mechmat(o,q)/mconnections(mechmat(l+l,q),17)
  61
  do ktgas(o, track(o,4); vol)
calc
  sum:=garray(track(o,4),4)*garray(track(o,4),6)*xvector(mm+1)*mechmat(o,q)/vol
  62
  do ktsteam(o, track(o,4); vol)
do cP(stearray(track(o,4),6)/vol,
xvector(mm+1),track(o,4);press)
calc sum:=press*mechmat(o,q)
endcase
calc total:=total+sum
endif
endloop
endloop
calc function(q):=total*mconnections(mechmat(l+l,q),17)
elseif
  53
  do ktspring(o,track(o,4);len)
calc sum:=-
sprarray(track(o,4),3)*len*mechmat(o,q)/mconnections(mechmat(l+l,q),17)
  61
  do ktgas(o, track(o,4); vol)
calc
  sum:=garray(track(o,4),4)*garray(track(o,4),6)*xvector(mm+1)*mechmat(o,q)/vol
  62
  do ktsteam(o, track(o,4); vol)
do cP(stearray(track(o,4),6)/vol,
xvector(mm+1),track(o,4);press)
calc sum:=press*mechmat(o,q)
endcase
calc total:=total+sum
endif
calc function(q):=total*mconnections(mechmat(l+l,q),17)
elseif
  q=special
  43
  case track(hf(2),3)
  45  
calc function(q):=xvector(mm+1) - hf(1)
  53
  do ktspring(hf(2),track(hf(2),4);len)
calc function(q):=len-hf(1)
  61
  case hf(3)
  62  
  1  
do ktgas(hf(2),track(hf(2),4);vol)
calc function(q):=vol - hf(1)
  2  
do ktgas(hf(2),track(hf(2),4);vol)
calc function(q):=(garray(track(hf(2),4),4)*garray(track(hf(2),4),6)*xvector(mm+1)/vol) - hf(1)
  3
  calc function(q):=xvector(mm+1) - hf(1)
  endcase
  62
  case hf(3)
  62  
  1  
do ktsteam(hf(2),track(hf(2),4);vol)
calc function(q):=vol - hf(1)
  2  
do ktsteam(hf(2),track(hf(2),4);vol)

102
do
cP(stearray(track(hf(2),4),6)/vol,xvector(mm+l),track(hf(2),4);press)
calc function(q):=press - hf(1)
endcase
endcase
do
endif
calc function(q):=xvector(mm+l) - hf(1)
endcase
endloop
case sit
1
calc sum:=0
total:=0
loop q:=1,1
calc o:=track(q,4)
case track(q,3)
docase
42
calc sum:=0
43
calc sum:=caparray(o,3)*caparray(o,4)*xvector(mm+l)
44
calc sum:=0
51
do ktspring(q,o;len)
calc sum:=-.5*sprarray(o,3)*len*len
52
calc sum:=0
61
calc sum:=garray(o,3)*garray(o,6)*xvector(mm+l)
62
do ktsteam(q,o;vol)
do cu(stearray(o,6)/vol,xvector(mm+l),o;u)
calc sum:=stearray(o,6)*u
case track(q,6)
calc sum:=0
endcase
endcase
calc total:=total+sum*track(q,6)
endloop
calc function(mm+l):=total-initial
3
calc sum:=0
total:=0
loop q:=1,1
calc o:=track(q,4)
case track(q,3)
docase
42
calc sum:=0
43
calc sum:=caparray(o,3)*caparray(o,4)*ln(xvector(mm+l))
44
calc sum:=0
51
calc sum:=0
52
calc sum:=0
53      calc  sum:=0
54      calc  sum:=0
61      do  ktgas(q,o;vol)
       calc  sum:=(garray(o,6)*garray(o,4)*ln(vol)) +
              (garray(o,6)*garray(o,3)*ln(xvector(mm+1)))
62      do  ktsteam(q,o;vol)
       do  cs(stearray(o,6)/vol,xvector(mm+1),o;sum)
       calc  sum:=stearray(o,6)*sum
endcase
calc  total:=total+sum*track(q,6)
endloop
calc  function(mm+1):=total-initial
endcase

unit  secondcheck(violate) $$ Performs a check to verify second law
* of Thermodynamics is not violated
merge,global:
f: sum, total,totalheat,heatsum,vol
i: q,o,violate
calc  sum:=0
total:=0
totalheat:=0
loop q:=1,1
calc  o:=track(q,4)
heatsum:=0
case  track(q,3)
42    calc  sum:=0
43    calc  sum:=caparray(o,3)*caparray(o,4)*ln(xvector(mm+1))
44    calc  sum:=0
       heatsum:=heat(o,3)
51    calc  sum:=0
52    calc  sum:=0
53    calc  sum:=0
54    calc  sum:=0
61    do  ktgas(q,o;vol)
       calc  sum:=(garray(o,6)*garray(o,4)*ln(vol)) +
              (garray(o,6)*garray(o,3)*ln(xvector(mm+1)))
62    do  ktsteam(q,o;vol)
       do  cu(stearray(o,6)/vol,xvector(mm+1),o;sum)
       calc  sum:=stearray(o,6)*sum
endcase
calc  total:=total+sum*track(q,6)
totalheat:=totalheat+heatsum*track(q,6)
endloop
do  makeinitial(3)

104
if workee=1 & heatee=0
if total-initial<0
  calc violate:=1
endif
endif
if heatee=1
endif

unit makeinitial(sit) $$ Calculate energy, entropy summation for each element
  *merge, global:
i: q,o,sit
f: sum, total
calc sum:=0
  total:=0
if sit=1 $or$ sit=2
  loop q:=1,1
    calc o:=track(q,4)
    case track(q,3)
      42   calc sum:=0
      43   calc sum:=caparray(o,3)*caparray(o,4)*caparray(o,5)
      44   calc sum:=heat(o,3)
      52   calc sum:=-1000*pressarray(o,3)
      53   calc sum:=.5*sprarray(o,3)*(sprarray(o,5)-sprarray(o,4))*(sprarray(o,5)-sprarray(o,4))
      54   calc sum:=work(o,3)
      61   calc sum:=garray(o,3)*garray(o,6)*garray(o,9)
      62   do
cu(stearray(o,6)/stearray(o,8),stearray(o,9),o;sum)
calc sum:=sum*stearray(o,6)
endcase
calc total:=total+sum*track(q,6)
endloop
elseif sit=3 $or$ sit=4
  loop q:=1,1
    calc o:=track(q,4)
    case track(q,3)
      42   calc sum:=0
      43   calc sum:=caparray(o,3)*caparray(o,4)*ln(caparray(o,5))
      44   calc sum:=0
      51   calc sum:=0
      52   calc sum:=0
      53   calc sum:=0
endcase
endloop

105
54        calc sum:=0
61        calc
          sum:=(garray(o,6)*garray(o,4)*ln(garray(o,8)))+(garray(o,6)*garray
0,3)*ln(garray(o,9))
62        do
          cs(starray(o,6)/starray(o,8),starray(o,9),o;sum)
calc sum:=sum*starray(o,6)
endcase
calc total:=total+sum*track(q,6)
endloop
endif
calc initial:=total

unit checkcons
  $$ Generates thermal and mechanical topology matrices
  *  merge, global:
i: q,qq
loop q:=1,1
  if track(q,3)=42 $or$ track(q,3)=43 $or$ track(q,3)=44
    loop qq:=1,tt
      if thermat(q,qq)=1
          calc track(q,6):=1
          outloop
     endif
    endloop
  endif
  elseif track(q,3)=52 $or$ track(q,3)=53 $or$ track(q,3)=51
    loop qq:=1,mm
      if mechmat(q,qq)!=0
          calc track(q,6):=1
          outloop
      endif
    endloop
  elseif track(q,3)=61 $or$ track(q,3)=62
    loop qq:=1,mm
      if mechmat(q,qq)!=0
          calc track(q,6):=1
          outloop
      endif
    endloop
  endif
  if track(q,6)=0
    loop qq:=1,tt
      if thermat(q,qq)=1
          calc track(q,6):=1
          outloop
      endif
    endloop
  endif
endloop

unit mmeh
  $$ Routine to enter data for handle in the case of reversible process problems
  *  merge, global:
i:q,in,yes,decis
calc decis:=0
if hf(1)! = 0
get mmscr;400,325;820,475
put mmblank;400,325
box 400,325;820,475
box 715,435;750,470
box 760,435;795,470
font zsans,25
at 405,330
write Do you wish to use your previous final condition?
at 715,443
write YES
at 764,443
write NO
at 405,380
case track(hf(2),3)
43
    write Thermal Capacity
    show track(hf(2),4)
    write : Final Temperature =
    show hf(1)
    write K
53
    write Ideal Spring
    show track(hf(2),4)
    write : Final Elongation =
    show hf(1)
    write m
61
    write Ideal Gas
    show track(hf(2),4)
    write :
    case hf(3)
1
        write Final Volume =
        show hf(1)
        write m^3
2
        write Final Pressure =
        show hf(1)
        write Pa
3
        write Final Temperature =
        show hf(1)
        write K
endcase
62
write Steam
show track(hf(2),4)
write :
    case hf(3)
1
        write Final Volume =
        show hf(1)
        write m^3
2
        write Final Pressure =
        show hf(1)
        write Pa
3
write Final Temperature =
show hf(1)
write K
dcase
endcase
loop
pause keys=touch(left:down)
if ztouchx<=750 & ztouchx>=715 & ztouchy<=470 &
ztouchy>=435
calc decis:=1
elseif ztouchx<=795 & ztouchx>=760 & ztouchy>=435 &
ztouchy<=470
calc decis:=2
endif
if decis!=0
outloop
endif
endloop
put mmscr;400,325
endif
if decis!=1
do clean1
write MME in mesh...Click on the icon whose final state you wish
to determine.
do fiveone(100,720)
loop
pause keys=touch(left:down)
do trackfinder(ztouchx,ztouchy;yes,in)
if yes=1
case track(in,3)
43
do capmmeh(in)
outloop
53
do springmmeh(in)
outloop
61
do gasmmeh(in)
outloop
62
do steammmeh(in)
outloop
endcase
endloop
mode xor
mode write
do fiveone(100,720)
do unhigh(100,720)
do clean2nd
do clean1
write Final state for reversible process chosen!
pause .5
endif
loop
get mmscr;400,325;800,475
put mmblank;400,325
box 400,325;800,475
at 410,370
font zsans,25
write Enter the number of intermediate states:
compute it
ok zreturn
endarrow
calc gascount:=0
loop q:=1,1
   if (track(q,3)=61$or$ track(q,3)=62) & track(q,6)=1
      calc gascount:=gascount+1
   endif
endloop
alloc itvalues((it+l)*gascount,3)
if it<0
   do clean
      write Please enter a positive number!
   else
      outloop
   endif
endloop

unit record $$ Saves mesh data to a file
merge, global:
i: q,o
button: can
do clean
write Save Mesh Selected...
addfile saved;zempty
if zreturn!=17 & zreturn!=18 & zreturn!=19
dataout saved;l
   loop q:=1,1
      loop o:=1,6
         dataout saved;track(q,o)
      endloop
   endloop
dataout saved;gal
   loop q:=1,gal
      loop o:=1,9
         dataout saved;garray(q,o)
      endloop
   endloop
dataout saved;fh1
   loop q:=1,fh1
      loop o:=1,3
         dataout saved;heat(q,o)
      endloop
   endloop
dataout saved;fw1
   loop q:=1,fw1
      loop o:=1,3
         dataout saved;work(q,o)
      endloop
   endloop
dataout saved;th1
   loop q:=1,th1
      loop o:=1,th1
         dataout saved;work(q,o)
      endloop
endloop
loop o:=1,3
dataout saved;thermresarray(q,o)
endloop
dataout saved;prl
loop q:=1,prl
loop o:=1,3
dataout saved;pressarray(q,o)
endloop
endloop
dataout saved;cpl
loop q:=1,cpl
loop o:=1,5
dataout saved;caparray(q,o)
endloop
endloop
dataout saved;stl
loop q:=1,stl
loop o:=1,10
dataout saved;starray(q,o)
endloop
endloop
dataout saved;spl
loop q:=1,spl
loop o:=1,5
dataout saved;sprarray(q,o)
endloop
endloop
dataout saved;tcon
loop q:=1,tcon
loop o:=1,16
dataout saved;tconnections(q,o)
endloop
endloop
dataout saved;mcon
loop q:=1,mcon
loop o:=1,17
dataout saved;mconnections(q,o)
endloop
endloop
dataout saved;tptcon
loop q:=1,tptcon
loop o:=1,17
dataout saved;temptconn(q,o)
endloop
endloop
dataout saved;tpmcon
loop q:=1,tpmcon
loop o:=1,18
dataout saved;tempmconn(q,o)
endloop
endloop
doi clean
write File Saved!!
setfile saved
pause 1.5
do clean
endif
do clean

unit clearall

* 
alloc track(0,0) 
alloc garray(0,0) 
alloc sarray(0,0) 
alloc sprarray(0,0) 
alloc caparray(0,0) 
alloc pressarray(0,0) 
alloc heat(0,0) 
alloc work(0,0) 
alloc thermresarray(0,0) 
alloc tconnections(0,0) 
alloc mconnections(0,0) 
alloc temptconn(0,0) 
alloc tempmconn(0,0) 
zero hf 
calc ga:=0 
    st:=0 
    sp:=0 
    pr:=0 
    cp:=0 
    th:=0 
    fh:=0 
    fw:=0 
    gal:=0 
    spl:=0 
    prl:=0 
    cpl:=0 
    thl:=0 
    fhl:=0 
    fwl:=0 
    stl:=0 
    mmee:=0 
    l:=0 
    tpmcon:=0 
    tptcon:=0 
    tcon:=0 
    mcon:=0 
    flag:=0 
    tpl:=0 
    sefl:=0 
do fresh 
do rebuild

unit getfile

merge,global: 
i: o,q 
button: can 
do clean 
write Open Mesh Selected... 
setfile toget;zempty;ro 
if zreturn!=17 & zreturn!=18 & zreturn!=19 
do clearall 
do clean 
write file: 
show zfilename(toget)
datain toget;1
alloc track(1,6)
loop q:=1,1
  loop o:=1,6
datain toget;track(q,o)
endloop
endloop
datain toget;gal
alloc garray(gal,9)
loop q:=1,gal
  loop o:=1,9
datain toget;garray(q,o)
endloop
endloop
datain toget;fh1
alloc heat(fh1,3)
loop q:=1,fh1
  loop o:=1,3
datain toget;heat(q,o)
endloop
endloop
datain toget;fw1
alloc work(fw1,3)
loop q:=1,fw1
  loop o:=1,3
datain toget;work(q,o)
endloop
endloop
datain toget;th1
alloc thermresarray(th1,3)
loop q:=1,th1
  loop o:=1,3
datain toget;thermresarray(q,o)
endloop
endloop
datain toget;pr1
alloc pressarray(pr1,3)
loop q:=1,pr1
  loop o:=1,3
datain toget;pressarray(q,o)
endloop
endloop
datain toget;cpl
alloc caparray(cpl,5)
loop q:=1,cpl
  loop o:=1,5
datain toget;caparray(q,o)
endloop
endloop
datain toget;stl
alloc stearray(stl,10)
loop q:=1,stre
  loop o:=1,10
datain toget;stearray(q,o)
endloop
endloop
if stl>0
calc sefl:=1
endif
datain toget; spl
alloc sprarray(spl, 5)
loop q:=1, spl
  loop o:=1, 5
datain toget; sprarray(q, o)
endloop
endloop
datain toget; tcon
alloc tconnections(tcon, 16)
loop q:=1, tcon
  loop o:=1, 16
datain toget; tconnections(q, o)
endloop
endloop
datain toget; mcon
alloc mconnections(mcon, 17)
loop q:=1, mcon
  loop o:=1, 17
datain toget; mconnections(q, o)
endloop
endloop
datain toget; tptcon
alloc temptconn(tptcon, 17)
loop q:=1, tptcon
  loop o:=1, 17
datain toget; temptconn(q, o)
endloop
endloop
datain toget; tpmcon
alloc tempmconn(tpmcon, 18)
loop q:=1, tpmcon
  loop o:=1, 18
datain toget; tempmconn(q, o)
endloop
endloop
loop q:=1, 1
  if track(q, 3)=51 & track(q, 6)=1
    calc mmee:=1
  endif
  if track(q, 3)=44 & track(q, 6)=1
    calc heatee:=1
  endif
  if track(q, 3)=54 & track(q, 6)=1
    calc workee:=1
  endif
endloop
setfile toget
calc flag:=2
do fresh
do rebuild
endif
if zreturn=17 $or$ zreturn=18 $or$ zreturn=19
  do clean
endif
unit pvv

\$\$ Generates P vs. V plot (reversible case)

merge, global:
i: q, yes, in, o
f: vol, pressure, scaldif, spress

pattern
if doone=l & mmee>0 & it>0
loop
  do clean1
  write Please click on the gas whose graph you wish to see.
  pause keys=touch(left:down)
  do trackfinder(ztouchx, ztouchy; yes, in)
  if yes=l & (track(in,3)=61 $or$ track(in,3)=62) outloop
  endif
  endloop

font zsans, 30
get grapht; 0, 0; 1230, 800
put graph; 0, 0
box 220, 0; 980, 700
at 450, 5
write Pressure (Pa) vs. Volume (m^3)
font zsans, 20
gorigin 320, 630
axes 560, 560

if track(in,3) = 61
  do ktgas(in, track(in,4); vol)
calc o := track(in,4)
  if vol > garray(o,8)
    scalex vol+.25, garray(o,8)-.25
  else
    scalex garray(o,8)+.25, vol-.25
  endif
  calc pressure := garray(o,6)*garray(o,4)*xvector(mm+l)/vol
  if pressure > garray(o,7)
    scaley pressure+10000, garray(o,7)-10000
  else
    scaley garray(o,7)+10000, pressure-10000
  endif
  labelx .20, .05
  calc scaldif := abs(pressure-garray(o,7))
  if scaldif < 200000
    labely 25000, 5000
  elseif scaldif > 200000 & scaldif < 500000
    labely 100000, 20000
  elseif scaldif > 500000
    labely 200000, 50000
  endif
  gat garray(o,8), garray(o,7)
  loop q:=1, (it+l)*gascount
    if itvalues(q,1)=in
      gdraw ; itvalues(q,2),
garray(o,6)*garray(o,4)*itvalues(q,3)/itvalues(q,2)
disk 3
gat
    itvalues(q,2), garray(o,6)*garray(o,4)*itvalues(q,3)/itvalues(q,2)
  endif
endloop
do  
ktsteam(in,track(in,4);vol)
calc  
o:=track(in,4)
if  
vol>starray(o,8)  
scalex  
vol+.25,starray(o,8)-.25
else  
scalex  
starray(o,8)+.25,vol-.25
endif  
do  
cP(starray(o,6)/vol,xvector(mm+l),o;pressure)
if  
pressure>starray(o,7)  
scaley  
pressure+10000,starray(o,7)-10000
else  
scaley  
starray(o,7)+10000,pressure-10000
endif  
labelx  
.20,.05  
calc  
scaldif:=abs(pressure-starray(o,7))
if  
scaldif<200000  
lably  
25000,5000
elseif  
scaldif>200000 & scaldif<500000  
lably  
100000,20000
elseif  
scaldif>500000  
lably  
200000,50000
endif  
gat  
starray(o,8),starray(o,7)
loop  
q:=1,(it+l)*gascount  
if  
itvalues(q,1)=in  
do  
cP(starray(o,6)/itvalues(q,2),itvalues(q,3),o;spress)
gdraw ;itvalues(q,2),spress
disk 3  
do  
cP(starray(o,6)/itvalues(q,2),itvalues(q,3),o;spress)
gat  
itvalues(q,2),spress
enddo  
endcase  
endloop  
do  
clean
write  
When done reviewing graph, press left mouse button.
pause  
keys=touch(left:down)
put  
graph;0,0
do  
clean
else  
if  
it=0 & doone=1 & mmee>0  
do  
clean
write  
No intermediate states were chosen.
else  
do  
clean
write  
You haven't solved a reversible process mesh yet.
endif  
endif  

unit  
pvt  
$$ Generates P vs. T plot (reversible case)  
merge, global:  
i:  
q,yes,in,o  
f:  
vol,pressure,scaldif,spress
pattern  
if  

doone=1 & mmee>0 & it>0
loop
do clean1
write Please click on the gas whose graph you wish to see.
pause keys=touch(left:down)
do trackfinder(ztouchx,ztouchy;yes,in)
if yes= & (track(in,3)=61 $or$ track(in,3)=62)
outloop
endif
endloop
font zsans,30
get grapht;0,0;1230,800
put graph;0,0
box 220,0;980,700
at 450,5
write Pressure (Pa) vs. Temperature (K)
font zsans,20
gorigin 320,630
axes 560,560
calc o:=track(in,4)
case track(in,3)
61
do ktgas(in,o;vol)
if xvector(mm+1)>garray(o,9)
scalex xvector(mm+1)+10,garray(o,9)-10
else
scalex garray(o,9)+10,xvector(mm+1)-10
endif
calc pressure:=garray(o,6)*garray(o,4)*xvector(mm+1)/vol
if pressure>garray(o,7)
scaley pressure+10000,garray(o,7)-10000
else
scaley garray(o,7)+10000,pressure-10000
endif
labelx 25,5
calc scaldif:=abs(pressure-garray(o,7))
if scaldif<200000
labely 25000,5000
elseif scaldif>200000 & scaldif<500000
labely 100000,20000
elseif scaldif>500000
labely 200000,50000
endif
gat garray(o,9),garray(o,7)
loop q:=1,(it+l)*gascount
if itvalues(q,1)=in
gdraw ;itvalues(q,3),
garray(o,6)*garray(o,4)*itvalues(q,3)/itvalues(q,2)
disk 3
gat
itvalues(q,3),garray(o,6)*garray(o,4)*itvalues(q,3)/itvalues(q,2)
endif
endloop
62
do ktsteam(in,o;vol)
if xvector(mm+1)>starray(o,9)
scalex xvector(mm+1)+10,starray(o,9)-10
else
scalex starray(o,9)+10,xvector(mm+1)-10
endif
do  cP(stearray(o,6)/vol,xvector(mm+1),o;pressure)
if  pressure>stearray(o,7)
  scaley pressure+10000,stearray(o,7)-10000
else
  scaley stearray(o,7)+10000,pressure-10000
endif
labelx 25,5
calc scaldif:=abs(pressure-stearray(o,7))
if  scaldif<200000
  labely 25000,5000
elseif scaldif>200000 & scaldif<500000
  labely 100000,20000
elseif scaldif>500000
  labely 200000,50000
endif
gat stearray(o,9),stearray(o,7)
loop q:=l,(it+l)*gascount
  if itvalues(q,1)=in
do
    cP(stearray(o,6)/itvalues(q,2),itvalues(q,3),o;spress)
gdraw ;itvalues(q,3),spress
disk 3
do
  cP(stearray(o,6)/itvalues(q,2),itvalues(q,3),o;spress)
gat itvalues(q,3),spress
endif
endloop
endcase
do cleanl
do cleanl
write When done reviewing graph, press left mouse button.
pause keys=touch(left:down)
put grapht;0,0
do cleanl
else
  if doone=l & mmee>0 & it=0
do clean
  write No intermediate states were chosen.
else
  do cleanl
  write You haven't solved a reversible process mesh yet.
endif
endif
unit vvt
$$ Generates V vs. T plot (reversible case)
merge, global:
i: q,yes,in,o
f: vol,pressure,scaldif
pattern
if  doone=1 & mmee>0 & it>0
loop
  do cleanl
  write Please click on the gas whose graph you wish to see.
pause keys=touch(left:down)
do trackfinder(ztouchx,ztouchy;yes,in)
  if yes=1 & (track(in,3)=61 $or$ track(in,3)=62)
    outloop
  endif
endloop
font zsans,30
get grapht;0,0;1230,800
put graph;0,0
box 220,0;980,700
at 450,5
write Volume (m^3) vs. Temperature (K)
font zsans,20
gorigin 320,630
axes 560,560
calc o:=track(in,4)
case track(in,3)
61
  do ktgas(in,track(in,4);vol)
  if xvector(mm+1)>garray(o,9)
    scalex xvector(mm+1)+10,garray(o,9)-10
  else
    scalex garray(o,9)+10,xvector(mm+1)-10
  endif
  if vol>garray(o,8)
    scaley vol+.25,garray(o,8)-.25
  else
    scaley garray(o,8)+.25,vol-.25
  endif
  labelx 25,5
  labely .2,.05
  gat garray(o,9),garray(o,8)
62
  do ktsteam(in,track(in,4);vol)
  if xvector(mm+1)>starray(o,9)
    scalex xvector(mm+1)+10,starray(o,9)-10
  else
    scalex starray(o,9)+10,xvector(mm+1)-10
  endif
  if vol>starray(o,8)
    scaley vol+.25,starray(o,8)-.25
  else
    scaley starray(o,8)+.25,vol-.25
  endif
  labelx 25,5
  labely .2,.05
  gat starray(o,9),starray(o,8)
endcase
loop q:=1,(it+1)*gascount
  if itvalues(q,1)=in
    gdraw ;itvalues(q,3),itvalues(q,2)
    disk 3
    gat itvalues(q,3),itvalues(q,2)
  endif
endloop
do cleanl
write When done reviewing graph, press left mouse button.
pause keys=touch(left:down)
put grapht;0,0
do cleanl
else
  if doone=1 & mmee>0 & it=0
    do clean
    write No intermediate states were chosen.
  endif
else
do   cleanl
write You haven't solved a reversible process mesh yet.
endif
endif
C.1.2 New Units

The following are the CAT units that were added to the original code to support steam element modeling.

**unit presteam(a,b,trnum)**  
$$\text{Initialization of the vector stearray}$$  
merge, global:  
i: init, trnum  
calc stl:=stl+1  
alloc stearray(stl, 10)  
zero stearray(stl,1),10  
at a-12, b+5  
font zsans, 20  
show stl  
calc track(trnum, 4):=stl  
do specsteam(a,b, stl, trnum)

**unit specsteam(a,b,st,tnum)**  
$$\text{Steam quality input routine}$$  
Specify whether is single or two-phase  
merge,global:  
f: nosig  
i: tnum, def  
calc fll:=0  
fl2:=0  
fl3:=0  
def:=1  
calc stearray(st,1):=a  
stearray(st,2):=b  
do highlight(a,b)  
do unhigh(20, 280)  
do sixone(20, 280)  
do clean2  
write Steam Properties  
show st  
font zsans, 25  
at 985, 70  
write x:  
font zsans, 18  
if stearray(st, 3) != 0  
at 1045, 90  
write {  
show stearray(st, 3)  
write }
endif  
calc def:=1  
do clean1  
write Please enter the value of the steam quality (if known).  
do clean2nd  
at 405, 750  
write Press return to skip entering this value.  
font zsans, 25  
mode rewrite  
at 1030, 70  
write  
mode write
unit specsteam2(tnum) \$\$ Input routine for the two-phase steam case
merge, global:
f: nospign,dens
i: tnum, def
calc f1:=0
f2:=0
put uconn; 980, 255
font zsans, 30
at 985, 150
write Initial State
show st
font zsans, 20
at 985, 210
write Enter one of two.
font zsans, 25
at 985, 245
write p (Pa):
at 985, 280
write T (K):
font zsans,20
at 985, 320
write Enter one of two.
font zsans,25
at 985, 355
write v (m^3):
at 985,390
write m (kg):
font zsans, 18
if stearray(st, 7) != 0
at 1045, 265
write { show stearray(st, 7) write }
endif
if stearray(st, 9) !=0
at 1045, 300
write { show stearray(st, 9) write }
endif
if stearray(st, 8) != 0
at 1045, 375

write {  
  show stearray(st, 8)  
write }
endif
if stearray(st, 6) != 0
  at 1045, 410
write {
  show stearray(st, 6)
write }
endif
loop
calc def:=1
do clean1
write Please enter the value of the initial pressure, \( p \), in Pa.
do clean2nd
at 355, 750
write Press return to skip entering this value.
font zsans, 25
mode rewrite
at 1080, 245
write
mode write
allow blanks
arrow 1080, 245
ok zjcount=0
calc def:=0
compute nosign
ok zreturn
endarrow
do clean2nd
if def!=0
calc stearray(st, 7):=nosign
  f1:=1
do Tsat(nosign;stearray(st,9))
mode rewrite
at 1080,280
write \lt\{s, stearray(st,9)\gt\>
outloop
endif
calc def:=1
do clean1
write Please enter the value of the initial temperature, \( T \), in K
do clean2nd
at 395, 750
write Press return to skip entering this value.
font zsans, 25
at 1080, 280
mode rewrite
write
mode write
allow blanks
arrow 1080, 280
ok zjcount=0
calc def:=0
compute nosign
ok zreturn
endarrow
do clean2nd
if def!=0
calc stearray(st,9):=nosign
f2:=1
do Psat(nosign;stearray(st,7))
mode rewrite
at 1080,245
write <|s,stearray(st,7)|>
outloop
endif
endloop
loop
calc def:=1
do cleanl
write Please enter the value of the initial volume, v, in m^3.
do clean2nd
at 395, 750
write Press return to skip entering this value.
font zsans, 25
at 1080, 355
mode rewrite
write
mode write
arrow 1080, 355
ok zjcount=0
calc def:=0
compute nosign
ok zreturn
endarrow
do clean2nd
if def!=0
calc stearray(st, 8):=nosign
f1:=1
do rho(stearray(st,3),stearray(st,9);dens)
calc m:=dens*stearray(st,8)
mode rewrite
at 1080,390
write <|s,m|>
calc stearray(st,6):=m
outloop
endif
calc def:=1
do cleanl
write Please enter the value of the mass,m, in kg
do clean2nd
at 395, 750
write Press return to skip entering this value
font zsans, 25
at 1080, 390
mode rewrite
write
mode write
arrow 1080, 390
ok zjcount=0
calc def:=0
compute nosign
ok zreturn
endarrow
do clean2nd
if def!=0
calc stearray(st,6):=nosign
    f2:=1
do rho(stearray(st,3),stearray(st,9);dens)
calc vlm:=stearray(st,6)/dens
mode rewrite
at 1080,355
write <js,vlm>
calc stearray(st,8):=vlm
outloop
endif
endloop
do unhigh(stearray(st,1), stearray(st,2))
do sixtwo(stearray(st,1), stearray(st,2))
do cleanl
write Data for item is complete. Continue with mesh creation.
do rhof(stearray(st,9);stearray(st,4))
do rhog(stearray(st,9);stearray(st,5))
calc track(tnum, 5):=1
    sefl:=1
    do mesh
unit specsteam3(tnum) $$ Input routine for the unknown steam quality case
* 
    merge, global:
    f:  nosign
    i:  tnum, def
    calc f1:=0
    f2:=0
    f3:=0
    f4:=0
    put uconn; 980, 255
font zsans, 30
at 985, 150
write Initial State
show st
font zsans, 20
at 985, 210
write Enter three of four.
font zsans, 25
at 985, 245
write m (kg):
at 985, 280
write p (Pa):
at 985, 315
write v (m^3):
at 985, 360
write T (K):
if stearray(st, 6) != 0
    at 1045, 265
    write {
        show stearray(st,6)
        write }
endif
if stearray(st, 7) !=0
    at 1045, 300
    write {
        show stearray(st, 7)
write }
endif
if stearray(st, 8) != 0
  at 1045, 335
  write {
    show stearray(st, 8)
    write }
endif
if stearray(st, 9) != 0
  at 1045, 380
  write {
    show stearray(st, 9)
    write }
endif
loop
  if f1=0 & (f2=0 $or$ f3=0 $or$ f4=0)
    calc def:=1
    do clean1
    write Please enter the value of the mass, m, in kg.
    do clean2nd
    at 355, 750
    write Press return to skip entering this value.
    font zsans, 25
    mode rewrite
    at 1080, 245
    write
    mode write
    allow blanks
    arrow 1080, 245
    ok zjcount=0
    calc def:=0
    compute nosign
    ok zreturn
darrow
    do clean2nd
    if def!=0
      calc stearray(st, 6):=nosign
      f1:=1
    endif
  endif
  if (f1=1 & f2=1 & f3=1) $or$ (f1=1 & f2=1 & f4=1) $or$ (f1=1 & f3=1 & f4=1) $or$ (f2=1 & f3=1 & f4=1)
    outloop
  endif
  if f2=0 & (f1=0 $or$ f3=0 $or$ f4=0)
    calc def:=1
    do clean1
    write Please enter the value of the initial pressure, p, in Pa.
    do clean2nd
    at 395, 750
    write Press return to skip entering this value.
    font zsans, 25
    at 1080, 280
    mode rewrite
    write
    mode write
allow blanks
arrow 1080, 280
ok zjcount=0
calc def:=0
compute nosign
ok zreturn
endarrow
do clean2nd
if def!=0
calc stearray(st, 7):=nosign
  f2:=1
endif
endif
if (fl=1 & f2=1 & f3=1) $or$ (f1=1 & f2=1 & f4=1) $or$ (f1=1 & f3=1 & f4=1) $or$ (f2=1 & f3=1 & f4=1)
  outloop
endif
if f3=0 & (f1=0 $or$ f2=0 $or$ f4=0)
calc def:=1
do cleanl
  write Please enter the value of the initial volume, v, in m^3.
do clean2nd
at 395, 750
write Press return to skip entering this value.
font zsans, 25
at 1080, 315
mode rewrite
write
mode rewrite
arrow 1080, 315
ok zjcount=0
calc def:=0
compute nosign
ok zreturn
endarrow
do clean2nd
if def!=0
calc stearray(st, 8):=nosign
  f3:=1
endif
endif
if (f1=1 & f2=1 & f3=1) $or$ (f1=1 & f2=1 & f4=1) $or$ (f1=1 & f3=1 & f4=1) $or$ (f2=1 & f3=1 & f4=1)
  outloop
endif
if f4=0 & (f1=0 $or$ f2=0 $or$ f3=0)
calc def:=1
do clean1
  write Please enter the value of the initial temperature, T, in K.
do clean2nd
at 395, 750
write Press return to skip entering this value.
font zsans, 25
at 1080, 360
mode rewrite
write
mode write
arrow 1080, 360
ok zjcount=0
calc def:=0
compute nosign
ok zreturn
endarrow
do clean2nd
if def=0
calc stearray(st, 9):=nosign
f4:=1
endif
endif
if (fl=l & f2=l & f3=l) $or$ (f1=l & f2=l & f4=l) $or$ (f1=l & f3=l & f4=l) $or$ (f2=l & f3=1 & f4=l) outloop
endif
endloop
do steamcomp3(f1, f2, f3, f4, tnum, 1)

unit steamcomp3(ss1, ss2, ss3, ss4, tnum, type) $$ \text{Routine to calculate missing properties for the unknown steam quality case}$$
* * *
merge, global:
i: tnum, type, ss1, ss2, ss3, ss4
f: mass, dens, temp, pres, rhg, rhf
if ss1=0
do Rho(stearray(st, 7), stearray(st, 9); dens)
calc stearray(st, 6):=dens*stearray(st, 8)
at 1085, 245
font zsans, 25
mode rewrite
write
mode write
at 1085, 245
show stearray(st, 6)
if stearray(st, 9) < Tc
do rhog(stearray(st, 9); stearray(st, 5))
do rhof(stearray(st, 9); stearray(st, 4))
calc stearray(st, 3):=(1/dens-1/stearray(st, 4))/(1/stearray(st, 5)-1/stearray(st, 4))
else
calc stearray(st, 3):=-1
endif
endif
if ss2=0
calc dens:=stearray(st, 6)/stearray(st, 8)
temp:=stearray(st, 9)
if stearray(st, 9) < Tc
do rhog(temp; stearray(st, 5))
do rhof(temp; stearray(st, 4))
calc stearray(st, 3):=(1/dens-1/stearray(st, 4))/(1/stearray(st, 5)-1/stearray(st, 4))
if stearray(st, 3) >= 0 & stearray(st, 3) <= 1
do Psat(temp; stearray(st, 7))
else
do P(dens, temp; stearray(st, 7))
endif
else
    calc starray(st,3)=-1
    do     P(dens,temp;starray(st))
endif
at 1085, 280
font zsans, 25
mode rewrite
write
mode write
at 1085, 280
show starray(st,7)
endif
if ss3=0
    do Rho(starray(st,7),starray(st,9);dens)
calc starray(st,8)=starray(st,6)/dens
at 1085, 315
font zsans, 25
mode rewrite
write
mode write
at 1085, 315
show starray(st, 8)
if starray(st,9) < Tc
    do rhog(starray(st,9);starray(st,5))
do rhof(starray(st,9);starray(st,4))
calc starray(st,3):=(1/dens-1/starray(st,4))/{1/starray(st,5)-1/starray(st,4))
else
    calc starray(st,3)=-1
endif
endif
if ss4=0
    calc dens:=starray(st,6)/starray(st,8)
do Tsat(starray(st,7);temp)
if temp < Tc
    do rhog(starray(st,9);starray(st,5))
do rhof(starray(st,9);starray(st,4))
calc starray(st,3):=(1/dens-1/starray(st,4))/{1/starray(st,5)-1/starray(st,4))
if starray(st,3) >= 0 & starray(st,3) =<1
    calc starray(st,9):=temp
else
    do T(dens,starray(st,7);starray(st,9))
endif
else
    calc starray(st,3)=-1
    do T(dens,starray(st,7);starray(st,9))
endif
at 1085, 360
font zsans, 25
mode rewrite
write
mode write
at 1085, 360
show starray(st,9)
endif
if starray(st,3) <=1 & starray(st,3)>=0
mode rewrite
at 1040,70
show stearray(st,3)
calc stearray(st,10):=stearray(st,3)
else
calc stearray(st,10):=-1
endif
if stearray(st, 6)<=0 $or$ stearray(st, 7)<=0 $or$ stearray(st, 8)<=0 $or$ stearray(st, 9)<=0
do cleanl
write You have entered illegal values. Please recheck your numbers.
pause 3
put refr; 1070, 250
do specsteam3(tnum)
endif
do unhigh(stearray(st,1), stearray(st,2))
do sixtwo(stearray(st,1), stearray(st,2))
do clean1
write Data for item is complete. Continue with mesh creation.
calc track(tnum, 5):=1
self:=l
if type=1
do mesh
elseif
dendif

unit satcheck
*$$ Routine to update steam quality at each
loop of the Newton-Raphson iteration
merge, global:
i: q, o
f: vol
loop q:=l, l
calc o:=track(q,4)
if track(q,3)=62 & xvector(mm+1)<Tc
do rhof(xvector(mm+1);stearray(o,4))
do rhog(xvector(mm+1);stearray(o,5))
do ktsteam(q,o;vol)
calc stearray(o,3):=(vol/stearray(o,6)-1/stearray(o,4))/(1/stearray(o,5)-1/stearray(o,4))
endif
endloop

unit ktsteam(tn,dex;vol)
*$$ Convert nodal displacement to
volume change for steam
merge, global:
i: q, dex, tn
f: vol
calc vol:=stearray(dex,8)
loop q:=l, mm
calc
vol:=vol+mechmat(tn,q)*mconnections(mechmat(l+1,q),17)*xvector(q)
endloop

129
unit \texttt{Psat(t;psat)} \quad \textit{Saturation pressure function}

merge, global:
\begin{align*}
\text{calc} & \quad \text{sum} := 0 \\
\text{loop} & \quad \text{i} := 1, 8 \\
\text{calc} & \quad \text{sum} := \text{sum} + F(\text{i})*(a_1*(t-T_p))^{(i-1)} \\
\text{endloop} \\
\text{calc} & \quad \text{psat} := P_c*\exp((T_c/t - 1)*\text{sum})
\end{align*}

unit \texttt{Tsat(\text{pre};tsat)} \quad \textit{Saturation temperature function}

merge, global:
\begin{align*}
\text{calc} & \quad \text{sum} := 0 \\
\text{tsat} := 1.33844e-5*\text{pre}+380.398574 \\
\text{loop} & \quad \text{i} := 1, 20 \\
\text{do} & \quad \text{dPsatdT(tsat;dpdt)} \\
\text{do} & \quad \text{Psat(tsat;psat)} \\
\text{calc} & \quad \text{delta} := (\text{psat}-\text{pre})/\text{dpdt} \\
\text{tsat} := \text{tsat} - \text{delta} \\
\text{if} & \quad \text{abs(delta)} < \text{TOL} \\
\text{outloop} \\
\text{endif} \\
\text{endloop}
\end{align*}

unit \texttt{dPsatdT(t;dpsatdt)} \quad \textit{Calculates dPsat/dT}

merge, global:
\begin{align*}
\text{calc} & \quad \text{sum} := 0 \\
\text{suml} := 0 \\
\text{loop} & \quad \text{i} := 1, 8 \\
\text{calc} & \quad \text{sum} := \text{sum} + F(\text{i})*(a_1*(t-T_p))^{(i-1)} \\
\text{suml} := \text{suml} + (i-1)*F(\text{i})*(a_1*(t-T_p))^{(i-2)} \\
\text{endloop} \\
\text{calc} & \quad \text{dpsatdt} := P_c*\exp((T_c/t - 1)*\text{sum})*(-T_c/t^2*\text{sum}+a_1*(T_c/t-1)*\text{suml})
\end{align*}

unit \texttt{rho(qual,t;rh)} \quad \textit{Density as a function of quality and T}

merge, global:
\begin{align*}
\text{do} & \quad \text{rhog(t;rhg)} \\
\text{do} & \quad \text{rhof(t;rhf)} \\
\text{calc} & \quad \text{rh} := 1/(1/\text{rhf}+\text{qual}*(1/\text{rhg}-1/\text{rhf}))
\end{align*}

unit \texttt{rhof(t;rhf)} \quad \textit{Saturated liquid density function}

merge, global:
\begin{align*}
\text{calc} & \quad \text{sum} := 0 \\
\text{loop} & \quad \text{i} := 1, 8 \\
\text{calc} & \quad \text{sum} := \text{sum} + D(\text{i})*(abs(1-t/T_c))^{(i/3)} \\
\text{endloop} \\
\text{calc} & \quad \text{rhf} := \text{rhoc}*(1+\text{sum})
\end{align*}
unit $rhog(t;rhg)$  \hspace{1cm} $$ Saturated vapor density function$
merge,global:
i:i
f:t,rhg,prs,psat,dpdr,delta
calc rhg:=3E-33*t^12.459
do Psat(t;psat)
loop i:=1,20
do dPdr(rhg,t;dpdr)
do P(rhg,t;prs)
calc delta:=(prs-psat)/dpdr
rhg:=rhg-delta
if abs(delta)<TOL
outloop
eendif
endloop

unit $drhofdT(t;drfdt)$  \hspace{1cm} $$ Calculates temperature derivative of saturated liquid density$
merge,global:
i:i1
f:t,drfdt,sum
calc sum:=0
loop i:=1,8
calc sum:=sum+i*D(i)*(t-Tc)^(i/3-1)
endloop
calc drfdt:=-rhoc*sum/(3*Tc)

unit $drhogdT(t;drgdt)$  \hspace{1cm} $$ Calculates temperature derivative of saturated vapor density$
merge,global:
f:drgdt,t,sl,s2
do rhog(t+TOL;sl)
do rhog(t;s2)
calc drgdt:=(sl-s2)/TOL

unit $uf(t;uf)$  \hspace{1cm} $$ Saturated liquid specific energy function$
merge,global:
f:t,rhf,uf
do rhof(t;rhf)
do u(rhf,t;uf)

unit $ug(t;ug)$  \hspace{1cm} $$ Saturated vapor specific energy function$
merge,global:
f:t,rhg,ug
do rhog(t;rhg)
do u(rhg,t;ug)

unit $qlty(rho,t;qlty)$  \hspace{1cm} $$ Steam quality as a function of density and temperature$
merge,global:
i:os
f:rho,t,qlty,rg,rf
do rhog(t;rg)
do rhof(t;rf)
calc \( q_{\text{lt}} = \frac{1}{\rho} - \frac{1}{\rho_f} \)/(1/\rho_g - 1/\rho_f) \)

\[ u = uf + q_{\text{lt}} \ast (ug - uf) \]

\[ u = uf + q_{\text{lt}} \ast (ug - uf) \]

\[ s = sf + q_{\text{lt}} \ast (sg - sf) \]

\[ s = sf + q_{\text{lt}} \ast (sg - sf) \]

\[ c = c + G(i) \ast t^{i-2} \]

\[ p = \rho \ast R \ast t \ast (l + \rho \ast q + \rho^2 \ast dqdr) \]

\[ t = 460 \]

\[ \text{loop } i = 1,10 \]
\[ \text{do } \text{dPdT}(\rho,t; \text{dpdt}) \]

\[ \text{unit usat}(\rho,t,u) \quad \text{Saturated steam specific energy} \]

\[ \text{unit ssat}(\rho,t,s) \quad \text{Saturated steam specific entropy} \]

\[ \text{unit sf}(t;sf) \quad \text{Saturated liquid specific entropy function} \]

\[ \text{unit sg}(t;sg) \quad \text{Saturated vapor specific entropy function} \]

\[ \text{unit cv}(t;c) \quad \text{Constant volume specific heat for steam} \]

\[ \text{unit P}(\rho,t;\text{pres}) \quad \text{Pressure function for single phase case} \]

\[ \text{unit T}(\rho,\text{prs};t) \quad \text{Temperature function for single phase case} \]
do
  P(rho,t;prss)
calc delta:=(prss-prs)/dpdt
t:=t-delta
if abs(delta)<TOL
  outloop
endif
endloop

unit Rho(prs,t; dens)
* Density function for single phase case
merge,global:
i: i
f: prs, t, dens, prss, rhoi, dpdr, delta, r1, r2, r3, r4
calc dens:=316.957
loop i:=1,10
do dPdr(dens,t;dpdr)
do P(dens,t;prss)
calc delta:=(prss-prs)/dpdr
dens:=dens-delta
if abs(delta)<TOL
  outloop
endif
endloop

unit dPdr(rho,t;dpdr)
* Calculates dP/drho
merge,global:
f: rho, t, dpdr, q, dqdr, dqdr2
do Q(rho,t;q)
do dQdr(rho,t;dqdr)
do dQdr2(rho,t;dqdr2)
calc dpdr:=R*t*(1+2*rho*q+4*rho^2*dqdr+rho^3*dqdr2)

unit dPdT(rho,t;dpdt)
* Calculates dP/dT
merge,global:
f: rho, t, dpdt, dtau, q, dqdt, dqdrdt, dqdr
do Q(rho,t;q)
calc dtau:= -1000/t^2
do dQdt(rho,t;dqdt)
do dQdrdt(rho,t;dqdrdt)
do dQdr(rho,t;dqdr)
calc dpdt:=rho*R*(1+rho*q+rho*t*dqdt*dtau+rho^2*t*dqdrdt*dtau+rho^2*t*dqdr)

unit Q(rho,t;q)
* Calculates Q function
merge, global:
i:i,j
f: rho,t,q,sum1,sum2,sum3
calc sum3:=0
loop j:=1,7
calc sum1:=0
  sum2:=0
loop i:=1,8
calc sum1:=sum1+A(i,j)*(rho-1000-366*(j=1))^(i-1)
endloop
loop i:=9,10
calc sum2:=sum2+exp(-E*rho)*A(i,j)*rho^(i-9)
endloop
calc sum3:=sum3+(1000/t-0.955087859*(j=1)-2.5)^((j-2))*(sum1+sum2)
endloop
calc q:=(1000/t-1.5449121)*sum3

unit dqdr(rho,t,dqdr) \$\$ Calculates dQ/d\rho
merge, global:
i:i,j
f: rho,t,dqdr,sum1,sum2,sum3
calc sum3:=0
loop j:=1,7
calc sum1:=0
  sum2:=0
  loop i:=2,8
    calc suml:=suml+(i-l)*A(i,j)*(rho-1000-366*(j=l))^(i-2)
  endloop
  loop i:=9,10
    calc sum2:=sum2+exp(-E*rho)*A(i,j)*rho^(i-9)
  endloop
  calc sum2:=sum2+exp(-E*rho)*A(10,j)
  calc sum3:=sum3+(1000/t-0.955087859*(j=1)-2.5)^((j-2))*(suml+sum2)
endloop
calc dqdr:=(1000/t-1.5449121)*sum3

unit dqdr2(rho,t,dqdr2) \$\$ Calculates d^2Q/d\rho^2
merge, global:
i:i,j
f: rho,t,dqdr2,sum1,sum2,sum3
calc sum3:=0
loop j:=1,7
calc sum1:=0
  sum2:=0
  loop i:=3,8
    calc suml:=suml+(i-l)*(i-2)*A(i,j)*(rho-1000-366*(j=l))^(i-3)
  endloop
  loop i:=9,10
    calc sum2:=sum2+exp(-E*rho)*A(i,j)*rho^(i-9)
  endloop
  calc sum2:=sum2-2*E*exp(-E*rho)*A(l10,j)
  calc sum3:=sum3+(1000/t-0.955087859*(j=1)-2.5)^((j-2))*(suml+sum2)
endloop
calc dqdr2:=(1000/t-1.5449121)*sum3

unit dqdt(rho,t,dqdt) \$\$ Calculates dQ/d\tau
merge, global:
i:i,j
f: rho,t,dqdt,sum1,sum2
calc dqdt:=0
loop j:=1,7
calc sum1:=0
  sum2:=0
  loop i:=1,8
    calc suml:=suml+A(i,j)*(rho-1000-366*(j=l))
  endloop
  loop i:=9,10
    calc sum2:=sum2+exp(-E*rho)*A(i,j)*rho^(i-9)
  endloop
calc dqdt := dqdt + ((j-2)*(1000/t-1.5449121)*(1000/t-0.955087859*(j=1)-2.5)^(j-3) + (1000/t-0.955087859*(j=1)-2.5)^(j-2))*(sum1+sum2)
endloop

unit dqdt2(rho,t;dqdt2) $$ Calculates d^2Q/d\tau^2
merge, global:
i:i,j
f: rho,t,dqdt2,sum1,sum2
calc dqdt2 := 0
loop j:=1,7
calc sum1 := 0
sum2 := 0
loop i:=1,8
calc sum1 := sum1 + A(i,j)*(rho-1000-366*(j=1))^(i-1)
endloop
loop i:=9,10
calc sum2 := sum2 + exp(-E*rho)*A(i,j)*rho^(i-9)
endloop
calc dqdt2 := dqdt2 + ((j-2)*(j-3)*(1000/t-1.5449121)*(1000/t-0.955087859*(j=1)-2.5)^(j-4)+2*(j-2)*(1000/t-0.955087859*(j=1)-2.5)^(j-3))*(sum1+sum2)
endloop

unit dqdrdt(rho,t;dqdrdt) $$ Calculates d^2Q/(d\rho)(d\tau)
merge, global:
i:i,j
f: rho,t,dqdrdt,sum1,sum2
calc dqdrdt := 0
loop j:=1,7
calc sum1 := 0
sum2 := 0
loop i:=2,8
calc sum1 := sum1 + (i-1)*A(i,j)*(rho-1000-366*(j=1))^(i-2)
endloop
loop i:=9,10
calc sum2 := sum2 - E*exp(-E*rho)*A(i,j)*rho^(i-9)
endloop
calc sum2 := sum2 + exp(-E*rho)*A(10,j)
dqdrdt := dqdrdt + ((j-2)*(1000/t-1.5449121)*(1000/t-0.955087859*(j=1)-2.5)^(j-3)+2*(j-2)*(1000/t-0.955087859*(j=1)-2.5)^(j-2))*(sum1+sum2)
endloop

unit u(rho,t;u) $$ Specific energy function, single phase case
merge, global:
i:i,j
f: rho,t,u,sum,dqdt
calc sum := 0
loop i:=2,6
calc sum := sum + G(i)*t^(i-1)/(i-1)
endloop
do dqdt(rho,t;dqdt)
calc u := G(1)*ln(t)+sum-564413.52+1000*rho*R*dqdt+2375020.7

unit dudr(rho,t;dudr) $$ Calculates du/d\rho
merge, global:
\[ f: \rho, t, d\mu/dt, d\mu/dt^2 \]
\[ dQ_{dt}(\rho, t; dq_{dt}) \]
\[ dQ_{dr/dt}(\rho, t; dq_{dr/dt}) \]
\[ d\mu/dt := 1000R(dq_{dt} + \rho dq_{dr/dt}) \]
\[ \text{unit } d\mu/dT(\rho, t; d\mu/dt) \]

\[ \text{merge, global:} \]
\[ f: \rho, t, d\mu/dt, C_v, dq_{dt2} \]
\[ d\mu/dt := C_v - \rho R dq_{dt2} * (1000/t)^2 \]
\[ \text{unit } s(\rho, t; s) \]
\[ \text{merge, global:} \]
\[ i:i, j \]
\[ f: \rho, t, s, \text{sum}, dq_{dt}, q \]
\[ \text{calc } \text{sum} := 0 \]
\[ \text{loop } i := 3, 6 \]
\[ \text{calc } \text{sum} := \text{sum} + G(i) * t^{(i-2)/(i-2)} \]
\[ \text{endloop} \]
\[ d\mu/dT(\rho, t; d\mu/dt) \]
\[ d\mu/dT := -dP/dT/\rho^2 \]
\[ \text{unit } c_P(\rho, t, os; \text{pres}) \]
\[ \text{merge, global:} \]
\[ i: \text{satfl}, os \]
\[ f: \rho, t, \text{pres} \]
\[ \text{calc } \text{satfl} := \text{abs} (\text{starray(os,3)} - 0.5) \]
\[ \text{case } \text{sign} (\text{satfl}) \]
\[ 0 \]
\[ \text{do } \text{Psat}(t; \text{pres}) \]
\[ 1 \]
\[ \text{do } P(\rho, t; \text{pres}) \]
\[ \text{endcase} \]
\[ \text{unit } c_u(\rho, t, os; u) \]
\[ \text{merge, global:} \]
\[ i: \text{satfl}, os \]
\[ f: \rho, t, u \]
\[ \text{calc } \text{satfl} := \text{abs} (\text{starray(os,3)} - 0.5) \]
\[ \text{case } \text{sign} (\text{satfl}) \]
\[ 0 \]
\[ \text{do } \text{usat}(\rho, t; u) \]
do \( u(\rho,t;u) \)
endcase

\textbf{unit cs(\( \rho, t, os; s \))} \hfill \text{	extdollar\$ Central specific entropy function}
merge,global:
i: satfl, os
f: \( \rho, t, s \)
calc \( satfl:=abs(stearray(os,3)-0.5) \)
case sign(satfl)
0
\hfill do \( ssat(\rho,t;s) \)
1
\hfill do \( s(\rho,t;s) \)
endcase

\textbf{unit cdPdr(\( \rho, t, os; dpdr \))} \hfill \text{	extdollar\$ Central dP/d\( \rho \) function}
merge,global:
i: satfl, os
f: \( \rho, t, dpdr \)
calc \( satfl:=abs(stearray(os,3)-0.5) \)
case sign(satfl)
0
\hfill do \( dpdr:=0 \)
1
\hfill do \( dPdr(\rho,t;dpdr) \)
endcase

\textbf{unit cdPdT(\( \rho, t, os; dpdt \))} \hfill \text{	extdollar\$ Central dP/dT function}
merge,global:
i: satfl, os
f: \( \rho, t, dpdt \)
calc \( satfl:=abs(stearray(os,3)-0.5) \)
case sign(satfl)
0
\hfill do \( dPsatdT(t;dpdt) \)
1
\hfill do \( dPdT(\rho,t;dpdt) \)
endcase

\textbf{unit cdudr(\( \rho, t, os; dudr \))} \hfill \text{	extdollar\$ Central d\( u \)/d\( \rho \) function}
merge,global:
i: satfl, os
f: \( \rho, t, dudr, uf, ug \)
calc \( satfl:=abs(stearray(os,3)-0.5) \)
case sign(satfl)
0
\hfill do \( uf(t;uf) \)
\hfill do \( ug(t;ug) \)
calc \( dudr:=(ug-uf)/(1/stearray(os,5)-1/stearray(os,4))*(-1/\rho^2) \)
1
\hfill do \( dudr(\rho,t;dudr) \)
endcase

\textbf{unit cdudT(\( \rho, t, os; dudt \))} \hfill \text{	extdollar\$ Central d\( u \)/d\( T \) function}
merge,global:
i: satfl, os
f: \( \rho, t, dudt, usat1, usat2 \)
calc satfl := abs(stearray(os,3) - 0.5)
case sign(satfl)
  0
    do usat(rho,t+TOL;usat1)
    do usat(rho,t;usat2)
    calc dudt := (usat1-usat2)/TOL
  1
do dudT(rho,t;dudt)
endcase

unit cdsdr(rho,t,os;dsdr) $$ Central ds/drho function
  merge,global:
  i: satfl, os
  f: rho,t, dsdr, sf, sg
  calc satfl := abs(stearray(os,3) - 0.5)
case sign(satfl)
  0
    do sf(t;sf)
    do sg(t;sg)
    calc dsdr := (sg-sf)/(1/stearray(os,5) - 1/stearray(os,4))*(-1/rho^2)
  1
do dsdr(rho,t;dsdr)
endcase

unit csdT(rho,t,os;dsdt) $$ Central ds/dT function
  merge,global:
  i: satfl, os
  f: rho,t, dsdt, ssatl, ssat
  calc satfl := abs(stearray(os,3) - 0.5)
case sign(satfl)
  0
do ssat(rho,t+TOL;ssatl)
  do ssat(rho,t;ssat2)
  calc dsdt := (ssat1-ssat2)/TOL
  1
do dsdT(rho,t;dsdt)
endcase
Appendix D Bibliography


Science has always been a fascination for Jose Luis Perez. As a child he used to go around the house looking for all chemical substances at his disposal (Clorox, Ajax, Mom’s perfumes, spray roach killer, ketchup, etc.) to mix them up in an attempt to create a new chemical compound or at least some violent chemical reaction. He never received a chemistry set for Christmas, though. He also enjoyed growing bean plants at the apartments’ backyard. It was a wonder for him how from a single bean grain, a plant could grow and provide more beans. He tried the same concept with a 10¢ dime, but to his disappointment, he realized it did not work for coins.

As a teenager he developed an interest for computers and programming. While in high school he spent more time than he should have, at home programming all sorts of computer games on his Commodore 64K. In 1987, after applying for college he was admitted to the Computer Engineering Department at the University of Puerto Rico, Mayaguéz Campus. After a couple of years he realized that the Computer Engineering program at Mayaguéz was still in its initial stages, and was no yet accredited by the Accreditation Board of Engineering and Technology (ABET). Not wanting to compromise his professional career, he switched to the Mechanical Engineering Department in 1989. Mechanical engineering offered the attractive of being a well-rounded discipline. Its curriculum included courses required by the Electrical, Computer, Civil, and Industrial Engineering Departments. During his four and a half years (out of a normal five) in Mayaguéz, he received various honors and awards, including membership in the Tau Beta Pi and Phi Kappa Phi Honor Societies. He was also recipient of the Governor’s Scholarship and the NACME Fellowship, which provided subsistence during those tough student years. He finally graduated Magna Cum Laude from Mayaguéz on December, 1991.

In 1993 he joined the Mechanical Engineering Department at MIT.