University of Illinois at Urbana-Champaign

ACRC

Air Conditioning and Refrigeration Center A National Science Foundation/University Cooperative Research Center

Room Air Conditioner System Modeling

C. E. Mullen and C. W. Bullard

ACRC TR-60 July 1994

For additional information:

Air Conditioning and Refrigeration Center
University of Illinois
Mechanical & Industrial Engineering Dept.
1206 West Green Street
Urbana, IL 61801

(217) 333-3115

Prepared as part of ACRC Project 41
Alternative Refrigerants in Room Air Conditioners
C. W. Bullard and W. E. Dunn, Principal Investigators

The Air Conditioning and Refrigeration Center was founded in 1988 with a grant from the estate of Richard W. Kritzer, the founder of Peerless of America Inc. A State of Illinois Technology Challenge Grant helped build the laboratory facilities. The ACRC receives continuing support from the Richard W. Kritzer Endowment and the National Science Foundation. The following organizations have also become sponsors of the Center.

Acustar Division of Chrysler
Allied Signal, Inc.
Amana Refrigeration, Inc.
Brazeway, Inc.
Carrier Corporation
Caterpillar, Inc.
E. I. duPont Nemours & Co.
Electric Power Research Institute
Ford Motor Company
Frigidaire Company
General Electric Company
Harrison Division of GM
ICI Americas, Inc.
Modine Manufacturing Company
Peerless of America, Inc.
Environmental Protection Agency
U.S. Army CERL
Whirlpool Corporation

For additional information:

Air Conditioning & Refrigeration Center
Mechanical & Industrial Engineering Dept.
University of Illinois
1206 West Green Street
Urbana, IL 61801

217 333 3115

Abstract

Increasingly stringent energy standards and the redesigning of room air conditioners for use with alternative refrigerants have highlighted the need for design and simulation models that are accurate, easy to modify, and flexible enough for a variety of design and simulation tasks. This report describes the latest version of the Air Conditioning and Refrigeration Center (ACRC) room air conditioner simulation model. The model is being continually improved using heat transfer and pressure drop correlations and other modifications which are added as a result of an ongoing experimental program.

The governing equations are solved using the Newton-Raphson method, which allows the equations to be listed in an order-independent fashion. A specialized Newton-Raphson equation solving package was developed that allows model variables and parameters in the governing equations to be easily "swapped" for design purposes and includes automated uncertainty and sensitivity analyses.

A two-stage approach toward room air conditioner instrumentation is an integral part of the plans for model validation and improvement. The first stage uses only air-side thermocouples, and the second stage introduces refrigerant-side thermocouples and pressure transducers. Comparison of data sets from the two stages allows the intrusive effects of the refrigerant-side instrumentation to be identified. Simultaneous measurements with air and refrigerant-side thermocouples make it possible to quantify the offset errors in surface thermocouple measurements of refrigerant temperatures.

The first stage of the room air conditioner instrumentation has been completed, and a baseline data set has been taken. The predictions of each component model in the room air conditioner system model are compared with preliminary experimental results obtained from this non-intrusive instrumentation, and component models which require improvement are identified. Several appendices address details of the model and equation solver, modeling condenser condensate spray, and the uncertainty analysis methods and results.

Table of Contents

List of Figures

Page

List of Tables

Chapter 1: Introduction

The Air Conditioner and Refrigeration Center (ACRC) room air conditioner model (RACMOD) and an equation solving package with special features for thermal systems modeling have been developed in response to the need for a flexible design and simulation tool for room air conditioners.

Hahn and Bullard (1993) originally developed the ACRC room air conditioner model based on governing equations embedded in the Oak Ridge National Laboratory (ORNL) heat pump model (Fischer and Rice, 1983) and modifications to those equations for room air conditioners (O'Neal and Penson, 1988). Approximately 100 governing equations were used, many of which were taken directly from the ORNL model.

Hahn and Bullard solved the equations with the Newton Raphson method rather than using the ORNL model's successive substitution algorithm. The Newton Raphson (NR) method allows the governing equations to be listed in any order (separately from the solution logic,) so that they are easier to understand and modify. The NR method also makes it possible to "swap" parameters and variables in the governing equations without changing the solution logic.

Improved refrigerant-side two-phase heat transfer correlations by Dobson *et al*. (1994) and Wattelet *et al*.(1994) were implemented by Hahn and Bullard. A refrigerant-specific curve fit of a finite-difference capillary tube developed by Peixoto and Bullard (1994) was also implemented as an alternative to the curve fit of the ASHRAE capillary tube model which was used by O'Neal and Penson.

Further development of RACMOD has been carried out by the authors. Charge inventory equations using the Hughmark (1962) correlation for void fraction were implemented, and ORNL's two-phase refrigerant pressure drop correlations were replaced with correlations by Souza *et al*.(1992). A technique was developed that allows the approximately 200 equations of Peixoto's finite difference capillary tube model to be incorporated directly into RACMOD using only four governing equations. Equations to model air recirculation at the evaporator and condenser and to account for condenser heat transfer enhancement due to condensate spray were added. Equations and equation-switching logic were also developed to handle two-phase exit conditions in the evaporator and condenser.

A Fortran equation solving and analysis program, the "ACRC solver," with special features for thermal systems, was also developed. The ACRC solver allows parameters and variables in the governing equations to be "swapped" without recompiling–making it simple to change a model from a simulation mode to a variety of design modes or to solve for an estimated model parameter given known output values. Automated uncertainty and sensitivity analyses are also included in the ACRC solver, as well as features that enhance the Newton Raphson solution method's speed and robustness.

References

Dobson, M. K. "Heat Transfer and Flow Regimes during Condensation in Horizontal Tubes." University of Illinois at Urbana-Champaign, ACRC TR-57, 1994.

Fischer, S. K. and C. K. Rice. The Oak Ridge Heat Pump Models. Oak Ridge National Laboratory, 1983. ORNL/CON-80/R1.

Hahn, Gregory W. “Modeling Room Air Conditioner Performance.” University of Illinois at Urbana-Champaign, ACRC TR-40, 1993.

Hughmark, G.A. “Hold-up in Gas-liquid Flow.” *Chemical Engineering Progress* 4 (1962).

O'Neal, D. L. and S. B. Penson. An Analysis of Efficiency Improvements In Room Air Conditioners. Texas A&M University, 1988. ESL/88-04.

Peixoto, R. A. and C. W. Bullard. "A Design Model for Capillary Tube-Suction Line Heat Exchangers." University of Illinois at Urbana-Champaign, ACRC TR-53, 1994.

Wattelet, J. P. "Heat Transfer Flow Regimes of Refrigerants in a Horizontal-Tube Evaporator." University of Illinois at Urbana-Champaign, ACRC TR-55, 1994.

Chapter 2: Model Description

The RACMOD system model consists of component models for the condenser, evaporator, compressor, and capillary tube, as well as equations for charge calculation. The governing equations for all of the component models are listed together and solved as one simultaneous set. This chapter describes the modeling strategies and correlations used for each of the component models.

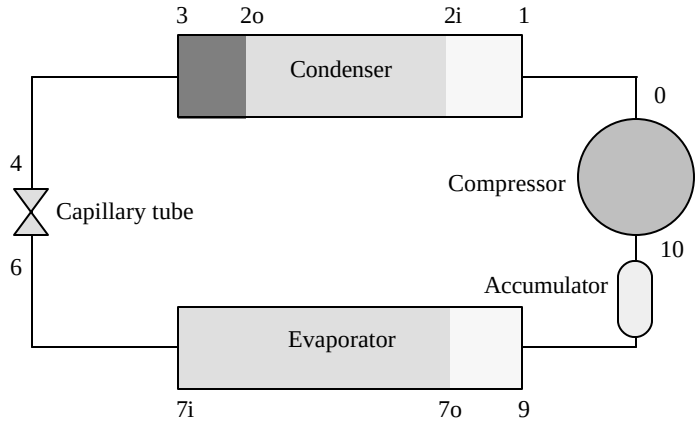

Figure 2.1 Schematic of room air conditioner model showing state points

2.1 Condenser

The room air conditioner condenser is modeled as a crossflow heat exchanger with uniform inlet air temperature and velocity. Three heat transfer zones–desuperheating, two-phase, and subcooling–are modeled using effectiveness-NTU heat transfer rate equations. The two-phase zone is modeled as isothermal along the flow direction, though it could be expanded using a finite-difference approach to handle refrigerants with a temperature glide. The correlations used for heat transfer correlations and pressure drop calculations are referenced in Table 2.1. As in the ORNL model (Fischer and Rice, 1983), condenser geometries with more than one refrigerant circuit are modeled as identical, parallel condensers.

The condenser heat transfer equations were modified to account for heat transfer enhancement due to condensate from the evaporator that is sprayed onto the condenser by the condenser fan's "sling ring." The enhancement is based on an effective mass flow rate of water multiplied by water's heat of vaporization. Details are given in Section 5.5.3 and in Appendix I.

Table 2.1 Correlations used by RACMOD and the ORNL heat pump model

	ORNL heat pump	RACMOD
Condenser heat transfer		
Subcooled	Dittus and Boelter (1930)	Same
Superheated	Kays and London (1974)	Same
Two-phase	Hiller (1976), Travis (1973), and Rohsenow (1973)	Dobson (1994)
Air-side	McQuiston (1978, 1981), Yoshi (1972), and Senshu (1979)	Same
Evaporator heat transfer		
Superheated	Dittus and Boelter (1930)	Same
Two-phase	Chaddock (1966) and Sthapak (1976)	Wattelet (1994)
Air-side	McQuiston (1978, 1981), Yoshi (1972), and Senshu (1979)	Same
Dehumidification analysis	Fischer(1983)	Same
Pressure Drop		
Two-phase condenser ΔP	Thom (1964)	Souza (1992)
Two-phase evaporator ΔP	Goldstein (1979)	Souza (1992)
Single phase ΔP	Moody friction factor	Same
Single phase return bend ΔP	Ito (1960)	Same
Two-phase return bend ΔP	Geary (1975)	Christoffersen (1993)

2.2 Evaporator

Like the condenser, the room air conditioner evaporator model assumes a crossflow heat exchanger with uniform inlet air temperature and velocity. Two heat transfer zones–two-phase and superheating–are modeled using effectiveness-NTU heat transfer rate equations. The correlations used for heat transfer correlations and pressure drop calculations are referenced in Table 2.1. Evaporator geometries with more than one refrigerant circuit are modeled as parallel evaporators.

The evaporator heat transfer and dehumidification analysis is performed by a subroutine taken directly from the ORNL heat pump model, although its solution logic has been modified for implementation with the Newton-Raphson method.

2.3 Compressor

A manufacturer-supplied compressor map is used to predict the compressor mass flow and power consumption as a function of condensing and evaporating temperatures. The ORNL model's correction of mass flow and power predictions for non-standard suction gas return temperatures was not found to improve the map's mass flow and power consumption predictions and was not used.

2.4 Capillary Tube

The ACRC finite-difference adiabatic capillary tube model developed by Peixoto and Bullard (1994) has been implemented in RACMOD. The ACRC capillary tube model assumes choked flow at the exit, and it uses a Blasius

friction factor for the subcooled pressure drop calculations and a correlation by Pate (1982) for the two-phase friction factor.

The ACRC finite difference adiabatic capillary tube model consists of over 200 equations that normally must be solved simultaneously. Adding 200 simultaneous equations to RACMOD, however, would significantly slow its execution time, and the 200^+ variables in the captube model would all require initial guesses, to which the captube model is very sensitive. This sensitivity to initial guesses would cause difficulty in converging on a solution for the captube model. These factors make incorporating the capillary tube equations directly into RACMOD impractical.

Normally, one thinks of the inlet subcooling and the length of the capillary tube as known inputs to the model, with the mass flow and other variables being outputs. However, upon examination of the captube equations for a case with a subcooled capillary tube entrance, it was found that when the inlet pressure (p4), captube diameter (Dcap), pressure and quality in the outlet region (Pcritcap, xcritcap), and a pressure step internal to the model (DeltaP) are considered "known," then the captube equations may be rewritten so that they are sequential* rather than simultaneous. These sequential equations can be solved explicitly for inlet subcooling, length of the captube, mass flow, and the quality at the refrigerant flash point. Thus the 200^+ equation capillary tube model can be expressed mathematically as four explicit functions:

$$\text{(inlet subcooling)} = f_d(\text{p4, Dcap, Pcritcap, DeltaP, and xcritcap}) \quad (2.1)$$

$$\text{(length of the captube)} = f_L(\text{p4, Dcap, Pcritcap, DeltaP, and xcritcap}) \quad (2.2)$$

$$\text{(mass flow)} = f_w(\text{p4, Dcap, Pcritcap, DeltaP, and xcritcap}) \quad (2.3)$$

$$\text{(quality at flash point)} = f_x(\text{p4, Dcap, Pcritcap, DeltaP, and xcritcap}) \quad (2.4)$$

These functions may be thought of as the sequential version of the capillary tube equations "backsubstituted" into four explicit functions. Because these four functions share so many calculations, they are implemented as one subroutine with four outputs: degsubcoolCalc, LcapCalc, wcapCalc, and XflashCalc. These outputs are used in the following governing equations:

$$\text{degsubcool} = \text{degsubcoolCalc} \quad (2.5)$$

$$\text{Lcap} = \text{LcapCalc} \quad (2.6)$$

$$\text{w} = \text{wcapCalc} \quad (2.7)$$

$$0 = \text{XflashCalc} \quad (2.8)$$

Equations 2.5-2.7 specify that the capillary tube model's calculations must agree with the appropriate NR variables and parameters, and Equation 2.8 specifies that the calculated flash point in the captube must indeed occur where the refrigerant quality is zero.

In this way, the 200^+ equations of the finite-difference capillary tube model are implemented as four NR equations, and the NR solver adjusts the values of the variables Pcritcap, DeltaP, xcritcap, and w, along with all of the other variables in RACMOD until all equations are satisfied. This "backsubstitution" of the capillary tube model into four NR equations dramatically improved the convergence of the model, because most internal variables are eliminated, reducing the possibility of inconsistent initial guesses.

---

* Although the equations are sequential, some of them are implicit in one variable and must be solved iteratively.

The ASHRAE curve-fit model (ASHRAE, 1988) was used by O'Neal and Penson (1988) and may also be selected when using RACMOD.

2.5 Refrigerant Charge Calculation

Simple volume and density calculations are used to calculate the refrigerant charge in the room air conditioner. The Hughmark correlation for void fraction (Hughmark, 1962) was chosen for calculating the charge in the two-phase regions of the condenser and evaporator because it was found to be most accurate in work by Rice and by Goodson (Goodson, 1994; Rice, 1987).

References

ASHRAE, 1988 Equipment handbook.

Chaddock J. B. and J. A. Noerager. "Evaporation of R12 in a Horizontal Tube with a Constant Wall Heat Flux." ASHRAE Transactions Vol. 72, Part 1, 1966.

Christoffersen, Brian, J. C. Chato, J.P. Wattelet, and A.L. de Souza. "Heat Transfer and Flow Characteristics of R-22, R-32/R-125 and R-134a in Smooth and Micro-fin Tubes." University of Illinois at Urbana-Champaign, ACRC TR-47, 1993.

Dittus, F. W. and L. M. K. Boelter. University of California (Berkeley) Publications on Engineering, Vol. 2, p. 443, 1930.

Dobson, M. K. "Heat Transfer and Flow Regimes During Condensation in Horizontal Tubes." University of Illinois at Urbana-Champaign, ACRC TR-57, 1994.

Fischer, S. K. and C. K. Rice. The Oak Ridge Heat Pump Models. Oak Ridge National Laboratory, 1983. ORNL/CON-80/R1.

Geary, D. F., "Return Bend Pressure Drop in Refrigeration Systems," *ASHRAE Transactions*., Vol. 81, Part 1, 1975.

Goldstein, S. D. "On the Calculation of R-22 Pressure Drop in HVAC Evaporators." ASHRAE Transactions., Vol. 85, Part 2, 1979.

Goodson, M. "Modeling of a Refrigerator/Freezer System." University of Illinois and Urbana-Champaign, ACRC TR-61, 1994.

Hahn, Gregory W. and C. W. Bullard "Modeling Room Air Conditioner Performance." University of Illinois at Urbana-Champaign, ACRC TR-40, 1993.

Hiller, C. C. and L.R. Glicksman, Improving Heat Pump Performance via Compressor Capacity Control – Analysis and Test, Vols. I and II, MIT Energy Laboratory Report No. MIT-EL 76-001, 1976.

Hughmark, G.A. "Hold-up in Gas-liquid Flow." Chemical Engineering Progress, 58 (4 1962).

Ito, H. "Pressure Losses in Smooth Pipe Bends." Journal of Basic Engineering: Transactions of the ASME. March 1960, p.135.

Kays And London, Compact Heat Exchangers, 1964, P. 182, Fig. 10.1, Surf. St-1.

McQuiston, F.C. "Correlation of Heat, Mass, and Momentum Transport Coefficients for Plate-Fin-Tube Heat Transfer Surfaces with Staggered Tubes." ASHRAE Transactions., Vol. 84, Part 1, 1978.

McQuiston, F.C. "Finned Tube Heat Exchangers: State of the Art for the Air Side." ASHRAE Transactions., Vol. 87, Part 1, 1981.

O'Neal, D. L. and S. B. Penson. An Analysis of Efficiency Improvements In Room Air Conditioners. Texas A&M University, 1988. ESL/88-04.

Pate, M. B. "A theoretical and experimental analysis of capillary tube-suction line heat exchangers." Ph. D., Purdue University, 1982.

Peixoto, R. and C. W. Bullard. "A Design Model for Capillary Tube-Suction Line Heat Exchangers." University of Illinois at Urbana-Champaign, 1993. ACRC TR-53, 1994.

Rohsenow, W. M. and J. P. Hartnett, eds., Handbook of Heat Transfer. McGraw-Hill, p. 12-29, 1973.

Sthapak, B. K., *et. al*. "Heat Transfer Coefficient in Dry-out Region of Horizontal Tube." ASHRAE Transactions Vol. 82, Part 2, 1976.

Travis, D.P, A. B. Baron, and W. M. Rohsenow, "Forced Convection Condensation Inside Tubes: A Heat Transfer Equation for Condenser Design." ASHRAE Transactions Vol. 79, Part 1, 1973.

Rice, C.K. “The Effect of Void Fraction Correlation and Heat Flux Assumption on Refrigerant Charge Inventory Predictions.” ASHRAE Transactions 93, Part 1 (1987): 341-367.

Senshu, T. *et. al*. "Surface Heat Transfer Coefficient of Fins Utilized in Air-Cooled Heat Exchangers," Reito. 54:615, 1979, pp. 11-17.

Souza, A. L., J. C. Chato, J. M. S. Jabardo, J. P. Wattelet, J. Panek, B. Christoffersen, and N. Rhines. Pressure Drop During Two-Phase Flow of Refrigerants in Horizontal Smooth Tubes. University of Illinois at Urbana-Champaign, ACRC TR-25, 1992.

Thom, J. R. "Prediction of Pressure Drop During Forced Circulation Boiling of Water." International Journal of Heat and Mass Transfer. Vol. 7, pp. 709-724, 1964.

Wattelet, J. P. "Heat Transfer Flow Regimes of Refrigerants in a Horizontal -Tube Evaporator." University of Illinois at Urbana-Champaign, ACRC TR-55, 1994.

Yoshii, T. "Transient testing technique for heat exchanger fin." *Reito*, 47:531, 1972, pp. 23-29.

Chapter 3: RACMOD and the ACRC Solver

The ACRC solver handles most input and output for RACMOD and solves its governing equations with a modified Newton Raphson method. The ACRC solver also performs ASME and Monte Carlo uncertainty analyses and a simple sensitivity analysis of the governing equations.

The organizational framework of the ACRC solver is based on the TrueBasic sensitivity analysis program by Porter and Bullard (1992). The ACRC solver includes modifications to the Newton Raphson algorithm that improve its ability to solve equations when poor guesses for variable values are given. A simple means of "swapping" variables and parameters in the governing equations is also implemented. Sparse-matrix Jacobian calculation was implemented by Hahn and Bullard (1993) and has been slightly modified in the present version.

The ACRC solver and RACMOD are described in detail in the appendices.

3.1 Model-Solver Relationship

The structure and organization of the ACRC room air conditioner model as implemented with the ACRC solver is depicted in Figure 3.1. The separate subroutines for model initialization, checking, and equation evaluation allow this structure to handle special problems that arise in thermal system simulations. For instance, the boundary checking in RACMOD determines whether the refrigerant at the evaporator exit is currently two-phase or superheated and switches to a slightly modified equation set if the condition has changed since the last iteration. Because the equations are listed separately and in an order-independent fashion, it is relatively easy to modify them or remove a component model and replace it with a new one.

3.2 Swapping Parameters and Variables

The basic requirement of the Newton-Raphson method is that there as many governing equations as variables and that the equations be independent and non-singular. Thus a given variable can become a parameter if a former parameter simultaneously becomes a variable (in order to maintain the same number of equations and variables), as long as the equations remain independent and no singularities.

For example, Figure 3.2 depicts a set of three equations, requiring three variables for solution. Conventionally, a designer might specify the evaporator area (A_{evap}) and solve for the COP, but "swapping" allows the NR method to solve for the A_{evap} that will yield a particular COP. Examples of using "swapping" with RACMOD include specifying capacity while solving for evaporator size or the length or diameter of the capillary tube. Total system refrigerant charge can be specified, solving for condenser subcooling, or vice-versa.

Variable-parameter swapping can also be used for parameter estimation. For example, the outlet conditions of a heat exchanger may be specified to allow a heat transfer coefficient to be solved for.

The ACRC solver allows swapping of a parameter and a variable by simply changing two flags in the input file. There is no need to change the program or recompile, making it simple to change the model from a simulation to a variety of design configurations.

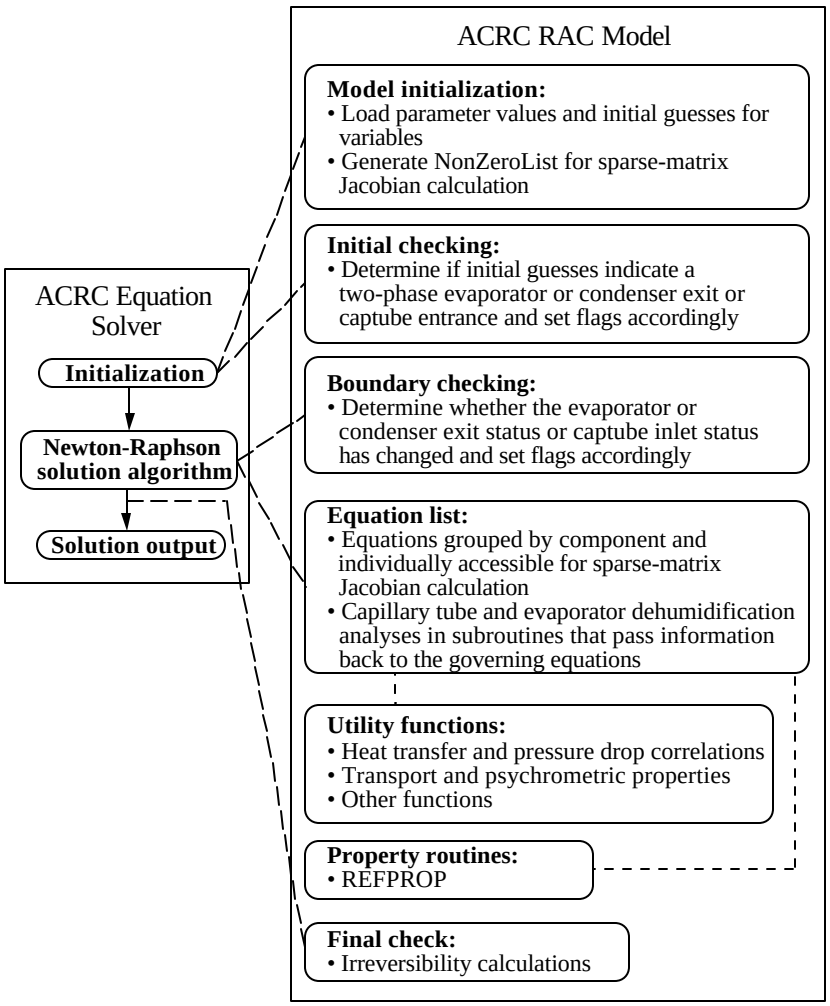

Figure 3.1 Organization of RACMOD and the ACRC solver

Equation set:	Normal configuration:		After swapping:	
	Variables:	Parameters:	Variables:	Parameters:
$COP = \frac{Q_{evap}}{\dot{W}_{comp}}$	COP	U	A_{evap}	U
$Q_{evap} = U \cdot A_{evap} \Delta T$	Q_{evap}	A_{evap}	Q_{evap}	COP
$\dot{W}_{comp} = f(\Delta T)$	$\dot{W}_{comp}$	ΔT	$\dot{W}_{comp}$	ΔT

Figure 3.2 Example of parameter-variable "swapping"

3.3 Speed Enhancements in the Model and Solver

The room air conditioner model consists of about 160 governing equations, many of which involve lengthy calls to property routines or pressure drop functions. A straightforward evaluation of the Jacobian matrix for the 160 equations would require 160^2 partial derivative calculations and involve considerable execution time. However, most of those equations contain only a few variables, so the majority of the partial derivatives are always zero. Such a system of equations is termed "sparse."

To improve execution time, the non-zero elements of the Jacobian are mapped in advance, and that information is used to ensure that only those partial derivatives that may be non-zero are evaluated when the Jacobian is calculated. The remainder of the Jacobian elements are always zero and no time is wasted by calculating them. Sparse-matrix Jacobian calculation is described in greater detail in Section B.4.

Similarly, a sparse-matrix Gaussian elimination routine given by Stoecker uses full pivoting and linked lists to speed that step of the Newton-Raphson solution (Stoecker, 1989). The results of the speed enhancements are presented in Table 3.1. Actual execution times vary by computer, but it is clearly demonstrated that while the sparse-matrix Gaussian elimination saves a significant amount of time per NR iteration, the largest enhancement is obtained through the sparse-matrix Jacobian calculation. The benefit of using sparse-matrix Jacobian calculation is more dramatic when using the ACRC finite-difference capillary tube model because it involves the execution of a lengthy subroutine, while the ASHRAE curve-fit capillary tube model requires little computation.

Table 3.1 Speed enhancement results

	ACRC captube	ASHRAE captube
Sec/iteration with no enhancement:	100	58
Adding sparse Gaussian elimination:	92	50
Adding sparse Jacobian calculation:	12	9

3.4 Automated Step Relaxation to Enhance Solution Robustness

The Newton-Raphson method is not globally convergent–a NR step may be calculated that does not bring the variables closer to a solution, particularly when the initial guesses are poor. A NR step may even result in an attempt to evaluate a function (e.g. a thermodynamic or transport property) outside of its domain. Common examples include attempting to calculate a refrigerant quality above the critical temperature or attempting to raise a negative number to a non-integer power (e.g. in a heat transfer or pressure drop correlation).

When either of the above instances occurs, the ACRC Newton-Raphson implementation recognizes it, retraces the step, reduces the NR step size by half, and retakes the shorter step. This technique greatly increases the model's robustness and somewhat reduces the need for good initial guesses. Details of the automatic step relaxation are given in Section A.4.3

References

Hahn, Gregory W. and C. W. Bullard “Modeling Room Air Conditioner Performance.” University of Illinois at Urbana-Champaign, ACRC TR-40, 1993.

Porter, K. J. and C. W. Bullard. Modeling and Sensitivity Analysis of a Refrigerator/Freezer System. University of Illinois at Urbana-Champaign, ACRC TR-31, 1992.

Stoecker, W. F. Design of Thermal Systems. 3rd ed., New York: McGraw-Hill, 1989.

Chapter 4: Status of Model Development

4.1 Introduction

As outlined in Chapter 1, RACMOD was based on governing equations from the ORNL heat pump model (Fischer and Rice, 1980) as modified by O'Neal and Penson (1983) for room air conditioners. The modified ORNL model, which has been used by the DoE for regulatory purposes, is not a full simulation model because it does not contain charge inventory equations, but instead relies on a specification of evaporator superheat. O'Neal and Penson validated the model against three experimental data points that were taken at the rating condition of 80°F indoor temperature, 95°F outdoor temperature, with no moisture removal by the evaporator. For the three points, they found that their model's EER and capacity predictions agreed with experimental measurements within 5% when the amount of evaporator superheat was specified. No validation at off-design conditions were reported for the model.

RACMOD models the condenser and evaporator in the same manner as the modified ORNL model, although updated refrigerant-side heat transfer and pressure drop correlations have been used. Like the ORNL model, RACMOD models the compressor with an empirical compressor map and contains the ASHRAE capillary tube model, as well as a first-principles capillary tube model by Peixoto and Bullard (1994). RACMOD may be run with a specified amount of superheat or subcooling, like the ORNL model as modified by O'Neal and Penson. The addition of charge inventory to the model also allows RACMOD to be run in a true simulation mode by specifying the total refrigerant charge in the system rather than specifying evaporator superheat or condenser subcooling.

This ability to run in a true simulation mode can be very valuable for the prediction of performance for geometries or operating conditions for which experimental estimates of condenser subcooling or evaporator superheat are not available. The lack of some experimentally determined data makes accurate prediction more difficult for a true simulation model, because errors in one component are always propagated to the other components. In a design model, specified superheat and/or subcooling provides a point of reference that removes one degree of freedom in the model and reduces error propagation.

A complete validation of a room air conditioner simulation model should include predictions at various operating conditions in order to test the limits of the model's accuracy. The various component models of RACMOD must all be shown to be valid on an individual basis before accuracy in a full simulation system model is to be expected. Component model validations will require accurate and detailed measurements of refrigerant and air side conditions at various locations in the refrigerant loop, as well as accurate mass flow measurements.

4.2 Instrumentation Schedule

Each room air conditioner tested by the ACRC room air conditioner project will be instrumented in stages in order to obtain the maximum amount of information about the units and the effects of installing refrigerant-side instrumentation in them.

The first stage is instrumentation with air-side thermocouples. This non-intrusive instrumentation estimates refrigerant-side temperatures using tube surface measurements, and refrigerant pressures can only be determined in two-phase regions, using saturation temperature and pressure relations. Though the air-side instrumentation does not provide the best absolute measurements, it provides a reference for comparison of system performance during

the second stage of instrumentation, which includes refrigerant-side measurements. Comparison of data on the same room air conditioner before and after the introduction of refrigerant-side instrumentation will allow the intrusive effects of such instrumentation to be quantified. During the second stage of instrumentation, analysis of temperatures measured simultaneously with refrigerant-side and surface thermocouples will help to quantify the offset errors associated with surface measurements, so that corrections to future air-side measurements can be estimated. Once RACMOD has been fully validated, the corrected air-side measurements may be adequate to check model predictions on new test units, and they may be used to evaluate the effects of model improvements as well.

The second stage of instrumentation involves the addition of refrigerant-side thermocouples and pressure transducers which will allow much more accurate determination of refrigerant-side state points. In the second stage, the pressure drop correlations and penalty factors for enhanced tubes may be evaluated directly by the pressure transducers, reducing the potential sources of error in the model. More accurate determination of refrigerant mass flows and the states at the inlet and outlet of components will allow precise component-level validations, so that problems with the first-principles models can be pinpointed and the governing equations can be improved.

4.3 Current Status

At present, the first stage of instrumentation has been completed, and an initial baseline data set has been taken. The baseline set contains data for 54 operating conditions covering a full range of indoor and outdoor room temperatures and humidities and three fan speeds. These data were obtained using a room air conditioner environmental chamber described in detail by coworkers (Feller and Dunn, 1993; Fleming and Dunn, 1993; Rugg and Dunn, 1994). The manufacturer's measurements of volumetric air flow rates were used, and a refrigerant-side energy balance was used with calorimetry measurement of the evaporator capacity to obtain refrigerant mass flow rates. Two-phase pressures were obtained through surface thermocouple temperature measurements and saturation pressure-temperature relations, and single-phase pressures were estimated using pressure drop correlations.

The experimental data currently available are adequate for checking the accuracy of component model predictions, and the results of comparisons between experimental and modeled results will guide efforts at improving first-principles modeling equations and pressure drop and heat transfer correlations in the component models. The data available in the first stage of instrumentation, however, are not accurate enough to be used in a final model validation or to thoroughly evaluate specific model improvements.

Chapter 5 presents comparisons of modeled and predicted results for the various component models and for the full system simulation model and provides greater detail on certain aspects of the heat exchanger modeling.

References

Feller, Scott and W. E. Dunn. “Design of the Outdoor Environmental Chamber of a Room Air Conditioner Test Facility.” University of Illinois at Urbana-Champaign, ACRC TR-43, 1992.

Fischer, S. K. and C. K. Rice. The Oak Ridge Heat Pump Models. Oak Ridge National Laboratory, 1983. ORNL/CON-80/R1.

Fleming, Jonathan and W. E. Dunn. “Design of the Psycrometric Calorimeter Chamber of a Room Air Conditioner Test Facility.” University of Illinois at Urbana-Champaign, ACRC TR-44, 1993.

O'Neal, D. L. and S. B. Penson. An Analysis of Efficiency Improvements In Room Air Conditioners. Texas A&M University, 1988. ESL/88-04.

Peixoto, R. and C. W. Bullard. "A Design Model for Capillary Tube-Suction Line Heat Exchangers." University of Illinois at Urbana-Champaign, 1993. ACRC TR-53, 1994.

Rugg, Steve and W. E. Dunn. "Design, Testing, and Validation of a Room Air Conditioner Test Facility" University of Illinois at Urbana-Champaign, ACRC TR-59, 1994.

Chapter 5: Preliminary Comparisons of Model Predictions

The comparisons presented here are based on the first stage of instrumentation of a room air conditioner and are thus preliminary. The primary purpose of the baseline data on which these comparisons are based is to provide a reference against which system performance can be compared after intrusive refrigerant-side instrumentation has been added. The comparisons presented here are adequate, though, to point out deficiencies in some of the first-principles component models, which will guide future efforts at model improvement. Improvements to the governing equations of these models can be made based data taken using the second stage of instrumentation.

For each component model, the inlet refrigerant and air conditions (including mass flows) were determined experimentally, and these values were specified in the component models. Predictions of outlet conditions were then compared with experimental data. Uncertainty analyses were used to determine the accuracy of the measured mass flows and to evaluate the expected scatter in component model predictions due to uncertainties in the measured inlet conditions.

Without validated component models, accuracy in the full system model is not to be expected, but the results of system simulation predictions are provided as a basis of comparison with predictions based on future model improvements.

5.1 Refrigerant Mass Flow Measurement

An accurate measurement of refrigerant mass flow rate is critical to all of the component model validations. Because the mass flow meter had not yet been installed, a refrigerant-side energy balance was used to determine refrigerant mass flow.

Temperatures at the inlet of the adiabatic capillary tube and at the evaporator exit were measured with surface thermocouples and pressures were determined using surface thermocouples in the two-phase region of the condenser and evaporator along with pressure drop calculations. These temperatures and pressures specify the enthalpies at the entrance and exit of the evaporator (h2i and h2o, respectively) for the 39 data points where the evaporator exit state is not two-phase. The measured evaporator capacity (q_{evap}) can then be used in the following equation to determine w_{eb}, the energy balance-based mass flow:

$$q_{evap} = w_{eb} * (h2o - h2i). \tag{5.1}$$

A Monte-Carlo uncertainty analysis was performed on the calculation of w_{eb}, the details of which are given in Appendix H. The uncertainty analysis indicates that w_{eb} should be accurate to within 2.8%. All of the component validations utilize w_{eb} and thus its uncertainty will limit the ability to discern component accuracies.

5.2 Volumetric Air Flow Rate Measurement

The volumetric flow rate of air is critical to evaporator and condenser performance. Manufacturer-supplied data on volumetric flow rates was obtained (Christoffersen, 1994a), although the uncertainty in those measurements is not known. As a check on the manufacturer's data, the volumetric flow rate of the evaporator was determined with an air side energy balance:

$$q_{evap} - q_{fan} = \dot{m}_{air} (T_{a,evapin} - T_{a,evapfanout}) \tag{5.2}$$

The equations needed to determine q_{evap} and q_{fan} from measurements, the estimation of the radiation error correction to $T_{a,\,evapin}$, and possible measurement error in $T_{a,\,evapfanout}$ are all potential sources of error in the estimate. $T_{a,\,evapin}$ is the average temperature of a grid of eight thermocouples mounted on wires directly in front of the evaporator, and Porter (1994) estimates a radiation error of -1.4°F due to the close proximity to the evaporator. $T_{a,\,evapfanout}$ is the average of three thermocouples at the outlet grill of the air conditioner. Because it is downstream of the evaporator, inadequate mixing of the colder air from the two-phase zone of the evaporator and warmer air from the superheated zone may introduce error in $T_{a,\,evapfanout}$. For this reason, only data points where the entire evaporator was two-phase were used for determination of the air flow rate. Data points where superheat occurred in the evaporator produced unrealistic scatter in the calculated air flow rate.

Flow rates were determined for three fan speeds, and an uncertainty analysis on the air-side energy balance calculations was performed at the highest fan speed. A summary of the results is given in Table 5.1.

Table 5.1 Evaporator volumetric air flow rate comparison

Fan Speed	Manufacturer's CFM	Energy balance CFM	Uncertainty
High	379	404	52
Medium	338	341	--
Low	269	298	--

An air-side energy balance was obtainable only for the evaporator because condenser outlet air temperatures and velocities are too nonuniform to allow an accurate average temperature measurement. Since no contradiction between the estimated air flow rate and manufacturer's measurements for the evaporator was found, the condenser measurements were trusted as well. However, Section 5.5.1 identifies a possible error in the manufacturer's measurements of condenser air flow rates.

5.3 Compressor Map

The compressor model's prediction of power consumption dominates the COP calculation, and the compressor–along with the capillary tube–sets the system mass flow rate. Manufacturer-supplied compressor maps were used to predict w_{map} and pwr_{map}, the refrigerant mass flow and compressor power consumption, respectively, for 39 experimental data points.

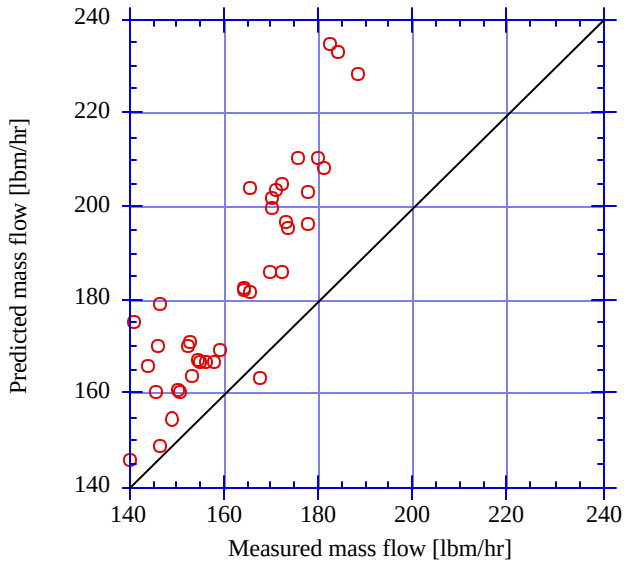

Figure 5.1 Compressor map mass flow prediction

Figure 5.1 shows that w_{map} significantly overpredicted the mass flow rate, with a 95% interval on the error of 14.4%, centered around an average error of 12.5%. As stated previously, the 95% confidence interval on the experimentally measured mass flow rate is 2.8%. The stated accuracy of the manufacturer's map is 5%.

An uncertainty analysis was also used to evaluate the effect of errors propagating through the compressor map, specifically errors in measuring condensing and evaporating temperatures and in pressure drop correlations. The analysis predicted a 2% error in the map's prediction of mass flow due only to propagation of inaccuracies in measured inputs. These uncertainties, when combined, do not fully account for the observed error in the map's predictions. The improved instrumentation that will be in place for the next stage of validation should improve our ability to evaluate the mass flow map's accuracy.

Figure 5.2 shows the compressor map's power consumption predictions, which had a 95% interval on error of 4.7%, centered around zero.

The ORNL model uses a compressor map correction that accounts for suction gas temperature variation. This correction was not used because it did not significantly improve the map's mass flow predictions and it made the compressor power predictions significantly worse.

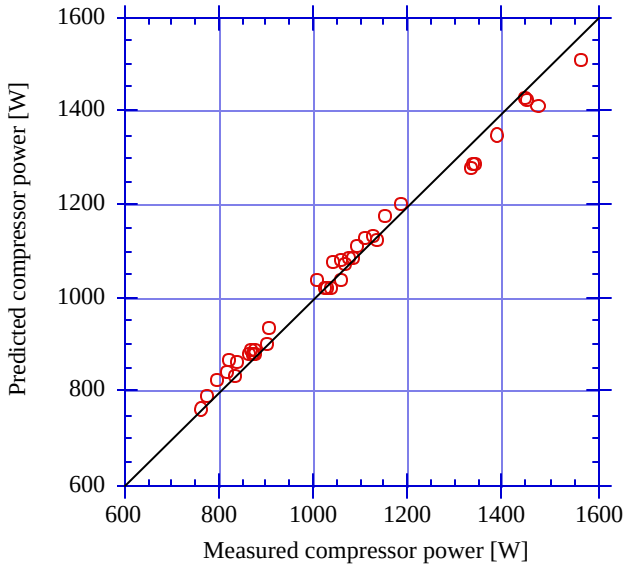

Figure 5.2 Compressor map power consumption prediction

5.4 Capillary Tube

As described in Section 2.4, the room air conditioner model allows the use of the ASHRAE capillary tube model or the ACRC capillary tube model. The capillary tube plays a critical role in determining the mass flow and low and high-side pressures in the system.

Both capillary tube models significantly overpredict mass flow rates, and both demonstrate an error trend with respect to the inlet subcooling and pressure, as shown in Figures 5.3 and 5.3. The 95% interval on error for the ACRC model is 20.0%, centered about an average error of 12.8%, and the 95% interval on error for the ASHRAE model is 18.0%, centered around an average error of 6.2%.

The uncertainty analysis given in Appendix H indicated that propagation through the capillary tube models of inaccuracies in measured inputs would account for 2.0% error in the ACRC model and 2.6% error in the ASHRAE model.

The overprediction of mass flow by the capillary tube will have a significant effect on the accuracy of the room air conditioner system model, and addressing this inaccuracy should take a high priority in future work. Capillary tube corrections based on inlet subcooling or pressure would be able to reduce the error in the capillary tube models, although a physical explanation for such corrections is not readily apparent.

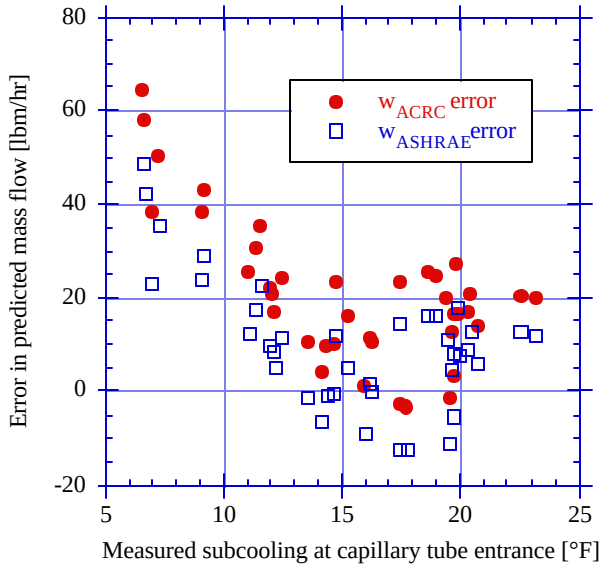

Figure 5.3 Capillary tube model error vs. subcooling

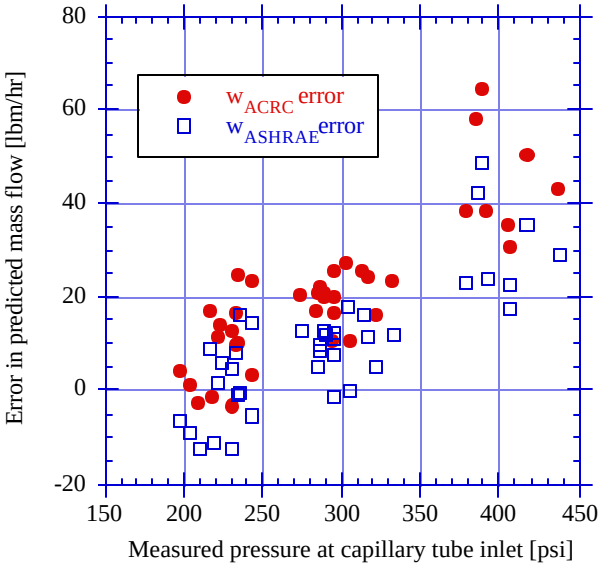

Figure 5.4 Capillary tube model error vs. inlet pressure

5.5 Condenser

5.5.1 Condenser Air Recirculation

Air recirculation may occur at the condenser, with condenser outlet air being drawn back through the inlet grille. This recirculation would cause the air temperature at the condenser inlet grille to be slightly higher than the outdoor temperature due to mixing with the warmer recirculated air. Data from 54 data points at three fan speeds were analyzed, and recirculation fractions ranging from about -0.01 to 0.02 were calculated, indicating that the actual amount of recirculation is too small to be discerned by our temperature measurements.

A different type of recirculation is clearly observable in our test unit, however. The condenser fins do not span the entire condenser exit grille, so that there is a gap of about an inch on either side of the condenser, through which exiting air is drawn back through the outlet grille to the lower pressure area upstream of the condenser. This recirculation can be readily observed and will actually suck a small string back into the outlet grill of the running air conditioner.

This type of recirculation will be called "internal" recirculation and must be handled differently than the external recirculation. The internally recirculated air never crosses a control volume boundary at the condenser outlet grille and would not pass through a volumetric flow measurement device. Thus the actual air volumetric flow through the condenser equals the measured flow plus the internally recirculated flow.

Figure 5.5 depicts both types of recirculation, where frecircC is the external condenser recirculation fraction, frecircCint is the internal condenser recirculation fraction, VdotaCmeas is the measured air flow rate, and VdotaC is the actual air flow rate through the condenser coil.

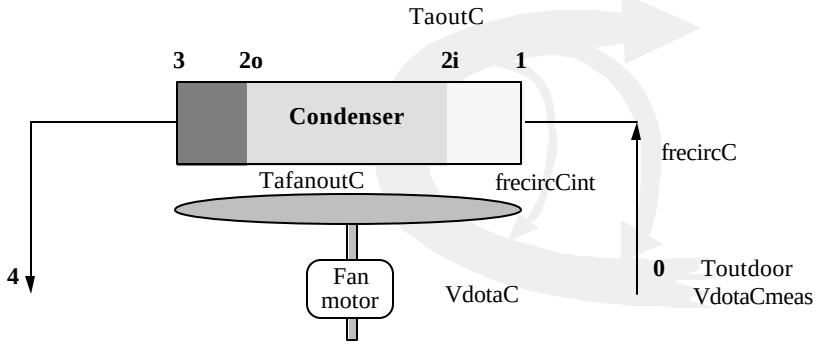

Figure 5.5 Condenser air flow recirculation

If, for example, the internal recirculation fraction were 0.1, then any measurement of condenser air mass flow would underestimate the actual mass flow by about 10% and the ?T for heat transfer would be reduced by 11.1%. These two affects are offsetting, so if one did not account for recirculation, i.e. used the measured air mass flow rate and did not compute the reduced ET, the COP would be underpredicted, because the underestimation of air mass flow has a greater effect than the overestimation of ET.

The magnitude of the internal recirculation fraction cannot be quantified using the limited air temperature measurements available, although the effect of a hypothetical change in the fraction can be evaluated using the room air conditioner model. A rough estimate of the internal recirculation was obtained by assuming a 0.75" strip of air on either side of the condenser flowing into the outlet grille at 100 ft/min, yielding a recirculation flow of 15.6 CFM, or about 2% of the *measured* condenser air flow rate.

If the air flow rate through the condenser coil is held constant, an internal recirculation of 0.02 will reduce the actual COP by about 0.01 (less than 1%). Thus the performance degradation due to internal recirculation on the room air conditioner in question does not appear to be significant.

5.5.2 Condenser Heat Transfer Predictions

The condenser model is described in Section 2.1. The condenser being modeled uses tubing that has been enhanced with internal microfins. Because heat transfer and pressure drop correlations for the enhanced tubing are

not available, refrigerant side heat transfer enhancement factors (EFrC) and pressure drop penalty factors (PFrC) are multiplied by the appropriate smooth tube correlations to account for the effects of microfinned tubes.

Limited data for condensing heat transfer enhancement factors are available, but Dobson (1994) has recommended using the area ratio between an enhanced tube and the equivalent smooth tube as a first (probably high) approximation. The area ratio of the tubing in the condenser being modeled (1.66) was used for EFrC. Christofferson (1994b) measured a pressure drop penalty factor of 1.37 for R22 in a similar enhanced tube, and this number is being used until pressure transducers are installed and PFrC can be evaluated directly for the test unit.

Using the parameter specifications above, the refrigerant temperature at the condenser exit (t3) was predicted for 26 data points where no condensate was being sprayed on the condenser. Accuracy in t3 corresponds to accuracy in the total heat transfer, and t3 directly relates to the amount of subcooling entering the capillary tube, which has a strong effect on system mass flow. Figure 5.6 compares the subcooling predictions with measured data. Each degree of subcooling overprediction represents an approximately 0.4% condenser heat transfer overprediction.

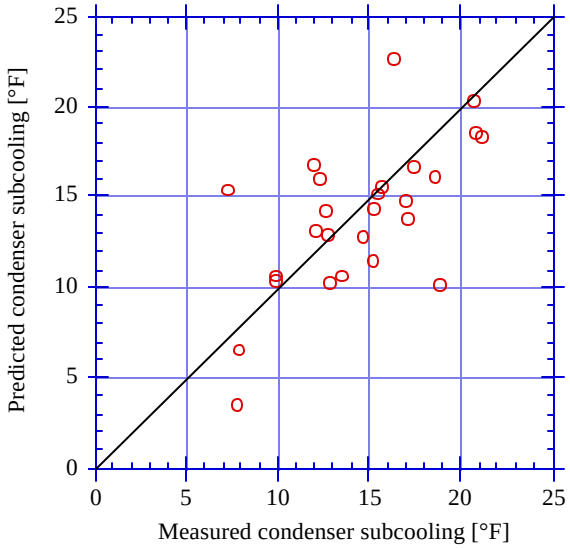

Figure 5.6 Condenser model's prediction of outlet subcooling for "dry" cases

5.5.3 Condensate Spray Heat Transfer Enhancement

When moisture removal occurs in the evaporator, the condensate is channeled to a "sling ring" on the condenser fan. The sling ring sprays the water onto the condenser, yielding a heat transfer enhancement that is examined in detail in Appendix I.

In reality, not all of the water coming from the evaporator ($\dot{m}_{total}$) will be utilized to enhance the condenser performance. A portion of $\dot{m}_{total}$ will be evaporated from the condensate pan ($\dot{m}_{pan}$) or from the "sling ring" or fan blade itself ($\dot{m}_{sling}$). Some water ($\dot{m}_{miss}$) may be slung onto the housing surrounding the fan and condenser rather than onto the condenser. Hopefully, the majority of the water ($\dot{m}_{eff}$, effective mass flow) will actually land on the coil and directly enhance heat transfer performance, though a portion of the water that does make it to the condenser could be blown completely through ($\dot{m}_{drift}$). Thus a mass balance on the water stream gives

$$\dot{m}_{eff} = \dot{m}_{total} - \dot{m}_{pan} - \dot{m}_{sling} - \dot{m}_{miss} - \dot{m}_{drift} . \qquad (5.3)$$

Order of magnitude calculations can estimate the importance of the various terms in the mass balance. Potential rates of water evaporation were estimated using the Lewis relation and a heat transfer correlation for laminar flow over a flat plate as given in Incropera and Dewitt (1990). These calculations indicated that evaporation from the condensate pan ($\dot{m}_{pan}$) would be around 0.05 lbm/hr and evaporation from the "sling ring" ($\dot{m}_{sling}$) would generally be less than 0.1 lbm/hr. These numbers are quite small compared to a $\dot{m}_{total}$ of 1 to 6 lbm/hr, which is typical of a room air conditioner. Observations of Tree and Goldschmidt (1978) and observations of air conditioners investigated in the current project indicate that $\dot{m}_{drift}$ is virtually zero, as no water is observed to be blown through the condenser. The quantity $\dot{m}_{miss}$ is difficult to estimate analytically because of lack of knowledge of the spray pattern, though it may prove to represent a significant mass flow of water evaporating into the air upstream of the condenser.

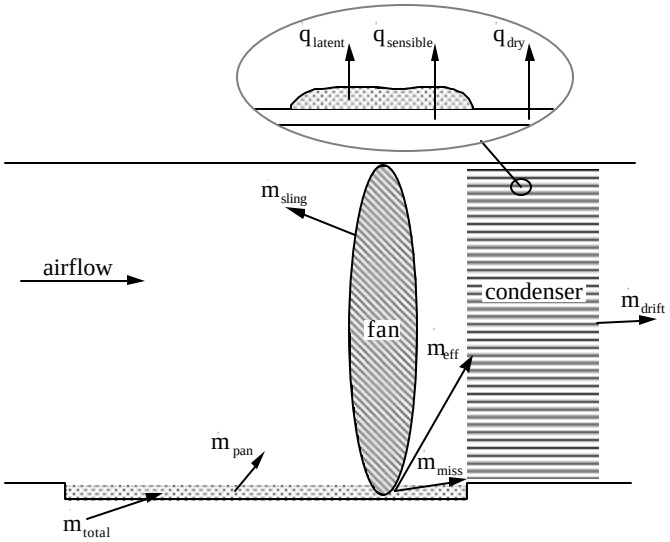

Figure 5.7 Condensate spray schematic.

Also, since each water droplet may initially be cooler than the air, it can draw a portion of its heat of vaporization from the air rather than the condenser fins, reducing the utilization of $\dot{m}_{eff}$'s cooling potential. However, the temperature difference between the water and the air would be significantly smaller than that between the water and the fin, and as the water becomes warmer than the air, it will begin sensibly heating the air rather than being heated by it. The conduction resistance between the water and the fin is also much smaller than the convection resistance between the water and the air. Thus the majority of the evaporative cooling potential should be drawn directly from the condenser fins.

In a typical condenser utilizing condensate spray, only a portion of the condenser surface will be wetted. Both latent and sensible heat transfer will occur on the wetted portion of the condenser. The latent transfer will be simply $\dot{m}_{eff}$ times the heat of vaporization of water.

The sensible heat transfer coefficient for the wet surface was calculated based on work by Myers (1967) and Threlkeld (1970) to be on the order of 10% higher for a wetted condenser. According to Tree and Goldschmidt's (1978) observations, generally less than 25% of the condenser is wetted by the spray, thus if the sensible heat transfer enhancement were 10%, the overall effect of the enhanced sensible heat transfer would be about 2.5%.

Considering the much larger uncertainty in evaporative losses prior to reaching the coil and the limits of our experimental accuracy, the sensible enhancement can reasonably be ignored, as Tree and Goldschmidt did.

Based on observations by Tree and Goldschmidt (1978), we expect that the enhancement due to the condensate spray would be affected most strongly by the mass flow of water supplied to the condenser pan. Geometric factors that determine how much of the sprayed water directly hits the condenser would also have a significant affect. The coil temperature and the humidity ratio of the ambient air should also affect the evaporative flux on a given area of the heat exchanger, but since excess water tends to be blown on to dry areas in actual operation, the overall heat transfer rate should not be affected strongly. However, the changes in the rate of evaporation before the water actually reaches the coil could have a significant effect on $\dot{m}_{eff}$.

A proposed form for the enhancement is given in Equation 5.4, where h_{fg} is the heat of vaporization of water, Q_{dry} is the total condenser heat transfer that would occur in a dry condenser for the same operating conditions, and Q_{wet} represents the enhanced total heat transfer when condensate spray occurs.

$$\dot{m}_{eff} = \dot{m}_{total} - h_m A * (\omega_o - \omega_\infty) - C_1 * \dot{m}_{total} \tag{5.4}$$

The term $h_m A$ represents an effective mass transfer coefficient times area (analogous to UA) for evaporation from the condensate pan, the cabinet, and other constant evaporations. The coefficient C_1 accounts for the physical efficiency of the fan in delivering water to the evaporator coil. Observations indicate that a significant portion of condensate is sprayed onto the condenser fan motor and the air conditioner cabinet, where it will draw heat from the those surfaces and thus not effect an heat transfer enhancement.

Predictions of heat transfer in the condenser can be made with the equation

$$Q_{wet} = Q_{dry} + \dot{m}_{eff} * h_{fg}, \tag{5.5}$$

where h_{fg} is the heat of vaporization of water, Q_{dry} is the total condenser heat transfer that would occur in a dry condenser for the same operating conditions, and Q_{wet} represents the enhanced total heat transfer when condensate spray occurs.

The coefficients $h_m A$ and C_1 in Equation 5.4 were evaluated based on experimental data taken with a room air conditioner testing facility. A least squares minimization indicated that the term containing $h_m A$ had a negligible effect, so $h_m A$ was set to zero. The estimated value for C_1 was 0.73, which, in this simplified correlation, must account for the physical efficiency of the sling ring as well as for small amounts of evaporation. Using this correlation, the condenser heat transfer enhancement ranges from about 8% to 34% of the total condenser heat transfer for the thirteen points tested. Tree and Goldschmidt indicated that heat transfer enhancements of up to 40% were obtainable.

Figure 5.8 shows the condenser model's prediction of subcooling for the 13 data points in which condensate spray occurred. Using the condensate spray enhancement, the 95% interval on subcooling error for all 39 data points was 7.8 °F, centered around an average error of -0.1 °F. This accuracy can be compared to a 95% interval on error of 6.6°F centered around 0.6°F for dry condenser cases alone.

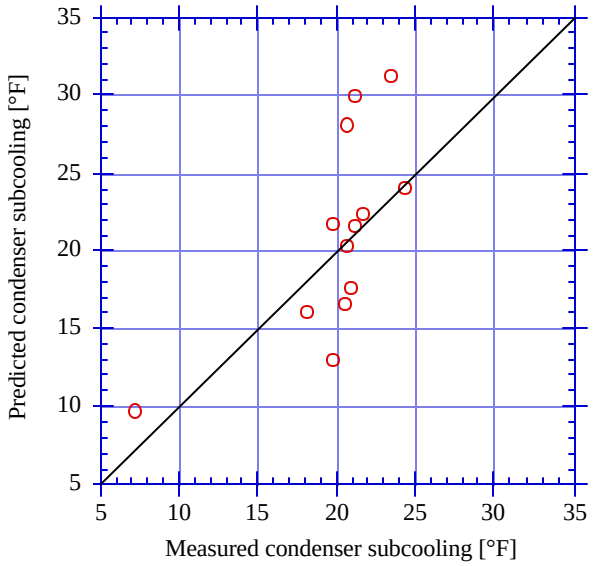

Figure 5.8 Condenser model's prediction of outlet subcooling for condensate spray

5.6 Evaporator

The evaporator model is described in Section 2.2. Like the condenser, the evaporator being modeled also uses microfin enhanced tubes, and an appropriate refrigerant-side heat transfer enhancement factor (EFrE) and pressure drop penalty factor (PFrE) are needed.

Christoffersen (1994) developed an enhancement factor correlation for evaporation, which was used to determine a value of 1.4 for EFrE for the evaporator being modeled. Christoffersen's measurement of 1.37 for the pressure drop penalty factor in a similar tube is also being used until the penalty factor for the evaporator can be measured with refrigerant-side pressure transducers.

Using the parameter specifications above, the evaporator model was used to predict the evaporator capacity and moisture removal for 39 data points. Figure 5.9 shows the evaporator capacity predictions versus measured values. The 95% bound on capacity error is 3.6%, centered around the average overprediction of 2.8%. This error is on the order of the uncertainty in the evaporator capacity measurement, which was determined to be 2.7% in Appendix H.

Figure 5.10 compares predicted and measured moisture removal rates and shows that the moisture removal rate is overpredicted by 0.7 lbm/hr on average, with a 95% interval of 0.8 lbm/hr centered around the average. In the context of a system model, the error in moisture removal prediction will adversely affect condenser model predictions, because the calculated moisture removal at the evaporator is used to determine the condensate spray enhancement in the condenser.

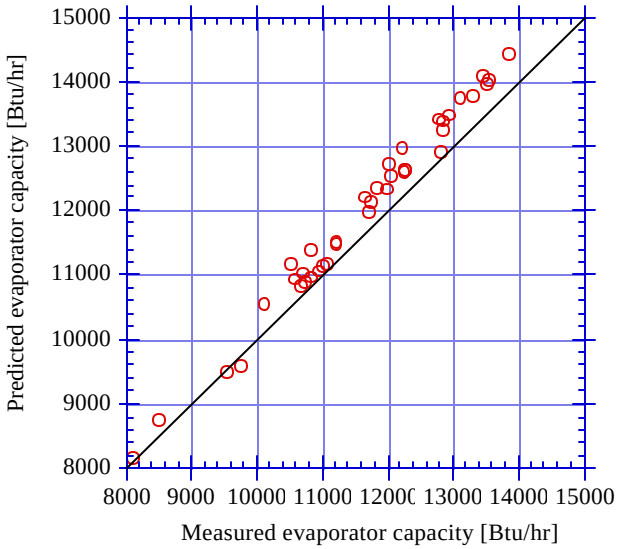

Figure 5.9 Comparison of predicted and measured evaporator capacity

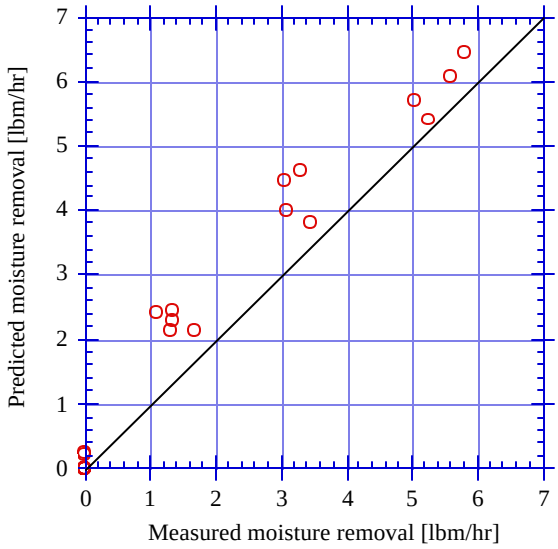

Figure 5.10 Comparison of predicted and measured evaporator moisture removal rates

5.7 System Model

The various components of RACMOD predict with varying degrees of accuracy, with the most notable problems being overprediction of mass flow rates by the compressor and capillary tube. The charge inventory equations also overpredict the total system charge, although this may be due primarily to errors in predicted subcooled and two-phase area fractions, which are influenced strongly by the evaporator and condenser models. The void fraction correlation and the estimated internal volume of the compressor are also potential sources of error.

At a rating condition of 80 °F indoor temperature and 95 °F outdoor temperature, RACMOD predicts a refrigerant charge of 1.65 lbm, compared to the actual charge of 1.375. In order to test the rest of the system model, the refrigerant charge was set to 1.65 lbm.

For testing the full system model, all component models were used without correction for known inaccuracies. The ASHRAE capillary tube model was used because of its smaller error. Indoor and outdoor room temperatures and humidities were specified, as were condenser and evaporator air flow rates and the fan power usage.

Figures 5.11 and 5.12 compare RACMOD's predictions for COP and evaporator capacity with measured values for 39 data points. The 95% interval on COP error is 0.71, centered around an average error of 0.33. The 95% interval for evaporator capacity error is 2800 Btu/hr, centered around an average error of 1460 Btu/hr. The evaporator capacity predictions of the system model are not as good as those of the evaporator component model alone because the system model's predictions are also influenced by the inaccuracies of the other component models.

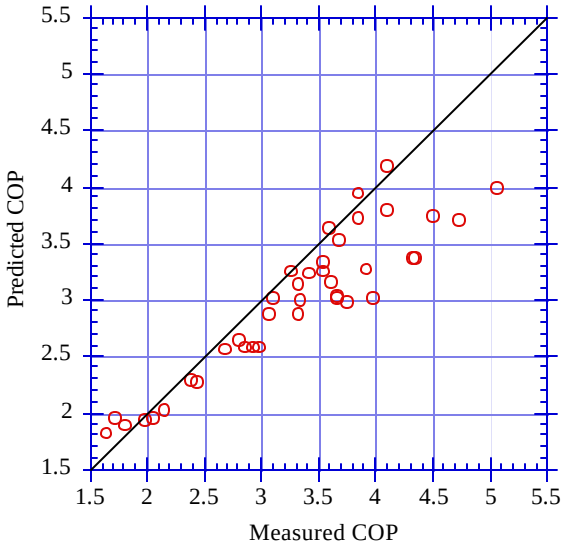

Figure 5.11 Comparison of predicted and measured COP

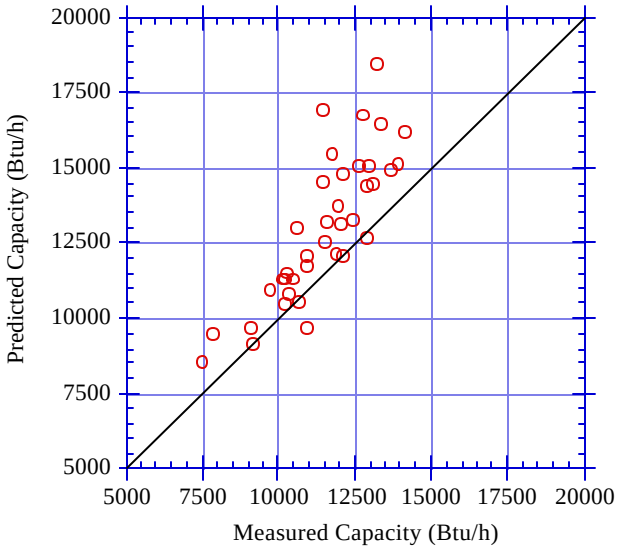

Figure 5.12 Comparison of predicted and measured evaporator capacities

References

Christoffersen, Brian, J. C. Chato, J.P. Wattelet, and A.L. de Souza. “Heat Transfer and Flow Characteristics of R-22, R-32/R-125 and R-134a in Smooth and Micro-fin Tubes.” University of Illinois at Urbana-Champaign, ACRC TR-47, 1993.

Dobson, M. K. "Heat Transfer and Flow Regimes During Condensation in Horizontal Tubes." University of Illinois at Urbana-Champaign, ACRC TR-57, 1994.

Fischer, S. K. and C. K. Rice. The Oak Ridge Heat Pump Models. Oak Ridge National Laboratory, 1983. ORNL/CON-80/R1.

Hahn, Gregory W. “Modeling Room Air Conditioner Performance.” University of Illinois at Urbana-Champaign, ACRC TR-40, 1993.

Hughmark, G.A. “Hold-up in Gas-liquid Flow.” *Chemical Engineering Progress*, 58 (4 1962).

Incropera, F.P. and D.P. DeWitt 1990. Introduction to Heat Transfer. 2nd ed. Wiley and Sons, New York.

Myers, R.J. 1967. "The Effect of Dehumidification on the Air Side Heat Transfer Coefficient for a Finned-tube Coil". Master's Thesis, University of Minnesota.

O'Neal, D. L. and S. B. Penson. An Analysis of Efficiency Improvements In Room Air Conditioners. Texas A&M University, 1988. ESL/88-04.

Pate, M. B. “A theoretical and experimental analysis of capillary tube-suction line heat exchangers.” Ph. D., Purdue University, 1982.

Peixoto, R. and C. W. Bullard. "A Design Model for Capillary Tube-Suction Line Heat Exhangers." University of Illinois at Urbana-Champaign, 1993. ACRC TR-53, 1994.

Porter, K. J. and C. W. Bullard. Modeling and Sensitivity Analysis of a Refrigerator/Freezer System. University of Illinois at Urbana-Champaign, ACRC TR-31, 1992.

Souza, A. L., J. C. Chato, J. M. S. Jabardo, J. P. Wattelet, J. Panek, B. Christoffersen, and N. Rhines. Pressure Drop During Two-Phase Flow of Refrigerants in Horizontal Smooth Tubes. University of Illinois at Urbana-Champaign, ACRC TR-25, 1992.

Threlkeld, J.L. 1970. Thermal Environmental Engineering. 2nd ed. Prentice Hall, Englewood Cliffs, New Jersey.

Tree, D.R., V.W. Goldschmidt, *et al*. 1978. "Effect of Water Sprays on Heat Transfer of a Fin and Tube Heat Exchanger." Proceedings of Sixth International Heat Transfer Conference, Vol. 4.

Wattelet, J. P. "Heat Transfer Flow Regimes of Refrigerants in a Horizontal -Tube Evaporator." University of Illinois at Urbana-Champaign, ACRC TR-55, 1994.

Chapter 6: Summary and Conclusions

The ACRC room air conditioner model has been developed based on governing equations from the ORNL heat pump model. Modifications have been made to the model including updated heat transfer and pressure drop calculations and the addition of charge inventory equations. A first-principles finite difference capillary tube model has been implemented using a "backsubstitution" technique that allows the capillary tube model's 200 equations to be solved by adding only four simultaneous equations to the system model. A calculation of condenser heat transfer enhancement due to condensate spray has also been added.

The ACRC solver, a specialized Newton-Raphson equation-solving and analysis program, was developed for RACMOD, the ACRC refrigerator model, and other models being developed in the ACRC. The solver allows variable and parameter swapping without recompiling, contains speed and robustness enhancements, and allows the modeler to implement subroutines to be called before, during, or after the solution process for purposes of variable checking, pre or post-processing, or implementing equation switching in a model.

The first stage of instrumentation for an air conditioner – utilizing air side thermocouples only – has been completed and measurements have been taken at 54 data points. These data establish a baseline for future evaluations of the intrusive effects of installing refrigerant-side thermocouples and for quantifying the error associated with surface thermocouples versus refrigerant side thermocouples.

The baseline data was also used here to make preliminary comparisons between predictions of the component models and experimental data. It identified several deficiencies in some of the first-principles component models that must be addressed with the help of more accurate and extensive experimental instrumentation in the second stage of the experimental program. The capillary tube and void fraction models for charge calculation are among the most difficult to accurately model from first principles, and they appear to be the limiting factors currently in the accuracy of RACMOD.

Appendix A: ACRC Equation Solver User's Reference

A.1 Introduction

The ACRC equation solver is a Fortran program for solving systems of nonlinear algebraic equations and for performing sensitivity and uncertainty analysis on those systems of equations. The original impetus for developing the program was to solve the systems of equations that make up the ACRC room air conditioner model (RACMOD) and refrigerator models. The RAC model is used here as an example model.

The solver is intended for research and design purposes, with a file-based user interface that provides flexibility in specifying of the program's operation and output. The organizational framework for the ACRC solver is based on a TrueBasic sensitivity analysis program described by Porter and Bullard (1992). Running the solver is straightforward--no programming knowledge is required, although making modifications to a model requires a basic understanding of Fortran. Parameter-variable switching allows a model to be easily switched from a simulation mode to a variety of design modes without re-compiling.

Understanding the governing equations is necessary to properly use any model, and the solver's use of the Newton-Raphson method allows all of the governing equations to be listed together in a single subroutine so that they are easier to understand and modify. Sparse matrix Gaussian elimination and automated step relaxation enhance the solver's speed and robustness, as described in Sections 3.3 and 3.4. The solver is also compatible with sparse matrix Jacobian calculation, which can be implemented in a model as described in Section B.4.

A.2 Equation Solver Fundamentals

A.2.1 Newton-Raphson Method

A thorough explanation of the Newton-Raphson (NR) method of numerically solving a set of nonlinear algebraic equations is given by Stoecker (1989). The NR method requires that the equations to be solved be written in "residual" format. For example, R_i = RHS - LHS would be the residual format of the equation LHS = RHS, where RHS and LHS are the right and left hand sides of the governing equation, respectively. "R_i = RHS - LHS" is called the i^{th} "residual equation," and R_i is called the i^{th} "residual value." If there are n equations and variables in the system to be solved, then a solution has been found when all residual values, $R_1 ... R_n$, are equal to zero. In practice, the system is considered solved when the residual values are less than a specified tolerance.

The first step in the NR method is the calculation of the Jacobian (**J**), which is done numerically by the ACRC solver. The Jacobian is the (n x n) coefficient matrix for the set of linear equations that best approximates the nonlinear set at a particular **x**, where **x** is the vector of variable values. The Jacobian is defined by

$$J = \begin{bmatrix} \frac{\partial R_1}{\partial x_1} & \cdots & \frac{\partial R_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial R_n}{\partial x_1} & \cdots & \frac{\partial R_n}{\partial x_n} \end{bmatrix}$$

where n is the total number of equations. Once the Jacobian is calculated, the linear system $\mathbf{J} \cdot (\mathbf{x} - \Delta\mathbf{x}) = \mathbf{R}$ is solved for $\Delta\mathbf{x}$, where **R** is the vector of residual values. Then, for the linear approximation, $\mathbf{J} \cdot (\mathbf{x} - \Delta\mathbf{x})$ will yield

residual values of zero. The correction to the variables, $\mathbf{x} = \mathbf{x} - \Delta\mathbf{x}$, is made, and the residual values of the actual nonlinear equations are calculated. The process of linear approximation and solution is repeated until all of the residual values are less than a user-specified tolerance.

The Newton-Raphson method requires initial guesses for each variable in order to calculate the first residual values and Jacobian. The NR method is not globally convergent, so poor guesses may not allow the program to converge on a solution, as described in Section 3.4.

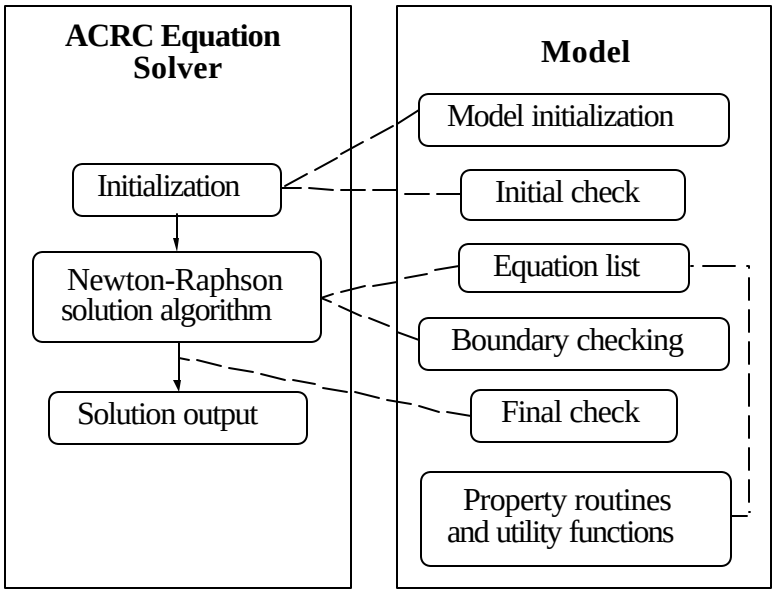

Figure A.1 Solver-model relationship

A.2.2 ACRC Solver Terminology

In the ACRC solver, variables in the governing equations are referred to by the letter "X", with X values being synonymous with variable values. Similarly, all user-specified parameters in the modeling equations are referred to by "K," and residual values are referred as "R" values. The X's and K's are stored together in the array XK, and a variable or parameter's position in that array determines its XK#.

It is important to understand what is meant by "model" and "solver." The solver is the Fortran program that keeps track of most input and output files and either solves a given system of governing equations or performs a sensitivity or uncertainty analysis on them. Strictly speaking, "model" refers to the governing equations that represent the system to be simulated. In the context of the ACRC solver, "model" also encompasses the Fortran subroutines that handle simulation-specific initialization tasks, check the XK values, and calculate the residual values for the governing equations. The solver-model relationship is depicted in Figure A.1, although the options of multiple runs and sensitivity and uncertainty analysis in the solver are not shown.

A.3 Input File Descriptions

Three files are used as input by the solver and model: a solver settings file, an instructions file, and an initialization file. These three files are described below, and an example analysis is given in Section A.4.

A.3.1 Initialization File

The XK initialization file, an abbreviated sample of which is given in Table A.1, contains input values for all of the model's parameters and initial guesses for all of its variables.

Table A.1 Sample XK initialization file

```
 ** XK initialization file: initializes variable guesses and parameter values.
 **   Parameters are flagged with 'K' and variables are flagged with 'X'
 **   The units are delimited with '[ ]'.
 **   The last number signifies the number of decimal places (0-10).
 ** The ORDER of the input lines CANNOT CHANGE without program modification.
 Flag Name                  XK#            Value  Units         # of digits
 ***********   DO NOT DELETE THESE FIRST SEVEN LINES!    ***************
X  COP                  = XK(  1) =           1.74 [ ]                  2
X  EER                  = XK(  2) =           5.95 [Btu/hr-W]           2
X  degsubcool           = XK(  3) =            0.0 [F]                  1
:
K  Toutdoor             = XK(162) =          111.9 [F]                  1
K  Tindoor              = XK(163) =           60.0 [F]                  1
K  RhaiC                = XK(164) =          0.420 [ ]                  3
:
```

The X and K flags indicate whether each line is specifying a variable or a parameter. The names correspond to the Fortran variable names for the model's variables and parameters, and the numbers in parentheses are the "XK #'s." The values are initial guesses for variables or specified values for parameters. Units must be delimited by square brackets, and the number of digits to be output in solution files is specified by the number in the rightmost column. The order of the names in the XK file cannot be changed without modifying parts of the program code, as is described in Section B.3.4.

A.3.2 Instructions File

The four analyses that may be performed with the ACRC solver are denoted "SINGLE," "MULTIPLE," "SENSITIVITY," and "UNCERTAINTY." The "SINGLE" analysis simply loads parameter values and initial guesses from the XK file, solves the equations with the NR method, and outputs the solution. The "MULTIPLE" analysis allows solutions of the equations to be performed for multiple sets of parameter values, so that a large number of parametric runs can be performed conveniently. The "SENSITIVITY" analysis finds the influence coefficients of the variables with respect to specified parameters, i.e. $\frac{\partial x_j}{\partial k_i}$. The "UNCERTAINTY" analysis performs either an ASME root-sum-square (RSS) or additive (ADD) uncertainty analysis or a Monte Carlo (MC) uncertainty analysis on the equations. The uncertainty analyses and their implementation are described in Appendix H.

The instructions file ("INSTR" in the case of the sample solver settings file in Table A.5) directs the execution of the program for each of the four analyses. Tables A.2-A.3 provide line-by-line explanations of sample instructions files for the single, multiple, and sensitivity analyses, and the uncertainty analysis is discussed in Appendix ?.

The instructions file specifies the type of analysis to be performed (the name is case-sensitive), the name of the XK initialization file, an extender to be appended to all output filenames, and any other information that may be

necessary for execution of the analysis. The separator '***' may be used to separate multiple sets of instructions, which will be performed in sequential order. In the instructions file, both variables and parameters are referred to by their XK#'s rather than by name.

Table A.2 contains instructions for two single runs, the first using the file "XK" as an initialization file and appending "out" to the output file names, while the second reads parameter values and initial guesses from the file "XK2" and appends "out2" to all output filenames.

Table A.2 Sample instructions file for "SINGLE" analysis

Example	Description
SINGLE	Type of analysis (solve for a single set of parameters)
XK	Name of initialization file
out	Extender to append to output file names
***	Separator (allows multiple sets of instructions in one file)
SINGLE	See above
XK2	"
out2	"

For a multiple analysis, the base parameter values and initial guesses are loaded from the initialization file. The instructions file for a multiple analysis contains a list of parameters which are to have their values modified between solutions, as well as a list of values for those parameters.

The user also specifies the number of intermediate steps to take between solutions. If the number of intermediate steps is one or more, then the "MULTIPLE" analysis will direct the NR solver to perform solutions for that number of intermediate solutions--linearly interpolating between the previous specified parameters and the next specified parameters. This enables a smooth transition from one set of parameters to another when a model is very sensitive to initial guesses. The number of intermediate steps should remain set to zero if no problem occurs in solving a multiple run, because the "extra" solutions for the intermediate steps increase the execution time.

As specified in the solver settings file, the solutions can be saved individually as initialization files and combined in a tab-delimited file to be read by a spreadsheet. Intermediate solutions are not saved.

Table A.3 Sample instructions file for "MULTIPLE" analysis

Example	Description
MULTIPLE	Type of analysis (solve for multiple sets of parameters)
XK	Name of initialization file
mlt	Extender to append to output file names
3,2,1	# of runs, # of parameters to vary, # of intermediate steps
250,274	XK#'s of the parameters to be varied
80.0,95.0	List of specified parameter values for run #1
90.0,95.0	List of specified parameter values for run #2
90.0,105.0	List of specified parameter values for run #3

For a sensitivity analysis, a solution is first performed for the parameter values given in the initialization file, and then influence coefficients are calculated for the parameters listed in the instructions file. In addition to the standard influence coefficients, $\frac{\partial x_j}{\partial k_i}$, a relative influence coefficient, $\frac{\partial x_j}{x_j} \Big/ \frac{\partial k_i}{k_i}$, is also calculated, which indicates the percent change in x_j (expressed in decimal format) given a percent change in k_i. The influence coefficients are saved to the file "SA.*extender*" in a tab-delimited format.

Table A.4 Sample instructions file for "SENSITIVITY" analysis

Example	**Description**
SENSITIVITY	Type of analysis (find influence coefficients)
XK	Name of initialization file
sns	Extender to append to output file names
5	Number of parameters for which to calculate influence coefficients
161,162,166,206,247	XK#'s of parameters for which to calculate influence coefficients

A.3.3 Solver Settings File

The file "SLVERSET" contains settings for various solver parameters, convergence criteria and tolerances, and output options. The sample SLVERSET file in Table A.5 is followed by descriptive notes. Some of the output settings (printing initial, intermediate, and final XK and R values) are primarily useful for debugging a model, and normally the solver settings do not need to be changed. However, the iteration summaries and final summaries may be turned off during a Monte Carlo uncertainty analysis to eliminate excessive output.

Table A.5. Sample solver settings "SLVERSET" file

Sample File	Note
**********	
******** NEWTON-RAPHSON SETTINGS ********	
Instruction file name : INSTR	1
Step factor for partial derivatives : .0001	2
Maximum allowable NR iterations : 60	3
Convergence criteria 1(Maximum residual): 1.0e-6	4
Convergence criteria 2 (RMS residual) : 1.0e-7	5
Selected convergence criteria (1 or 2) : 1	6
NR step relaxation parameter : 1.0	7
Use sparse matrix techniques? : .TRUE.	8
Update guesses between runs? : .TRUE.	9
******** GENERAL OUTPUT SETTINGS ********	
Send general output to screen? : .TRUE.	10
Send general output to a file? : .FALSE.	11
Print abbreviated solver settings? : .TRUE.	12
Print initial XK values? : .FALSE.	13
Print initial residual values? : .FALSE.	14
Print iteration summaries? : .TRUE.	15
Print intermediate XK values? : .FALSE.	16
Print intermediate residual values? : .FALSE.	17
Print final XK values? : .FALSE.	18
Print final residual values? : .FALSE.	19
Print a final summary : .TRUE.	20
******** SOLUTION OUTPUT SETTINGS ********	
Save XK values in input file format? : .TRUE.	21
Save XK values in spreadsheet format? : .TRUE.	22
Output digits 0-10 (-1 = as in XK file) : -1	23

Notes:

1. Specifies the name of the file from which instructions for particular analyses are to be read.
2. When numerical partial derivatives are taken ($\frac{\partial R_i}{\partial x_j} = \frac{\Delta R_i}{\Delta x_j}$) this factor is used to determine Δx_j, i.e. $\Delta x_j = x_j \cdot (\text{step factor})$.
3. Maximum number of iterations allowed before the NR routine prematurely terminates.
4. Maximum residual convergence criteria.
5. Root-mean-square (RMS) convergence criteria.
6. Selects which of the above two convergence criteria are to be used.

7. A parameter that is multiplied by the correction to the x values for each NR step, i.e. $\mathbf{x} = \mathbf{x} - (\text{relaxation parameter}) \cdot \Delta\mathbf{x}$.
8. Use sparse matrix Gaussian elimination techniques if "true."
9. If "true," during a multiple run the solution for the first set of parameters will be used as initial guesses for the second solution, and that solution will be used for the third, etc. If "false," guesses are read from the initialization file before each solution.
10. Specifies that output is sent to the screen during program execution.
11. Specifies that output is sent to a file named "O.*extender*" during program execution.
12. Specifies that part of the solver setting information be printed in the general output.
13. Print XK values in a condensed format before the solution begins. In the condensed format, the integer followed by a colon on each row is the XK or residual number corresponding to the first value on that row. For the example shown in Table A.2, XK #5 has the value 0.59003 and XK #8 has the value 576.27.

Table A.2 Sample condensed XK value output.

```
XK Values :
   1 :   5.5023       6.3092      0.72624      0.83445
   5 :  0.59003       955.20       744.13       576.27
   9 :   526.99       49.283       112.85       98.217
  13 :   112.85       54.291       76.664       62.327 .......
```

14. Print residual values in a condensed format before the solution begins.
15. Print a summary of iteration #, step relaxation parameter, and maximum and RMS residual.
16. Print XK values in a condensed format after each iteration.
17. Print residual values in a condensed format after each iteration.
18. Print XK values in a condensed format after solution.
19. Print residual values in a condensed format after solution.
20. Print summary of number of iterations, maximum residual, and RMS residual after solution.
21. Save the solution to a file "XK.*extender*" in the format of an XK initialization file.
22. Save the solution to a file "S.*extender*" in a tab-delimited format for use by a spreadsheet.
23. If this number is an integer from 0 to 10, then the specified number of output digits in the XK file will be overridden, and the number given here will be used instead.

A.4 Running the Solver

A.4.1 Single Run

To solve a model (RACMOD being used here as an example) for a given set of input parameters, the three input files must be located within the same directory as the compiled solver program. The file "SLVERSET" must contain the name of the instructions file to be used, and the instructions file should be set up as in the first part of Table A.2. The XK file should contain the parameter values for the case for which a solution is desired.

When the executable name of the compiled solver ("rac" in this example) is entered, the following should appear:

```
$ rac

  Single Run using extender: out

  Rebuild the NonZeroList?
   "y" if residual equations have been modified or if parameters and/or
    variables have been swapped (y/n).
```

Because the NonZeroList keeps track of which Jacobian elements will always be zero (see Sections 3.3 and B.4), it must be rebuilt when the equations are modified. Swapping variables and parameters indirectly changes the equations, so this also necessitates rebuilding the NonZeroList, as does switching between the two available capillary tube models. When the program is run for the first time with a new or modified model, "y" should be entered. Building the NonZeroList involves taking all n^2 partial derivatives for the Jacobian, where n is the number of equations, so the building process may take a few minutes for a large equation set. The NonZeroList is saved to the file "NONZEROL.IST" so that it can be read from a file in the future to save time.

After selecting "y," the following will appear:

```
Creating NonZeroList.
 Using regular evaporator equations.
 Using regular condenser equations.
 Using regular cap-tube equations.
```

The statements indicating that "regular" equations are being used are printed by RACMOD's initial checking subroutine, which determines, for example, whether the initial parameter values and variable guesses indicate that the evaporator exit is "regular" or two-phase. Once the NonZeroList has been built, the output will continue and is shown here for a case where the output flags are set as in Table A.5. The second and third iterations are deleted for brevity.

```
NonzeroList saved to file "NONZEROL.IST"
 Using regular evaporator equations.
 Using regular condenser equations.
 Using regular cap-tube equations.

******** NEWTON-RAPHSON SETTINGS ********
Instruction file name                  :    INSTR
Step factor for partial derivatives    : 0.100E-03
Maximum allowable NR iterations        :    60
Convergence criteria 1(Maximum residual): 0.100E-03
Convergence criteria 2 (RMS residual)  : 0.100E-04
Selected convergence criteria (1 or 2) :    1
NR step relaxation parameter           :  1.00
Use sparse matrix techniques?          :    T
Update guesses between runs?           :    T

Maximum residual:  1828.8      in Eqn #111
    RMS residual:  146.35

Beginning iteration #  1
Took a Newton-Raphson step with Beta =   1.0000
Maximum residual:  66.998      in Eqn #101
    RMS residual:  6.2030
:
Beginning iteration #  4
Took a Newton-Raphson step with Beta =   1.0000
Maximum residual: 0.20118E-08  in Eqn #111
    RMS residual: 0.23042E-09
```

```
Finished in  4 iterations.
Maximum residual: 0.20118E-08
    RMS residual: 0.23042E-09

Writing solution as an initialization file: XK.out
Writing spreadsheet-readable solution file: S.out
 Program Execution Complete.
```

"Beta" is the current value of the NR step relaxation parameter. The "EQN #" indicates the number of the residual equation that has the maximum residual value, which can be useful in debugging a new model.

A.4.2 Parameter-variable swapping

In a normal simulation mode, the total refrigerant charge in the system is specified as a parameter (Mtotal) and the degrees of subcooling is a variable. If one wanted to instead specify a particular amount of subcooling and solve for the amount of charge that would be required, only two flags need to be changed in the XK initialization file. The lines

```
X  degsubcool          = XK(  3) =          12.1 [F]                 1
K  Mtotal              = XK(170) =       1.4335 [lbm]               4
```

would have to be changed to

```
K  degsubcool          = XK(  3) =          12.1 [F]                 1
X  Mtotal              = XK(170) =       1.4335 [lbm]               4
```

Then the solver can be run as described above. Caution must be exercised when swapping variables and parameters to ensure that the equations are not made singular or non-independent because of the swapping. When the configuration of the equations is changed through swapping, the model may also become more sensitive to initial guesses.

It is recommended that a solution of the original equation set be found before swapping, and that solution be used as a basis for the new XK file with swapped parameters and variables. This ensures that the initial guesses will be very good for the first solution attempted with the new configuration.

If the equations are very sensitive to a particular parameter, making a large change in the value of that parameter may cause difficulty in solving. In this case, it is best to achieve the desired parameter change with a series of intermediate steps, with each intermediate solution being used as the initial guesses for the next step. The multiple analysis option is convenient for this technique, although solution output files can also be renamed and used as input files for the single analysis.

A.4.3 Automated step relaxation

If a NR step is taken that does not decrease either the maximum residual value or the root-mean-square residual value, the solver will untake that step, halve the step size, and take the smaller step. The process will be repeated if successive steps do not improve the residual values, although if the residuals do not improve after reducing the step size to 2^{-10}, the solution will be terminated with an error.

In the course of a solution, the model's equations may attempts to divide by zero or raise non-integer numbers to negative powers, and other numerical problems may occur. If the ACRC solver is run on a machine that does not halt program execution immediately upon the occurrence of such errors, the subroutine VectorInfo will check to see if any residuals have the value NaN ("Not a Number".) If they do, it will give that residual a value of 99^{99}. Thus when the maximum and RMS residual values are checked, they will have increased and the automated step relaxation discussed above will untake the bad step.

Automated step relaxation greatly increases the robustness of the solution process, and consequently reduces the required accuracy in initial guesses.

A.4.4 Troubleshooting

Care must be taken for all input files to ensure that floating point numbers contain a decimal point and that numbers which should be integers do not contain a decimal point. Otherwise, the input information may be misinterpreted.

There should also always be at least one empty line at the end of any of the input files, or the last line of input may not be read. This may cause the solver to falsely report that not enough variables or parameters were found.

A good way to ensure that the parameter and variable values are being read properly is to select in SLVERSET the option to print initial XK values. These values can be checked against the XK file to ensure their accuracy.

If there is difficulty solving for a particular set of parameters, a solution can often be found by starting with a previous solution and gradually changing the parameter values to the desired ones, as described in Section A.4.2.

Choosing a smaller NR relaxation parameter will sometimes improve the solution robustness, although it will also increase the required number of NR iterations for a solution.

If there is a singularity in the Jacobian, the program will return a message that either one equations contains no variables or one variable does not appear in any equation. These problems may be due to improperly switching variables and parameters or from a problem with the equations themselves.

Specifying that the NonZeroList be rebuilt will solve convergence problems if the NonZeroList has become out-of-date by X-K swapping or otherwise altering the equations.

References

Porter, K. J. and C. W. Bullard. Modeling and Sensitivity Analysis of a Refrigerator/Freezer System. University of Illinois at Urbana-Champaign, ACRC TR-31, 1992.

Stoecker, W. F. Design of Thermal Systems. 3rd ed., New York: McGraw-Hill, 1989.

Appendix B: ACRC Solver Programming Reference

B.1 Overview of ACRC Solver Features

The ACRC Solver is intended for general use with any system of nonlinear governing equations, although it contains features that make it particularly useful for thermal systems problems. The solver is a multifunction program for solving equations, as well as performing other analyses, and outputting the results. The solver is distinguished from the model according to the definition given in Section A.2.2.

The solver uses a Newton-Raphson algorithm that performs automatic step relaxation and allows the governing equations to be "switched" in mid-solution. It optionally outputs information about variable, parameter, and residual values and solution progress that is useful for model development and debugging. Model-specific "checking" subroutines are accessed before, during, and after the solution process so that checking and pre- and/or post-processing of NR variables and parameters can occur, or other model-specific tasks can be performed.

The solver's NR variable handling structure allows variables and parameters to be referred to by name in the governing equations rather than by number and allows variables and parameters to be "swapped" in the model without recompilation of the Fortran code. The solver numerically calculates the Jacobian and provides information to the model that allows the implementation of a sparse-matrix Jacobian calculation method within the model, though its implementation is not required.

The solver can perform multiple parametric solutions of the governing equations as well as sensitivity and uncertainty analyses, and output for solutions and analyses is produced in spreadsheet-readable and XK initialization file formats.

Pseudocode for the ACRC solver program, along with details of the file structure and subroutine descriptions is provided in Appendix C.

B.2 The Solver-Model Interface

In order to use the solver's features, guidelines for solver-model interface must be followed, as described below. The sample model in Appendix D demonstrates the essentials of the solver-model interface, as well as the implementation of sparse-matrix Jacobian calculation and equation "switching" in a model. Following the example of Appendix D and of the ACRC room air conditioner model listing in Appendix K is the best way to ensure that a new model interfaces properly with the solver.

B.2.1 Use of "Included" Files

The main method of communication between the solver and model is Fortran common blocks which allow variables to be shared between various subroutines. These common blocks, along with many frequently-used variable declarations, are usually located in "include" files, denoted by the extender "INC." These files are included in subroutines via the Fortran "include" statement. Table B.1 shows which "include" files are used by particular subroutines of the model and solver, and Table B.2 describes the contents of the various "include" files.

Table B.1 "Include" files in the solver and model

Include File Name	Model Subroutines	ACRC Solver Subroutines
DIMENSN.INC	InitializeModel, CreateNonZeroList, ReadNonZeroList	SingleRun, MultipleRun, SensitivityRun, UncertaintyAnalysis, SensCoeffCalc, GetX, PutX, NRMethod, CalcD, CheckForSingular, InitializeModel, CalcR, Nzero, XGauss, Gauss, OutputInitFile, OutputSprdShtFileHeader, OutputSprdShtFileLine
EQUIVLNT.INC	CalcR, Initialize, IC, BC, FC	SensitivityRun, MultipleRun, UncertaintyAnalysis, SensCoeffCalc, GetX, PutX, XKPrnt, OutputInitFile, OutputSprdShtFileHeader, OutputSprdShtFileLine
ERRCOM.INC	CalcR, IC, BC, FC	MAIN ACRCsolver, GetInstructions, GetSeparator, GetSolverSettings, SingleRun, MultipleRun, SensitivityRun, UncertaintyAnalysis, NRMethod, InitializeModel, XGauss, Gauss, StoreError, PrintErrorlist, OutputSprdShtFileLine
GAUSSCOM.INC		SensCoeffCalc, NRMethod, Nzero, XGauss
GENCOM.INC	IC, BC, FC	MAIN ACRCsolver, GetInstructions, GetSeparator, GetSolverSettings, SingleRun, MultipleRun, SensitivityRun, UncertaintyAnalysis, SensCoeffCalc, NRMethod, CalcD, XKPrnt, VectPrnt, PrintSolverSettings
NISTCOM.INC	Initial, psatt, tsatp, hpt, htx, hpx, vpx, vtx, vpt, saturation, xph, cvpt, cvpsat, spt, spx	
NRCOMMON.INC	CreateNonZeroList	MultipleRun, SensitivityRun, UncertaintyAnalysis, SensCoeffCalc, NRMethod, CalcD, CheckForSingular, Nzero, XGauss, Gauss
XANDKCOM.INC	InitializeModel, CreateNonZeroList	SingleRun, MultipleRun, SensitivityRun, UncertaintyAnalysis, SensCoeffCalc, GetX, PutX, NRMethod, CalcD, CheckForSingular, Nzero, XGauss, Gauss
XKCOM.INC	InitializeModel	MultipleRun, SensitivityRun, UncertaintyAnalysis, SensCoeffCalc, GetX, PutX, OutputInitFile, OutputSprdShtFileHeader, OutputSprdShtFileLine

Table B.2 Description of "include" files

Include File Name	Description
DIMENSN.INC	Defines the Fortran parameters Xmax, Kmax, and XKmax, which specify the maximum allowable number of variable, parameters, and the sum of Xmax and Kmax, respectively, for use in array declaration statements.
EQUIVLNT.INC	Declaration and common block for the XK array, and "equivalence" statements that cause the elements of the XK array to be interchangeable with corresponding variable and parameter names
ERRCOM.INC	Declarations and common blocks pertaining to error handling
GAUSSCOM.INC	Declarations and common blocks used by the Gaussian elimination routine
GENCOM.INC	Declarations and common blocks for all variables set by the SLVERSET file
NISTCOM.INC	Declarations and common blocks for property functions
NRCOMMON.INC	Declarations and common blocks for the residual array, the Jacobian, the delta X step array, etc. for the Newton-Raphson routine
XANDKCOM.INC	Declarations and common blocks for X, Xnm, NumVar, and NumPar
XKCOM.INC	Declarations and common blocks for the XK name and flag arrays and Xmap

B.2.2 The XK and X Arrays

Two important features of the solver are that variable and parameter names may be used directly in the governing equations (unlike most "canned" Newton-Raphson solvers) and variables and parameters can be "swapped" without recompiling, as described in Section 3.2. In order to facilitate these features, a system of handling NR variables and parameters was created as described below.

The primary storage location for variables and parameters is the XK array, the elements of which are made "equivalent" to Fortran variables that bear the names of the variables and parameters as they occur in the governing equations. For example, in the room air conditioner model, XK(1) and COP are "equivalent" and thus may be used interchangeably. As discussed in Section A.2.2, parameters and variables are usually referred to by their XK# in the solver, while the governing equations and IC, BC, and FC routines refer to them by name. The XK array and the "equivalent" Fortran variable names are declared and put in a common block in EQUIVLNT.INC so that they can be easily included in different subroutines.

The Newton-Raphson and Jacobian calculation subroutines require that the variables be in a single array, X. The array Xmap keeps track of which elements of the XK array are variables. For example, Xmap(1) = (XK# of the first variable) and Xmap(23) = (XK# of the 23rd variable). The subroutine GetX is used by NRMethod and CalcD to transfer values from XK to X, and PutX transfers values back from X to XK.

B.3 Modifying a Model

Appendix D demonstrates the requirements for the solver-model interface and is helpful for understanding how to modify an existing model. The checklist in Section B.3.1 covers the main issues involved with modifying a model, and the subsequent sections provide details for specific modifications.

B.3.1 Model Modification Checklist

1. There must be an equal number of residual equations and NR variables and they must be defined so that the equations are neither singular nor non-independent.
2. The Fortran parameters Xmax, Kmax, and XKmax in DIMENSN.INC must be large enough to accommodate all variables and parameters, and XK must be dimensioned in EQUIVLNT.INC to have the same size as XKmax.
3. The XK initialization file and EQUIVLNT.INC must be consistent in variable and parameter ordering and all model parameters and variables must be declared as double precision Fortran variables in EQUIVLNT.INC. All XK#'s should be consecutive.
4. The number of parameters and variables (NumPar and NumVar) specified in InitializeModel must be correct.
5. If sparse-matrix Jacobian calculation is being used, all line labels in the "computed GOTO" statement in CalcR must be consistent with the labels of the appropriate residual equations.
6. For sparse-matrix Jacobian calculation, any equation that may sometimes not reflect its dependence on all of the variables that it may *ever* contain should be augmented with a auxiliary residual equation that is executed only when NonZeroFlag is true. The replacement equation should contain every variable or parameter that may ever occur as a variable in that equation. (See Section B.4 and the example in Appendix D.)
7. The IC, BC, and FC subroutines should be appropriately programmed for the specific needs of the model, and if they are not needed, they should be left as empty subroutines.
8. All functions and auxiliary Fortran variables used by the governing equations should be properly declared in CalcR.

B.3.2 Changing Equations

Making a modification to existing equations only involves making the changes in the governing equations in CalcR and recompiling the program, unless parameters are removed or added, as described in Section B.3.2.

B.3.2 Adding or Deleting Parameters or Variables

If changes to the governing equations involve the removal of old parameters or variables or the creation of new ones, the appropriate changes in the XK initialization file, EQUIVLNT.INC, and InitializeModel must be made in accordance with checklist items 2, 3, and 4.

B.3.3 Adding or Deleting Equations

Changing the number of equations entails changing the number of variables as described in Section B.3.2. The appropriate line labels should be inserted in or removed from the "computed GOTO" statement in CalcR, and the residual equations should be numbered consecutively and consistently with the "computed GOTO" statement. The room air conditioner model uses pointers that indicate the residual equation number at which a particular component model begins. This allows equations to be added to or removed from a component model without renumbering the residuals in all subsequent components--only the pointers to subsequent components need to be changed. Checklist item 6 also applies when an equation is added.

B.3.4 Rearranging the XK Initialization File

The variables and parameters in the XK initialization file can be listed in any order as long as the conditions of checklist item 3 are met.

B.3.5 Modifying IC, FC, and BC

If new code is to be included in any of the checking subroutines, the appropriate common blocks must be included in the subroutines. In most cases, only the file "EQUIVLNT.INC" should be needed. Appendix D gives more information on implementing the checking routines.

B.4 Implementation of Sparse-matrix Jacobian Calculation

This section provides an overview of sparse-matrix Jacobian calculation and summarizes the motivation for its use. Individual aspects are discussed further throughout this thesis.

Section 3.3 explains that sparse-matrix Jacobian calculation is necessary to reduce the number of partial derivatives that must be calculated when generating the Jacobian, which would otherwise require n^2 partial derivative evaluations, where n is the number of equations.

The logic that must be included with the governing equations to implement sparse-matrix Jacobian calculation makes them less readable, but it reduces execution time for the RACMOD and other large models by a factor of 6 to 9 times and is thus well worth the clutter in the equations.

The concept behind sparse-matrix Jacobian calculation is simple. In the initialization stage of running a model, the full Jacobian matrix of n^2 partial derivatives is calculated, which takes a few minutes for large models. The Jacobian is then scanned to determine which elements are nonzero, and this information is stored in the two-dimensional array NonZeroList. Later, only Jacobian elements that may potentially be nonzero are actually calculated, and the rest are assumed to be zero.

Table B.3 gives an example Jacobian for a set of four equations, and Table B.4 depicts the corresponding NonZeroList. Recall that the Jacobian is the matrix of the partial derivatives of residual equation values with respect to the NR variables.

$$J = \begin{bmatrix} \frac{\partial R_1}{\partial x_1} & \cdots & \frac{\partial R_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial R_n}{\partial x_1} & \cdots & \frac{\partial R_n}{\partial x_n} \end{bmatrix}$$

Table B.3 Sample Jacobian matrix

	x_1	x_2	x_3	x_4
R_1	.54	0	0	0
R_2	0	-.02	0	0
R_3	2.6	0	-4.55	0
R_4	-.22	.61	0	3.8

Table B.4 Sample NonZeroList

#0	#1	#2	#3	#4
1	1	2	3	4
2	3	4	0	0
3	4	0	0	0
4	0	0	0	0

The sample NonZeroList indicates that variable #1 occurs in residual equations #1, 3, and 4; Variable #2 occurs in the second and fourth residual equations, etc. Column #0 of NonZeroList is used when it is necessary to evaluate all residual values at once*.

When the Jacobian matrix is being calculated in the course of the NR solution, CalcR must be called once for every variable with respect to which a partial derivative is being calculated. The residual values R thus attained are used to numerically calculate the derivatives $\frac{\Delta R}{\Delta x}$. Only those residual values which are affected by a given variable need to be calculated in a given call to CalcR. CalcR contains logic that evaluates only those equations which are affected by a particular variable. The demonstration model in Appendix D illustrates this logic, which uses a "computed GOTO" statement to execute equations individually, using the NonZeroList to determine which equations must be evaluated when calculating the partial derivatives with respect to a particular variable.

As explained in Appendix A, each time the solver is run, the user is asked whether the NonZeroList should be rebuilt. The NonZeroList is saved to the file "NONZEROL.IST" each time it is generated, so that if the equations have not changed in any way, the NonZeroList can simply be loaded from a file, saving a significant amount of time.

---

* Once per NR iteration, all of the residual values must be evaluated at once so that their maximum and root-mean-square residual values can be found. Column #0 of NonZeroList contains a list of all equation numbers and is used for this purpose.

The NonZeroList must be rebuilt when the equations are changed directly or when they are changed indirectly by variable-parameter swapping or selecting a different capillary tube model with the CapTubeSelect parameter.

Certain governing equations may not always reflect their dependence on every variable that they may ever contain. For example, in the equation $R_n = y - \max(a,b)$, $\frac{\partial R_n}{\partial a}$ will be nonzero when a>b, and $\frac{\partial R_n}{\partial b}$ will be nonzero when b>a. In order to ensure that the NonZeroList accounts for every Jacobian element that may *ever* be nonzero, an auxiliary residual equation is required that is only executed when the NonZeroList is being generated. The flag NonZeroFlag is set true when the NonZeroList is being built, so the statement "If (NonZeroFlag) Then $R_n = y + a + b$" would ensure that the NonZeroList indicates that residual #n may have nonzero partial derivatives with respect to y, a, or b.

Appendix C: ACRC Solver Pseudocode and Subroutine Descriptions

The program structure of the solver is depicted in Figure C.1 and outlined in detail by the pseudocode presented below. Only the major subroutines and program operations are shown. Pseudocode is given for the main program and for the subroutines SingleRun, MultipleRun, SensitivityRun, UncertaintyAnalysis, and NRMethod. The actual subroutine names are used in the "Call" statements. An overview of the file organization with brief descriptions of all of the solver's subroutines is given in Table C.1.

Within the pseudocode, the notation "(in model)" indicates that the subroutine being called is actually part of the "model" program, not the solver.

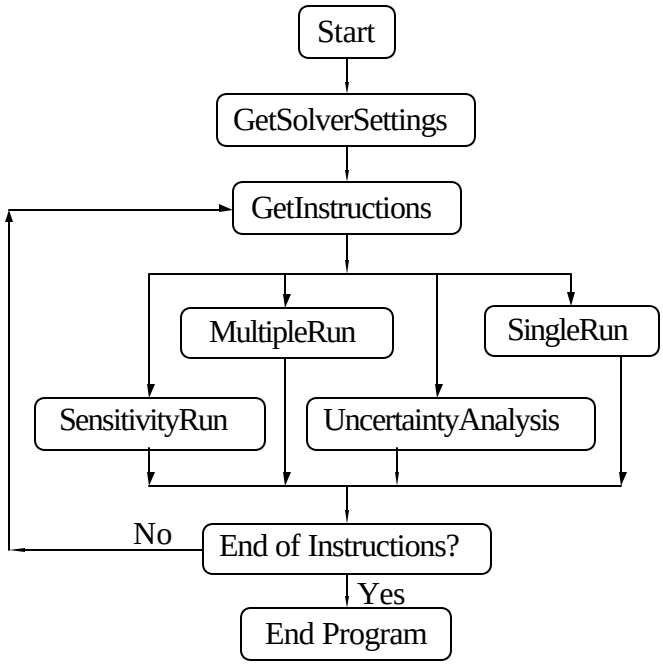

Figure C.1 Flowchart of solver organization

The ACRC solver is organized into many files which are compiled using a "make" file that automatically compiles and links the subroutines in the various files. The use of "include" files within various subroutines is discussed in Section B.2.1.

Table C.1 ACRC solver file structure and subroutine descriptions

File or Subroutine Name	Description
MAIN.f	This file contains the main program structure of the ACRC solver as depicted in Figure C.1
MAIN ACRCsolver	Main program unit
GetInstructions	Reads the analysis type, XK filename, and output filename extender from the instructions file
GetSeparator	Looks for the separator "***", which indicates the start of a new set of instructions
GetSolverSettings	Reads the information in the file SLVERSET
shortn	Concatenates two strings
SINGLE.f	This file contains the subroutine for the "SINGLE" analysis
SingleRun	Performs a solution of the governing equations for a single set of input parameters, as given in the XK initialization file
MULTIPLE.f	This file contains the subroutine for the "MULTIPLE" analysis
MultipleRun	Performs multiple solutions of the governing equations using a base set of input parameters from the XK file, with specified parameters modified according to the instructions file
SENSTVTY.f	This file contains subroutines for the "SENSITIVITY" analysis
SensitivityRun	Performs a sensitivity analysis, i.e. calculation of influence coefficients, for parameters that are specified in the instructions file
SensCoeffCalc	Calculates the influence coefficients
UNCRTNTY.f	This file contains subroutines for the "UNCERTAINTY" analysis
UncertaintyAnalysis	Performs an ASME, ADD, or RSS uncertainty analysis or a Monte Carlo analysis according to instructions in the instructions file
ran	Random number generator
rect	Rectangular distribution generator
normal	Normal distribution generator
NRMETHOD.f	This file contains subroutines for performing the Newton-Raphson method
NRMethod	Solves the governing equations using a modified Newton-Raphson method
CalcD	Numerically calculates the Jacobian matrix
CheckForSingular	Checks for singularities in the Jacobian
VectorInfo	Calculates the maximum and root-sum-square values of a vector (usually the residual value array) and changes NaN to 99e99
AssVect	Assigns one array the values of another
VectDiff	Subtracts two arrays, storing the result in a third array
VectAdd	Adds two arrays, storing the result in a third array
ScaMult	Multiplies an array by a scalar
MatDif	Subtracts one matrix from another
GetX	Transfers NR variable values from the XK array to the X array
PutX	Transfers NR variable values from the X array to the XK array
GAUSSIAN.f	This file contains subroutines for Gaussian elimination
Nzero	Used to create the linked list of non-zero Jacobian elements for XGauss
XGauss	Performs sparse-matrix Gaussian elimination with full pivoting
Gauss	Performs standard Gaussian elimination with full pivoting

OUTPUT.f	This file contains subroutines for program output
StoreError	Stores error codes for later printing by PrintErrorList
PrintErrorList	Prints the errors stored in ErrLst to the file *instruction-file-name*.EF.
XKPrnt	Prints the XK array in a concise format using VectPrnt
VectPrnt	Prints any array in a concise format
PrntIntArray	Prints an integer array, only used for debugging purposes
PrintSolverSettings	Prints a portion of the solver settings
OutputInitFile	Saves a solution file in the same format as an XK initialization file
OutputSprdShtFileHeader	Saves a header of labels for a solution file in a tab-delimited format
OutputSprdShtFileLine	Saves a solution on one line of the solution file in a tab-delimited format
"Model" Subroutines	These subroutines are part of the model, but are briefly described here
InitializeModel	Reads XK initialization file, builds or loads the NonZeroList for sparse-matrix Jacobian calculation, and performs other model-specific tasks
CalcR	Calculates the residual values of the governing equations
IC, BC, and FC	These subroutines perform model-specific checking or other tasks before, during, or after the Newton-Raphson solution process

C.1 Main Program: ACRCsolver

```
      Call GetSolverSettings
1     Call GetInstructions
      Which analysis is to be performed?  Call the appropriate one below
            Call SingleRun
            Call MultipleRun
            Call SensitivityRun
            Call UncertaintyAnalysis
      Has the end of the instructions file been reached?
            No, Call GetSeparator and Goto 1
            Yes, End the program
```

C.2 Subroutine: SingleRun

```
      Call InitializeModel  (in model)
      Call NRMethod
      Call OutputInitFile
      Call OutputSprdShtFileHeader
      Call OutputSprdShtFileLine
      Return
```

C.3 Subroutine: MultipleRun

```
      Call InitializeModel (in model)
      Read # of runs, # of parameters to vary, and # of intermediate runs from instructions
      Read the list of parameters to vary and all parameter values from instructions
      Call OutputSprdShtFileHeader
1     If not "updating," Call InitializeModel  (in model)
     XKold=XK
      Set parameter values according to values read from instructions
      If # of intermediate runs = 0 then Goto 3
      XKdiff = XK - XKold
      XK = XKold + XKdiff*(1/(1+ # of intermediate runs))
      Do 2, n=1, # of intermediate runs
            Call NRMethod
```

2 XK = XK + XKdiff*(1/(1+ # of intermediate runs))
Call NRMethod
Call OutputSprdShtFileLine
Call OutputInitFile
Finished all runs?
No, Goto 1
Yes, Return

C.4 Subroutine: SensitivityRun

Call InitializeModel (in model)
Read number of parameters and parameter list from the instructions file
Call SensCoeffCalc
Write influence coefficients to the file SA.*extender*
Return

C.5 Subroutine: UncertaintyAnalysis

Call InitializeModel (in model)
Read all information for the uncertainty analysis from the instructions file
Call NRMethod
ASME analysis or Monte Carlo?
ASME:
Call SensCoeffCalc
Calculate bias, precision, and uncertainty
Combine bias and precision into uncertainty according to ADD or RSS method
Goto 5
Monte Carlo:
If MCF analysis (analyzing previous MC runs), Goto 3
1 Generate parameter values according to distributions specified in the instructions
Call NRMethod
Was the solution successful?
No, save parameters and error flag to the MCtemp file and Goto 2
Yes, save selected variables to the MCtemp file
2 Restore original XK values
Have enough Monte-Carlo runs been completed?
No, Goto 1
Yes, proceed
Was one of the attempted solutions unsuccessful?
Yes, print warning and Return
No, Goto 4
3 Read the results of a previous MC analysis from file
4 Calculate offset, precision, and uncertainty values
Generate a histogram of variable values and save to the file H.*extender*
Locate the boundaries of the actual 95% coverage interval for each variable
5 Save MC output to a file MC.*extender*
Return

C.6 Subroutine: NRMethod

Call IC (in model)
Call CalcR (in model)
Display initial XK and R values as specified by SLVERSET
1 Are the residual equations solved?
Yes, Goto 3
No, proceed
Call CalcD (calls CalcR repeatedly, in model)

```
    Call CheckForSingular
    Call Gaussian elimination routine (XGauss or GAUSS)
    Call GetX
    Calculate a Newton-Raphson step
2   Scale the Newton-Raphson step by the relaxation parameter
    Take the Newton-Raphson step
    Call PutX
    Call BC  (in model)
    Call CalcR  (in model)
    Display intermediate XK and R values and summary as specified by SLVERSET
    Did neither the maximum residual nor the RMS
     residual decrease or did a "not a number" occur?
        No, the step is good, Goto 1
        Yes, the step is poor, proceed
    Untake the last step
    Halve the NR step relaxation parameter
    Is the step size too small?
        Yes, terminate execution (Goto 3)
        No, now take the shorter step (Goto 2)
3   Call FC
    Print final XK and R values and summary as specified by SLVERSET
    Set error flag 'true' if a solution was not reached
    Return
```

Appendix D: Demonstration Model

D.1 Introduction

A four equation sample model is presented here to demonstrate the essential features of a model as implemented with the ACRC solver. These routines can be used as an "empty" model in which to insert new modeling equations.

The sample model demonstrates the utilization of sparse-matrix Jacobian calculation, although its use in the model is not required by the solver. Comments in the code indicate lines that may be deleted if sparse-matrix Jacobian calculation is not desired.

The use of the IC and BC subroutines for equation-switching is also demonstrated here, but if a new model has no need for extra processing or checking before, during, or after the Newton-Raphson solution, the IC, BC, and FC routines may be left empty, although they must exist because they are always called by the solver.

D.2 CalcR

The CalcR (calculate residuals) subroutine contains the governing equations written in residual format. It is evaluated repeatedly by CalcD for Jacobian calculation and once per Newton-Raphson iteration for determination of all of the residual values at once. If sparse-matrix Jacobian calculation is not used, CalcR may contain only a simple list of the residual equations, as shown in Section D.2.2.

D.2.1 CalcR Using Sparse-matrix Jacobian Calculation

Sparse-matrix Jacobian calculation introduces extra logic among the equations, but greatly reduces execution time for large models.

```
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

        Subroutine  CalcR(VariableNum,R)

C***********************************************************************
C
C   PURPOSE:
C     Calculate the values of the residual equations.  A model's
C     governing equations are converted into residual format according
C     to the following example:
C           Eqn #1 :  LHS = RHS  --->   R(1) = RHS - LHS
C     Where LHS and RHS are the left-hand and right-hand sides of the
C     equation respectively.  The equation is considered solved when
C     the residual value is equal to zero within a specified
C     tolerance.
C
C     CalcR is called repeatedly by CalcD (Jacobian matrix
C     calculation) and is called by NRMethod once per Newton-Raphson
C     iteration.
C
C   INPUTS:
C     VariableNum  - current variable number
C          If VariableNum=0, all of the residual values are to be
```

```
C          determined at once.
C          Otherwise, VariableNum represents the variable with respect
C            to which the partial derivatives are being taken.
C
C   SHARED IN COMMON BLOCKS:
C       NonZeroList   - list of nonzero elements of the Jacobian to
C         facilitate sparse-matrix Jacobian calculation.
C         Column #n of NonZeroList contains a list of all of the
C         residual equations that contain variable #n.
C         Column #0 of NonZeroList contains a list of ALL residual
C         equations and is used when all of the residual values are
C         to be determined at once.
C     NonZeroFlag - this flag is set .true. only when the NonZeroList
C         is being built.  This information is needed when a
C         particular residual equation does not always show its
C         dependence on all of the variables that it may contain, e.g.
C         when certain equations may be switched during the course of
C         a solution or a function such as 'max' or 'min' is used.
C         The Jacobian that is calculated during building the
C         NonZeroList is not used for any calculations, thus it does
C         not to be the true Jacobian, but it must be non-zero
C         EVERYWHERE that the true Jacobian will EVER be non-zero.
C     XK - the array of variable and parameter values
C     All variable and parameters - these are included in EQUIVLNT.INC
C         and are made equivalent to XK array elements via the
C         'Equivalence' statement
C
C     OUTPUT
C     R - the array of residual values
C
C***********************************************************************

        Implicit None

        Include 'DIMENSN.INC'
        Integer VariableNum
        Double Precision R(Xmax)

C ***  EQUIVLNT.INC  declares all the variables and parameters and
C ***  sets them to be equivalent to elements of the XK array.
        Include 'EQUIVLNT.INC'

C *** The common block below contains flags that are used to switch
C *** equations in this sample model in mid-solution.  If equation-
C *** switching is not required in a model, no such common block is
C *** required.
        Logical Curve1
        Common/EqnFlags/Curve1


C *** All of the code from this point to the Governing Equations is
C *** necessary only if sparse-matrix Jacobian calculation is being
C *** used, as are the 'Goto 100' statements after every equation.
```

```
        Integer EQNUM, ELEMENT
        Integer NonZeroList(Xmax,0:Xmax+1)
        Common/NonZeroBlock/NonZeroList
        Logical NonZeroFlag,AlreadyAsked
        Common/NZL/NonZeroFlag,AlreadyAsked

C *** The following code implements the sparse-matrix Jacobian
C *** calculation. If partial derivatives are currently being taken with
C *** respect to variable #n, the list of equation numbers contained in
C *** column #n of NonZeroList indicates which residual equations may
C *** possibly be affected by a change in variable #n.  The 'computed
C *** Goto' statement below enables CalcR to execute only those
C *** equations that are affected by variable #n.

        ELEMENT=1

C *** The commented-out write statements are useful for tracking the
C *** execution of the equations when debugging a model
C       Write(*,*)
C       Write(*,*) 'VariableNum = ',VariableNum

100     EQNUM=NonZeroList(ELEMENT,VariableNum)
        ELEMENT=ELEMENT+1

C        Write(*,*) EQNUM

C *** The computed goto statement allows each residual equation to be
C *** accessed individually. This facilitates sparse-matrix Jacobian
C *** calculation by making it possible to evaluate only those equations
C *** which contain the variable with respect to which a partial
C *** derivative is currently being taken.

C *** If EQNUM=1, Goto 1000; if EQNUM=2, Goto 1020; etc.  If EQNUM=0,
C *** the 'computed Goto' statement will not execute.

        Goto (1000, 1020, 1040, 1060), EQNUM

        If (EQNUM .EQ. 0) Goto 5000

C ********************************
C ****** GOVERNING EQUATIONS ******
C ********************************

C *** Note that the equations in this sample have no real meaning, but
C *** are intended only to demonstrate the basic model and solver
C *** features.

1000    R(1) = Param1*1.5 - Var1
        Goto 100

1020    R(2) = Param2 - Var2
        Goto 100

C *** This is an example of an equation that would not indicate its
```

```
C *** dependence on both Var1 and Var2 if the NonZeroFlag were not used.
1040    If (NonZeroFlag) then
            R(3) = Var1 + Var2 + MaxVar
        Else
            R(3) = max(Var1,Var2) - MaxVar
        Endif
        Goto 100

C *** This is an example of equation switching.  The flag Curve1 is
C *** first set in IC and then checked in BC according to the
C *** criteria:
C ***   If MaxVar<=200 then Curve1 = .true., otherwise Curve1 =.false.
C *** The NonZeroFlag must be used to ensure that the NonZeroList
C *** reflects that the partial derivative of R(4) with respect to both
C *** Var1 and Var2 MAY be non-zero at any given time
1060    If (Curve1) then
            R(4) = 100. + Var2*5 - y
        Else
            R(4) = 300. + Var1*4 - y
        Endif
        If (NonZeroFlag) R(4) = Var1 + Var2 + y
5000    Return
        End
```

D.2.2 CalcR Without Sparse-matrix Jacobian Calculation

For a small model, sparse matrix Jacobian calculation is not warranted, and in this case the previous CalcR subroutine could appear as follows (with comments deleted for brevity).

```
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
        Subroutine  CalcR(VariableNum,R)

        Implicit None
        Include 'DIMENSN.INC'
        Integer VariableNum
        Double Precision R(Xmax)
        Include 'EQUIVLNT.INC'
        Logical Curve1
        Common/EqnFlags/Curve1

C *********************************
C ****** GOVERNING EQUATIONS ******
C *********************************
        R(1) = Param1*1.5 - Var1
        R(2) = Param2 - Var2
        R(3) = max(Var1,Var2) - MaxVar
        If (Curve1) then
            R(4) = 100. + Var2*5 - y
        Else
            R(4) = 300. + Var1*4 - y
        Endif

        Return
        End
```

D.3 InitializeModel, CreateNonZeroList, and ReadNonZeroList

The InitializeModel subroutine's first executable lines are:

```
C ***   Specify the number of parameters and number of Newton-Raphson
C ***   variables in the model.
      NumPar = 2
      NumVar = 4
```

These lines must be updated whenever the number of parameters or variables is changed. The InitializeModel subroutine also reads the XK initialization file and administrates building the NonZeroList if sparse-matrix Jacobian calculation is being used. The "Initial" subroutine to initialize the REFPROP property routines may also be called here. Because the only difference between these subroutines in this demonstration model and in RACMOD are the specified number of parameters and variables and that the subroutine Initial is not called by the demo model, the reader is referred to the ACRC room air conditioner model program listing in Appendix ¥ for details of the subroutines.

D.4 IC, BC, FC

The IC (initial checking) subroutine for the demonstration model checks the value of the variable MaxVar and sets an equation-switching flag accordingly. The BC (boundary checking) routine performs a similar check, and the FC (final checking) subroutine is left empty. A simple model may have no use for these three subroutines, although they must exist even if they are empty.

```
C*****************************************************************
      Subroutine IC

C      PURPOSE - The IC (Initial Check) routine for the model may be
C           used for pre-processing of variable initial guesses or
C           parameter values, checking XK values and setting equation
C           flags accordingly, or performing any other model-specific
C           operations before the start of the Newton-Raphson solution
C           process.

        Implicit None
        Include 'EQUIVLNT.INC'

C *** The common block below contains flags that are used to switch
C *** equations in this sample model in mid-solution.  If equation-
C *** switching is not required in a model, no such common block is
C *** required.
        Logical Curve1
        Common/EqnFlags/Curve1

C *** Set the flag Curve1, which is used for equation switching
        If (MaxVar .le. 200.) then
            Curve1 = .true.
            Write(*,*) '  Using curve #1.'
        Else
            Curve1 = .false.
```

```
            Write(*,*) '  Using curve #2.'

         Endif

         Return
         End

C*********************************************************************
      Subroutine BC(AbortStep,Switch)

C      PURPOSE - The BC (Boundary Check) routine for the model may be
C           used to check variable values and ensure that they fall
C           within certain boundaries.  The BC subroutine is also
C           intended to facilitate switching of equations in
C           mid-solution. Based on the variable values, logical flags
C           may be set that "switch" model equations in or out.

         Implicit None
         Include 'EQUIVLNT.INC'
         Logical AbortStep, Switch

C *** The common block below contains flags that are used to switch
C *** equations in this sample model in mid-solution.  If equation-
C *** switching is not required in a model, no such common block is
C *** required.
         Logical Curve1
         Common/EqnFlags/Curve1

C   OUTPUTS:
C       AbortStep - if .true., tells the solver that the NR step that
C           has just been taken should be aborted, the Jacobian
C           recalculated, and a new step taken.  It may be necessary
C           sometimes when equations are switched.
C       Switch - If .true., indicates to the solver that an equation
C           flag has been switched, in which case the automatic step
C           relaxation will be overridden for this iteration.

         AbortStep = .false.
         Switch = .false.

         If (Curve1 .and. (MaxVar.gt.200.)) then
             Write(*,*) ' MaxVar>200. Switching to curve #2. '
             Curve1 = .false.
             Switch = .true.
         Endif
         If (.not. Curve1 .and. (MaxVar.le.200.)) then
             Write(*,*) ' MaxVar<=200. Switching to curve #1. '
             Curve1 = .true.
             Switch = .true.
         Endif

         Return
         End
```

```
C*********************************************************************
      Subroutine FC

C      PURPOSE - The FC (Final Check) routine for the model may be
C           used for post-processing of variable or parameter values,
C           checking XK values and setting equation flags accordingly,
C           or performing any other model-specific operations after the
C           Newton-Raphson solution is complete.

        IMPLICIT NONE

C *** NO FINAL CHECKING IS IMPLEMENTED IN THIS DEMONSTRATION MODEL

        Return
        End
```

D.5 EQUIVLNT.INC

EQUIVLNT.INC contains the declarations of all variables and parameters used in the model, along with "Equivalence" statements that make the variable and parameter names interchangeable with elements of the XK array. It is included in every subroutine where the values of the XK array or individual variables and parameters are needed (see Table B.1). The order in which the variables and parameters occur in this file must exactly match the order of the XK initialization file.

```
C ********************************************************************
C
C   This file contains Equivalence statements that allow variable and
C   parameter names from the model to share memory locations with
C   elements of the XK array, thus when one is changed, the other is
C   automatically updated.

C   The file is included anywhere the variables or parameters need to be
C   accessed by name or by XK#.
C ********************************************************************

        Double Precision XK(300)
        Common/XKtogether/XK

C *****  Variable and parameter name declarations *****
        Double Precision Var1, Var2, MaxVar, y, Param1, Param2

C *** Make XK elements equivalent to named variables ***
      Equivalence     ( XK(  1), Var1          )
      Equivalence     ( XK(  2), Var2          )
      Equivalence     ( XK(  3), MaxVar        )
      Equivalence     ( XK(  4), y             )
      Equivalence     ( XK(  5), Param1        )
      Equivalence     ( XK(  6), Param2        )
```

D.6 XK Initialization File

The XK initialization file must contain the proper number of variables and parameters, flagged appropriately. The variables and parameters must appear in the same order as they do in the EQUIVLNT.INC file.

```
 ** XK initialization file: initializes variable guesses and parameter values.
 **   Parameters are flagged with 'K' and variables are flagged with 'X'
 **   The units are delimited with '[ ]'.
 **   The last number signifies the number of decimal places (0-10).
 ** The ORDER of the input lines CANNOT CHANGE without program modification.
 Flag Name                   XK#            Value  Units         # of digits
 ***********   DO NOT DELETE THESE FIRST SEVEN LINES!    ***************
X  Var1                 = XK(  1) =            2.  [ ]                1
X  Var2                 = XK(  2) =            1.  [ ]                1
X  MaxVar               = XK(  3) =            1.  [ ]                0
X  y                    = XK(  4) =            1.  [ ]                2
K  Param1               = XK(  5) =          100.  [ ]                0
K  Param2               = XK(  6) =          200.  [ ]                0
```

Appendix E: ACRC Room Air Conditioner Model User's Reference

A general overview of the ACRC room air conditioner model (RACMOD) is given in Chapter 2. This appendix provides details on the use of RACMOD. Hahn (1993) provides a more detailed description of many of the modeling equations in his Appendix A.

E.1 Configuring the Model

Running the model requires that all input parameters be set to the proper values for the room air conditioner to be simulated. Appendix F defines all parameters and variables used in RACMOD, and Figure E.1 depicts the basic structure of RACMOD, indicating state point locations and some important variables and parameters.

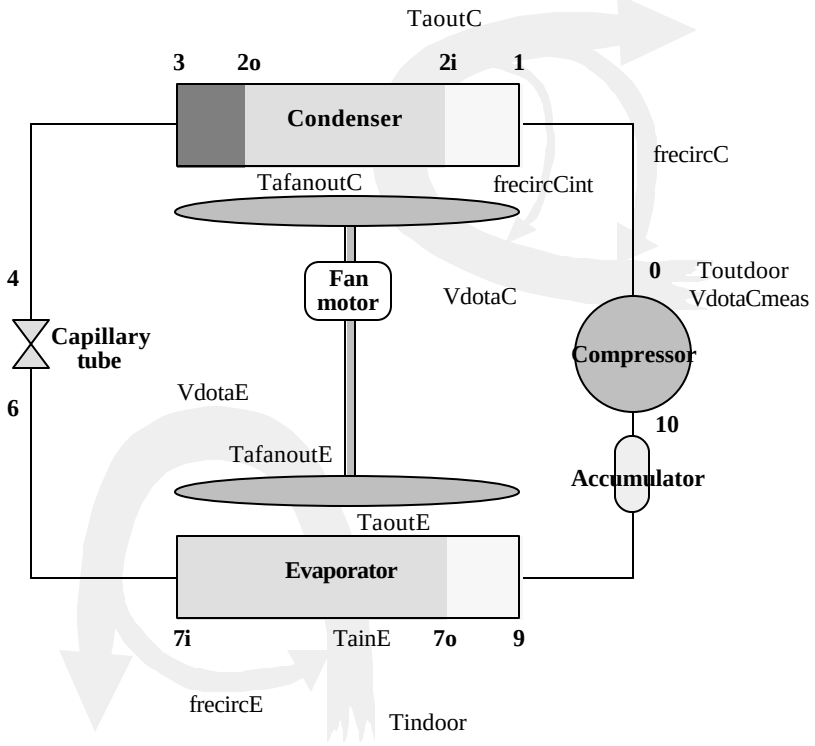

Figure E.1 Schematic of room air conditioner model

E.1.1 The XK Initialization File

The XK initialization file for RACMOD is organized with the variables first, followed by the parameters. The most important variables are grouped at the beginning, and most other variables are grouped alphabetically. The key parameters are also listed first, and all parameters are grouped by component or parameter type.

Some of RACMOD's "parameters" are not parameters in the true sense of the word. The ACRC solver considers everything that is not a Newton-Raphson variable to be a parameter. For example, the pressure drop values and the calculated evaporator moisture removal rate are calculated by the model as intermediate values, but are not needed as NR variables. Such parameters are denoted "pseudo-constants" in Appendix F and are included with the actual parameters only for convenience in reporting their values. Thus attempting to specify the two-phase

condenser pressure drop (pd2phC), for example, through the XK file would be futile–a new pressure drop would be calculated immediately by the model when the following code is executed:

```
460   vvtp = VTX(t2i,1.0d0)
      vltp = VTX(t2o,0.0d0)
      x2i = 1.0
      pd2phC = PD2phACRC(w, PFrC, EqCircuitC, DinC, TubeLenC, NumRtbC,
     &    VTubeDistC, f2phC, t2i, t2o, x2i, x2o, vvtp, vltp,15)
      R(cond+8) = pd2phC - (p2i - p2o)
      Goto 1
```

The code could, however, be changed so that the statement "pd2phC=....." is commented out. Then, after recompilation, the pressure drop value specified in the XK file would be used.

E.1.2 Reconfiguring RACMOD for a New Air Conditioner

The program comes with an XK initialization file representing a solution to RACMOD for a typical 12,000 Btu/hr air conditioner. When setting up an XK initialization file for a different air conditioner, it is recommended that one begin with an existing XK solution file and that the changeover to a new set of input parameters be made in stages, as described in Section E.2, because making drastic changes to many model parameters at once may change the system so severely that the initial guesses for the variables are no longer adequate.

Changing the model configuration also requires replacing the current compressor map and recompiling. The compressor mass flow and power map subroutines--"wmap" and "pwrmap"--can be found in the file EQNSUBS.f, as described in Appendix G.

E.1.3 Capillary Tube Options

Three capillary tube options are available in RACMOD. Either the ACRC finite-difference captube model or the ASHRAE standard curve-fit may be used, or one can choose to have no capillary tube model and instead specify the amount of evaporator superheat. A captube option is selected by appropriately setting the parameter "CapTubeSelect." Setting CapTubeSelect=1 specifies the ACRC captube and CapTubeSelect=2 specifies the ASHRAE captube.

If CapTubeSelect is a negative number, then the capillary tube mass flow equation is replaced with an equation that sets the evaporator superheat equal to the absolute value of CapTubeSelect, effectively switching the equations to design mode. Care must be taken that the variable "degsup" is still flagged with an "X" in the XK file, because it remains a NR variable even though it is effectively set by the user when switching to design mode. See Sections E.5 and E.6 for details on the implementation of the capillary tube models.

E.2 Initial Guesses for Model Variables

The ACRC solver contains features that improve its ability to converge on difficult solutions, but it is not fail-safe. Selecting reasonable initial guesses for variables values is still necessary. Fortunately, RACMOD comes with several solution files in XK file format, and these files can be used as a basis for initial guesses.

The best technique for ensuring adequate initial guesses is to make the required parameter changes in old solution files. If the parameter changes are large enough that the model still will not converge, the transition to the new parameter values can be made in stages, updating the initial guesses with intermediate solution files along the

way. The MUPLTIPLE analysis of the ACRC solver is convenient for systematically changing parameter values while updating the initial guesses along the way.

E.3 Equation Switching

In a room air conditioner, the evaporator exit may be either superheated or two-phase--a single set of governing equations will not properly model both of these conditions. RACMOD contains equations for both a two-phase and a superheated zone of the evaporator, but when it becomes apparent that evaporator exit is actually two-phase, the superheated equations are switched with new equations that, among other things, specify that the area of the superheated zone is zero. The initial and boundary checking routines determine the status of the evaporator exit, and set the flag "Evap2phX" (evaporator two-phase exit) accordingly. Logic statements within the equations determine which equations are executed. RACMOD can thus automatically handle solutions with both two-phase and superheated evaporator exits.

Similarly, the condenser exit and the capillary tube entrance may be subcooled or superheated, and the flags "Cond2phX" and "Cap2phIn" are set by the initial and boundary checking routines to select the appropriate equations.

E.4 Reading the RACMOD Equations

The governing equations for RACMOD are contained in the subroutine CalcR (calculate residual values) in the file EQNS.f and are listed in Appendix ¥. The equations also call subroutines in EQNSUBS.f, various functions in FUNCTION.f, and functions that interface with the NIST property routines in PROPFNCT.f, all of which are listed in Appendix ¥.

Implementation of sparse-matrix Jacobian calculation and equation switching necessitate that some program logic be included in CalcR along with the governing equations. The demonstration model in Appendix D explains this logic with a much simpler model. Explanations are also given in the comments at the beginning of CalcR.

In addition to using logic for equation switching and the use of "NonZeroFlag" for proper building of the NonZeroList for sparse-matrix Jacobian calculation, RACMOD also uses the flags UCDone, UEDone, FWDone, and CTDone. These flags are used by logic statements to ensure that certain computationally intensive subroutines are only executed once per call to CalcR.

E.6 Implementation of Multiple Capillary Tube Models

The various options available for capillary tube modeling make the capillary tube equations one of the most complex parts of CalcR. The equations are discussed here so that they may be better understood, and as an example of some of the logic statements that are included throughout the governing equations.

At the start of CalcR, CapTubeSelect is checked and the flags ACRCcaptube and DesignMode are set.

```
If (CaptubeSelect .le. 1.5) ACRCcaptube = .true.
If (1.5 .lt. CaptubeSelect) ASHRAEcaptube = .true.
If (CaptubeSelect .le. 0.) DesignMode = .true.
```

The ACRC captube uses five residual equations, whereas the ASHRAE captube model and design mode only require two equations. Because the number of equations cannot be changes without modifications to various part of the program and input files, three additional "dummy" equations that assign arbitrary values to xcritcap,

DeltaP, and Pcritcap are used with the ASHRAE model and design mode, in addition to the two required equations. Definitions of all variables and parameters in the governing equations are given in Appendix F.

When in design mode, the equations are:

```
R(captube+0) = degsup - abs(CaptubeSelect)
R(captube+1) = h7i - h4
R(captube+2) = xcritcap - x7i*.9
R(captube+3) = DeltaP - 8.
R(captube+4) = Pcritcap - p7i
```

For the ASHRAE capillary tube model, the equations are:

```
R(captube+0) = wcapCalc - w
R(captube+1) = h7i - h4
R(captube+2) = xcritcap - x7i*.9
R(captube+3) = DeltaP - 8.
R(captube+4) = Pcritcap - p7i
```

where wcapCalc is a Fortran intermediate variable calculated by the subroutine CapTubeASHRAE.

The ACRC capillary tube model is discussed in Section 2.4, although this section does not mention the equation for h7i, because it does not involve the captube subroutine. The equations used for the ACRC captube are:

```
R(captube+0) = wcapCalc - w
R(captube+1) = h7i - hcapout
R(captube+2) = degsubcoolCalc-degsubcool
R(captube+3) = LcapCalc - Lcap
R(captube+4) = XinflashCalc - x4
```

There is a block of code for each of the four capillary tube equations. Each block contains logic that directs which capillary tube model is to be executed. If DesignMode is true, the appropriate equation is executed and control returns to the "on GOTO" statement.

If ACRCcaptube is true, the code checks to see if NonZeroFlag is also true. If it is, a dummy residual containing all of the variables that could ever influence the residual when the captube entrance is subcooled or two-phase. The code checks to see whether CTDone is true before executing the CaptubeACRC subroutine. CTDone indicates whether the subroutine–which returns values for all four functions used in the ACRC captube residuals–has already been executed. Unnecessarily executing the lengthy subroutine would slow the solution process.

If ACRCcaptube is false, the appropriate equations for the ASHRAE captube are executed. There is also a check for NonZeroFlag to ensure that the Jacobian will be correct for cases with both subcooled and two-phase captube entrances.

The portion of CalcR which contains the capillary tube equations is listed below.

```
C ***** CAPILLARY TUBE EQUATIONS *****
1260    If (DesignMode) Then
            R(captube+0) = degsup - abs(CaptubeSelect)
            Goto 1
        EndIf
        If (ACRCcaptube) Then
            If (NonZeroFlag) Then
                R(captube+0) = p4 + Dcap + Pcritcap + DeltaP + xcritcap
     &            + degsubcool + Lcap + x4 + wcapCalc + CT1 +
```

```
     &            CT2 + w + h7i + p2o
                Goto 1
            EndIf
            If (.not.CTDone) Call CaptubeACRC(Cap2phIn, p2o, p4,
     &        Dcap, Pcritcap, DeltaP, xcritcap, x4, degsubcoolCalc,
     &        LcapCalc,wcapCalc,XinflashCalc, hcapout)
            wcapCalc = wcapCalc * Numcaps
            CTDone = .true.
         Else
            If (NonZeroFlag) Then
                R(captube+0) = p4 + Dcap + Lcap + wcapCalc +CT1+
     &       degsubcool + CT2 + w + p7i + x4 + NumCaps + t2o + t4
                Goto 1
            EndIf
            Call CapTubeASHRAE(Lcap, Dcap, Numcaps, Cap2phIn, t2o,
     &        t4, x4, p4, p7i, wcapCalc)
         EndIf
         R(captube+0) = wcapCalc - w
C        Capillary tube massflow correction based on subcooling
         Goto 1

1280     R(captube+1) = h7i - h4
         If (ACRCcaptube) Then
            If (NonZeroFlag) Then
                R(captube+1) = p4 + Dcap + Pcritcap + DeltaP + xcritcap
     &            + degsubcool + Lcap + x4 + wcapCalc + CT1 +
     &            CT2 + w + h7i + p2o
                Goto 1
            EndIf
            If (.not.CTDone) Call CaptubeACRC(Cap2phIn, p2o, p4,
     &        Dcap, Pcritcap, DeltaP, xcritcap, x4, degsubcoolCalc,
     &        LcapCalc,wcapCalc,XinflashCalc, hcapout)
            CTDone = .true.
         R(captube+1) = h7i - hcapout
         EndIf
         Goto 1

1285     R(captube+2) = xcritcap - x7i*.9
         If (ACRCcaptube) Then
            If (NonZeroFlag) Then
                R(captube+2) = p4 + Dcap + Pcritcap + DeltaP + xcritcap
     &            + degsubcool + Lcap + x4 + wcapCalc + CT1 +
     &            CT2 + w + h7i + p2o
                Goto 1
            EndIf
            If (.not.CTDone) Call CaptubeACRC(Cap2phIn, p2o, p4,
     &        Dcap, Pcritcap, DeltaP, xcritcap, x4, degsubcoolCalc,
     &        LcapCalc,wcapCalc,XinflashCalc, hcapout)
            CTDone = .true.
         R(captube+2) = degsubcoolCalc-degsubcool
         EndIf
         Goto 1

1290     R(captube+3) = DeltaP - 8.
```

```
         If (ACRCcaptube) Then
             If (NonZeroFlag) Then
                 R(captube+3) = p4 + Dcap + Pcritcap + DeltaP + xcritcap
     &             + degsubcool + Lcap + x4 + wcapCalc + CT1 +
     &             CT2 + w + h7i + p2o
                 Goto 1
             EndIf
             If (.not.CTDone) Call CaptubeACRC(Cap2phIn, p2o, p4,
     &         Dcap, Pcritcap, DeltaP, xcritcap, x4, degsubcoolCalc,
     &         LcapCalc,wcapCalc,XinflashCalc, hcapout)
             CTDone = .true.
         R(captube+3) = LcapCalc - Lcap
         EndIf
         Goto 1

1295     R(captube+4) = Pcritcap - p7i
         If (ACRCcaptube) Then
             If (NonZeroFlag) Then
                 R(captube+4) = p4 + Dcap + Pcritcap + DeltaP + xcritcap
     &             + degsubcool + Lcap + x4 + wcapCalc + CT1 +
     &             CT2 + w + h7i + p2o
                 Goto 1
             EndIf
             If (.not.CTDone) Call CaptubeACRC(Cap2phIn, p2o, p4,
     &         Dcap, Pcritcap, DeltaP, xcritcap, x4, degsubcoolCalc,
     &         LcapCalc,wcapCalc,XinflashCalc, hcapout)
            CTDone = .true.
         R(captube+4) = XinflashCalc - x4
         EndIf
         Goto 1
```

References

Hahn, Gregory W. “Modeling Room Air Conditioner Performance.” University of Illinois at Urbana-Champaign, ACRC TR-40, 1993.

Appendix F: RACMOD Variable and Parameter Definitions

Tables F.1 and F.2 contain descriptions of all variables and parameters in the governing equations of RACMOD, based on a typical simulation mode configuration. The XK initialization file for RACMOD is organized with the variables first, followed by the parameters. The most important variables are grouped at the beginning, and most other variables are grouped alphabetically. The key parameters are also listed first, and all parameters are grouped by component or parameter type. Of course, changing the 'X' and 'K' flags for variable-parameter swapping may cause the variables and parameters to become intermingled. The variables and parameters are listed in the same order in which they appear in the XK initialization file.

Table F.1 Definition of variables used in the room air conditioner model

XK name	Definition	Typical Values	Units	Com-ponent
COP	coefficient of performance for the room air conditioner	2.65	[-]	All
EER	energy efficiency ratio for the room air conditioner	9.05	[Btu/W]	All
qEvap	total heat transfer rate of the evaporator	11605	[Btu/hr]	Evap
degsubcool	degrees of subcooling at the condenser exit	12.1	[°F]	Cond
degsup	degrees superheat at the end of the superheated zone of the evaporator	16.8	[°F]	Evap
SLsuperheat	degrees of superheat at the end of the suction line	27.6	[F]	Comp
tsatincomp	saturation temperature of the refrigerant entering the compressor	43.5	[°F]	Comp
tsatoutcomp	saturation temperature of the refrigerant exiting the compressor	125.9	[°F]	Cond
w	refrigerant mass flow rate throughout the room air conditioner	168.3	[lbm/hr]	All
PwrComp	electrical power used by the compressor	3530	[Btu/hr]	Comp
qComp	heat transfer rate of the compressor to the condenser inlet air stream	1450	[Btu/hr]	Comp
RejectRatio	ratio of the heat rejected by the compressor to its power consumption	0.411	[-]	Comp
Acond	total refrigerant-side area of the condenser	4.349	[ft^2]	Cond
Aevap	total refrigerant-side area of the evaporator	2.871	[ft^2]	Evap
asupC	refrigerant-side area of the superheated zone of the condenser	0.453	[ft^2]	Cond
a2phC	refrigerant-side area of the two-phase zone of the condenser	3.534	[ft^2]	Cond
asubC	refrigerant-side area of the subcooled zone of the condenser	0.361	[ft^2]	Cond
a2phE	refrigerant-side area of the two-phase zone of the evaporator	2.589	[ft^2]	Evap
asupE	refrigerant-side area of the superheated zone of the evaporator	0.281	[ft^2]	Evap
caC	air mass flow * Cp for the entire condenser airflow	698.5	[Btu/hr-°F]	Cond
caE	air mass flow * Cp for the entire evaporator airflow	380.4	[Btu/hr-°F]	Evap
casupC	air mass flow * Cp for the superheated zone of the condenser	72.8	[Btu/hr-°F]	Cond
ca2phC	air mass flow * Cp for the two-phase zone of the condenser	567.7	[Btu/hr-°F]	Cond
ca2phE	air mass flow * Cp for the two-phase zone of the evaporator	343.1	[Btu/hr-°F]	Evap

XK name	Definition	Typical Values	Units	Com-ponent
casupE	air mass flow * Cp for the superheated zone of the evaporator	37.3	[Btu/hr-°F]	Evap
crsubC	refrigerant mass flow * Cp for the subcooled zone of the condenser	50.6	[Btu/hr-°F]	Cond
crsupC	refrigerant mass flow * Cp for the superheated zone of the condenser	37.8	[Btu/hr-°F]	Cond
crsupE	refrigerant mass flow * Cp for the superheated zone of the evaporator	31.5	[Btu/hr-°F]	Evap
cmaxsubC	maximum heat capacity of the refrigerant and air streams for the subcooled zone of the condenser	58	[Btu/hr-°F]	Cond
cmaxsupC	maximum heat capacity of the refrigerant and air streams for the superheated zone of the condenser	72.8	[Btu/hr-°F]	Cond
cmaxsupE	maximum heat capacity of the refrigerant and air streams for the superheated zone of the evaporator	37.3	[Btu/hr-°F]	Evap
cminsubC	minimum heat capacity of the refrigerant and air streams for the subcooled zone of the condenser	50.6	[Btu/hr-°F]	Cond
cminsupC	minimum heat capacity of the refrigerant and air streams for the superheated zone of the condenser	37.8	[Btu/hr-°F]	Cond
cminsupE	minimum heat capacity of the refrigerant and air streams for the superheated zone of the evaporator	31.505	[Btu/hr-°F]	Evap
esupC	heat exchanger effectiveness of the superheated zone of the condenser	0.653	[-]	Cond
e2phC	heat exchanger effectiveness of the two-phase zone of the condenser	0.703	[-]	Cond
esubC	heat exchanger effectiveness of the subcooled zone of the condenser	0.456	[-]	Cond
e2phE	heat exchanger effectiveness of the two-phase zone of the evaporator, pseudo-constant	0.936	[-]	Evap
esupE	heat exchanger effectiveness of the superheated zone of the evaporator	0.419	[-]	Evap
fsupC	fraction of the condenser which is superheated, on an area basis	0.104	[-]	Cond
f2phC	fraction of the condenser which is two-phase, on an area basis	0.813	[-]	Cond
fsubC	fraction of the condenser which is subcooled, on an area basis	0.083	[-]	Cond
f2phE	fraction of the evaporator which is two-phase, on an area basis	0.902	[-]	Evap
fsupE	fraction of the evaporator which is superheated, on an area basis	0.098	[-]	Evap
GrC	mass flux in one equivalent circuit of the condenser	281717	[lbm/(hr-ft^2)]	Cond
GrE	mass flux in one equivalent circuit of the evaporator	156509	[lbm/(hr-ft^2)]	Evap
h0	enthalpy of the refrigerant exiting the compressor	126.1	[Btu/lbm]	Comp
h1	enthalpy of the refrigerant entering the superheated zone of the condenser	125.2	[Btu/lbm]	Cond
h2i	enthalpy of the refrigerant entering the two-phase zone of the condenser (saturated vapor)	113.4	[Btu/lbm]	Cond
h2o	enthalpy of the refrigerant exiting the two-phase zone of the condenser (saturated liquid)	47	[Btu/lbm]	Cond

XK name	Definition	Typical Values	Units	Com-ponent
h3	enthalpy of the refrigerant exiting the subcooled zone of the condenser	43.3	[Btu/lbm]	Cond
h4	enthalpy of the refrigerant entering the capillary tube	43	[Btu/lbm]	Captube
h7i	enthalpy of the refrigerant entering to two-phase zone of the evaporator	43	[Btu/lbm]	Evap
h7o	enthalpy of the refrigerant exiting the two-phase zone of the evaporator	109.2	[Btu/lbm]	Evap
h9	enthalpy of the refrigerant exiting the superheated zone of the evaporator	112	[Btu/lbm]	Evap
h10	enthalpy of the refrigerant entering the compressor	113.8	[Btu/lbm]	Comp
MtotC	mass of refrigerant in the entire condenser	0.82	[lbm]	Charge
MtotE	mass of refrigerant in the entire evaporator	0.529	[lbm]	Charge
MsupC	mass of refrigerant in the superheated zone of the condenser	0.027	[lbm]	Charge
M2phC	mass of refrigerant in the 2ph zone of the condenser	0.611	[lbm]	Charge
MsubC	mass of refrigerant in the subcooled zone of the condenser	0.183	[lbm]	Charge
M2phE	mass of refrigerant in the 2ph zone of the evaporator	0.525	[lbm]	Charge
MsupE	mass of refrigerant in the superheated zone of the evaporator	0.004	[lbm]	Charge
Maccum	mass of refrigerant in the accumulator	0.019	[lbm]	Charge
Mcaptube	mass of refrigerant in the capillary tube	0.003	[lbm]	Charge
Mcomp	mass of refrigerant in the compressor	0.002	[lbm]	Charge
MdisLine	mass of refrigerant in the discharge line	0.008	[lbm]	Charge
MEheader	mass of refrigerant in the evaporator header	0.007	[lbm]	Charge
MliqLine	mass of refrigerant in the liquid line	0.038	[lbm]	Charge
MsuctLine	mass of refrigerant in the suction line	0.009	[lbm]	Charge
muf2	viscosity of liquid refrigerant at state point 2	0.414	[lbm/ft-hr]	Charge
muf7	viscosity of liquid refrigerant at state point 7	0.544	[lbm/ft-hr]	Charge
mug2	viscosity of gaseous refrigerant at state point 2	0.037	[lbm/ft-hr]	Charge
mug7	viscosity of gaseous refrigerant at state point 7	0.03	[lbm/ft-hr]	Charge
p0	pressure of the refrigerant exiting the compressor	297.6	[psia]	Comp
p1	pressure of the refrigerant entering the superheated zone of the condenser	297	[psia]	Cond
p2i	pressure of the refrigerant entering the two-phase zone of the condenser	296.5	[psia]	Cond
p2avg	average saturation pressure of the two-phase zone of the condenser	293.6	[psia]	Cond
p2o	pressure of the refrigerant exiting the two-phase zone of the condenser	290.8	[psia]	Cond
p3	pressure of the refrigerant exiting the subcooled zone of the condenser	290.7	[psia]	Cond
p4	pressure of the refrigerant entering the capillary tube	290.7	[psia]	Captube
p7i	pressure of the refrigerant entering the evaporator	91.6	[psia]	Evap
p7avg	average saturation pressure of the two-phase zone of the evaporator	90.3	[psia]	Evap
p7o	pressure of the refrigerant exiting the two-phase zone of the evaporator	89	[psia]	Evap

XK name	Definition	Typical Values	Units	Com-ponent
p9	pressure of the refrigerant exiting the superheated zone of the evaporator	88.8	[psia]	Evap
p10	pressure of the refrigerant entering the compressor	88.6	[psia]	Comp
qCond	total heat transfer rate of the condenser	13784	[Btu/hr]	Cond
qspray	latent heat transfer associated with the condensate spray on the condenser	0	[Btu/hr]	Cond
qsupC	heat transfer rate of the superheated zone of the condenser	1998	[Btu/hr]	Cond
q2phC	heat transfer rate of the two-phase zone of the condenser	11175	[Btu/hr]	Cond
qsubC	heat transfer rate of the subcooled zone of the condenser	611	[Btu/hr]	Cond
q2phE	heat transfer rate of the two-phase zone of the evaporator	11127	[Btu/hr]	Evap
qsupE	heat transfer rate of the superheated zone of the evaporator	478	[Btu/hr]	Evap
qgainAccum	heat transfer rate in the accumulator required to ensure a quality of 1 at the compressor inlet	0	[Btu/hr]	Comp
rhof2	density of liquid refrigerant at state point 2	67	[lbm/ft^3]	Charge
rhof3	density of liquid refrigerant at state point 3	68.2	[lbm/ft^3]	Charge
rhof5	density of liquid refrigerant at state point 5	74.2	[lbm/ft^3]	Charge
rhof7	density of liquid refrigerant at state point 7	78.5	[lbm/ft^3]	Charge
rhog1	density of gaseous refrigerant at state point 1	8.1	[lbm/ft^3]	Charge
rhog2	density of gaseous refrigerant at state point 2	5.5	[lbm/ft^3]	Charge
rhog5	density of gaseous refrigerant at state point 5	2.8	[lbm/ft^3]	Charge
rhog7	density of gaseous refrigerant at state point 7	1.6	[lbm/ft^3]	Charge
rho7avg	average density of two-phase refrigerant at state point 7	6.7	[lbm/ft^3]	Charge
rhog9	density of gaseous refrigerant at state point 9	1.9	[lbm/ft^3]	Charge
rhog10	density of gaseous refrigerant at state point 10	2.1	[lbm/ft^3]	Charge
t0	refrigerant temperature exiting the compressor	183.1	[°F]	Comp
t1	refrigerant temperature entering the condenser	178.8	[°F]	Cond
t2i	refrigerant temperature entering the two-phase zone of the condenser	125.9	[°F]	Cond
t2avg	average saturation temperature of the two-phase zone of the condenser	125.2	[°F]	Cond
t2o	refrigerant temperature exiting the two-phase zone of the condenser	124.4	[°F]	Cond
t3	refrigerant temperature exiting the condenser	112.3	[°F]	Cond
t4	refrigerant temperature entering the capillary tube	111.4	[°F]	Captube
t7i	refrigerant temperature entering the evaporator	45.5	[°F]	Evap
t7avg	average saturation temperature in the two-phase zone of the evaporator	44.7	[°F]	Evap
t7o	refrigerant temperature exiting the two-phase zone of the evaporator	43.8	[°F]	Evap
t9	refrigerant temperature exiting the evaporator	60.6	[°F]	Evap
t10	refrigerant temperature entering the compressor	71.1	[°F]	Comp
tafanoutC	temperature of air leaving the condenser fan, immediately before entering the condenser	97.9	[F]	Cond
tasupoutC	air temperature exiting the superheated zone of the condenser	125.4	[°F]	Cond
ta2phoutC	air temperature exiting the two-phase zone of the condenser	117.6	[°F]	Cond
tasuboutC	air temperature exiting the subcooled zone of the condenser	108.4	[°F]	Cond

XK name	Definition	Typical Values	Units	Com-ponent
taoutC	air temperature exiting the condenser, assuming all three zone air streams are mixed	117.6	[°F]	Cond
TainE	air temperature entering the evaporator	70.1	[°F]	Evap
ta2phoutE	air temperature exiting the two-phase zone of the evaporator	47.7	[°F]	Evap
tasupoutE	air temperature exiting the superheated zone of the evaporator	67.2	[°F]	Evap
taoutE	air temperature exiting the evaporator, assuming that both zone air streams are mixed	49.6	[°F]	Evap
tafanoutE	air temperature exiting the evaporator fan and entering the room	50.1	[°F]	Evap
usupC	overall heat transfer coefficient in the superheated zone of the condenser	123.2	[Btu/hr-ft^2-°F]	Cond
u2phC	overall heat transfer coefficient in the two-phase zone of the condenser	194.7	[Btu/hr-ft^2-°F]	Cond
usubC	overall heat transfer coefficient in the subcooled zone of the condenser	123.3	[Btu/hr-ft^2-°F]	Cond
u2phE	overall heat transfer coefficient in the two-phase zone of the evaporator	299.2	[Btu/hr-ft^2-°F]	Evap
usupE	overall heat transfer coefficient in the superheated zone of the evaporator	83.6	[Btu/hr-ft^2-°F]	Evap
vdotaC	actual volume air flow rate passing through the condenser, with recirculation	700	[cfm]	Condc
mdotaC	mass flow rate of air passing through the condenser	2910	[lbm/hr]	Cond
mdotaE	mass flow rate of air passing through the evaporator	1585	[lbm/hr]	Evap
TubeLenC	length of tubing in condenser per equivalent circuit	50.2	[ft]	Cond
TubeLenE	length of tubing in evaporator per equivalent circuit	18.4	[ft]	Evap
Vcond	volume of the condenser, excluding return bends	0.032	[ft^3]	Charge
Vevap	volume of the evaporator, excluding return bends	0.021	[ft^3]	Charge
VRtrnBndC	volume of a condenser return bend	0	[ft^3]	Charge
VRtrnBndE	volume of a evaporator return bend	0	[ft^3]	Charge
Vcap	volume of the capillary tube	0	[ft^3]	Charge
Vdisc	volume of the discharge line	0.001	[ft^3]	Charge
VliqLine	volume of the liquid line	0.001	[ft^3]	Charge
WgC	heat-flux-averaged void fraction in the condenser	0.711	[-]	Charge
WgE	heat-flux-averaged void fraction in the evaporator	0.668	[-]	Charge
Wgheader	heat-flux-averaged void fraction in the evaporator header	0.934	[-]	Charge
x2o	quality at the end of the 2ph zone of the condenser	0	[-]	Cond
x4	quality at state point 4, the capillary tube entrance	0	[-]	Captube
x7i	quality at the evaporator entrance	0.229	[-]	Evap
x7o	quality at the end of the 2ph zone of the evaporator	1	[-]	Evap
DeltaP	pressure step taken in the two-phase portion of the capillary tube (ACRC captube model only)	8	[psia]	Captube
Pcritcap	critical pressure in the exit section of the capillary tube (ACRC captube model only)	91.6	[psia]	Captube
xcritcap	critical quality in the exit section of the capillary tube (ACRC captube model only)	0.206	[-]	Captube
mH2Ormvd	mass of water removed from the air by the evaporator, pseudo-constant	0	[lbm]	Evap

XK name	Definition	Typical Values	Units	Com-ponent
q2phtotalE	heat transfer rate of the two-phase zone of the evaporator if the entire evaporator was two-phase, multiplied by f2phE to get q2phE	12336	[Btu/hr]	Evap
tftmout	temperature of the trailing edge of the evaporator coil in the event that the coil is fully, or partially wet	46.4	[°F]	Evap
twalli	wall temperature of the leading edge of the evaporator coil in the event that the coil is fully wet	46.9	[°F]	Evap
wairo	humidity ratio of the air leaving the evaporator coil	0.002	[lbmw/lbmda]	Evap

All of the "pseudo-constants" in the parameters are grouped at the end. The term "pseudo-constant" is defined in Section E.1.1 and refers to intermediate variables in the equations that are not needed as NR variables. Pseudo-constants are included with the parameters only for convenience in reporting their values. Specifying the value of a pseudo-constant in the XK file will have no effect and pseudo-constants may not be involved in variable-parameter swapping.

Table F.2 Definition of parameters used in the room air conditioner model

XK name	Definition	Typical Values	Units	Com-ponent
Toutdoor	temperature of the outdoor room	95	[F]	Cond
Tindoor	temperature of the indoor room	85	[F]	Evap
RhaiC	relative humidity of the air entering the condenser	0.42	[-]	Cond
RhaiE	relative humidity of the air entering the evaporator	0.1	[-]	Evap
vdotaCmeas	measured volume flow rate of air passing through the condenser, assuming no recirculation	700	[cfm]	Cond
vdotaE	actual volume air flow rate passing through the evaporator, with recirculation	350	[cfm]	Evap
frecircC	fraction of condenser outlet air that is recirculated to the its inlet grille	0	[-]	Cond
frecircCint	fraction of condenser outlet air that is recirculated back into its outlet grille, "internal" recirculation	0	[-]	Cond
frecircE	fraction of evaporator outlet air that is recirculated to the its inlet grille	0	[-]	Evap
Mtotal	total refrigerant charge	1.4335	[lbm]	Charge
patm	atmospheric pressure	14.24	[psia]	-
TcompRetGas	temperature of the suction gas for which the compressor map was generated	95	[°F]	Comp
UAcomp	area averaged heat transfer coefficient for the compressor shell	16.47	[Btu/hr-°F]	Comp
wmapCor	multiplicative correction factor for the compressor mass flow map	1	[-]	Comp
pwrmapCor	multiplicative correction factor for the compressor power map	1	[-]	Comp
SprayC1	a constant for condensate spray heat transfer enhancement calculation	0	[-]	Cond
SprayhmA	(not used in current model)	0	[lbm/hr]	Cond

XK name	Definition	Typical Values	Units	Com-ponent
CapTubeSelect	selects captube 1=ACRC, 2=ASHRAE, negative #= design mode, i.e. no captube, and degsup=abs(CapTubeSelect)	2	[-]	Captube
CT1	unimplemented, possible captube corrective factor	1	[-]	Captube
CT2	unimplemented, possible captube corrective factor	0	[1/F]	Captube
NumCaps	number of parallel capillary tubes in the unit	1	[-]	Captube
Lcap	length of the capillary tube	38.2	[in]	Captube
Dcap	inside diameter of the capillary tube	0.064	[ft]	Captube
Dsuct	inside diameter of the suction line	0.038	[ft]	Suclin
Ddisc	inside diameter of the discharge line	0.02242	[ft]	DisLin
Dliq	inside diameter of the liquid line	0.02758	[ft]	Liqlin
DinC	inside diameter of the condenser tubing	0.02758	[ft]	Cond
DinE	inside diameter of the evaporator tubing	0.02758	[ft]	Evap
DoutC	outside diameter of the tubing of the condenser	0.03125	[ft]	Cond
DoutE	outside diameter of the tubing of the evaporator	0.03125	[ft]	Evap
EFaC	enhancement factor for condenser air-side heat transfer coefficient	1	[-]	Cond
EFrC	enhancement factor for condenser refrigerant-side heat transfer coeff.	1.66	[-]	Cond
EFaE	enhancement factor for evaporator air-side heat transfer coefficient	2	[-]	Evap
EFrE	enhancement factor for evaporator refrigerant-side heat transfer coeff.	1.66	[-]	Evap
LeqDL	discharge line equivalent length (for pressure drop)	2.43	[ft]	Disclin
LeqLL	liquid line equivalent length (for pressure drop)	0.93	[ft]	Liqlin
LeqSL	suction line equivalent length (for pressure drop)	3.65	[ft]	Suclin
PFrC	refrigerant-side pressure drop penalty factor for the condenser	1	[-]	Cond
PFrE	refrigerant-side pressure drop penalty factor for the condenser	1	[-]	Evap
pwrfanC	fan power added to the air stream before the condenser	580	[Btu/hr]	Cond
pwrfanE	fan power added to the air stream after the evaporator	194	[Btu/hr]	Evap
qgainSL	specified suction line heat gain	300	[Btu/hr]	Suclin
qlossDL	specified discharge line heat loss	150	[Btu/hr]	Disclin
qlossLL	specified liquid line heat loss	50	[Btu/hr]	Liqlin
AfrontC	Frontal area of crossflow condenser	2.083	[ft^2]	Cond
AfrontE	Frontal area of crossflow evaporator	1.375	[ft^2]	Evap
EQcircuitC	number of equivalent circuits in the condenser	1	[-]	Cond
EQcircuitE	number of equivalent circuits in the evaporator	1.8	[-]	Evap
TubeRowsC	number of tubes in the airflow direction of the condenser	2	[-]	Cond
TubeRowsE	number of tubes in the airflow direction of the evaporator	2	[-]	Evap
HtubeDistC	horizontal tube spacing in the airflow direction of the condenser	0.07217	[ft]	Cond
HtubeDistE	horizontal tube spacing in the airflow direction of the evaporator	0.07217	[ft]	Evap
VtubeDistC	vertical tube spacing in the condenser	0.083	[ft]	Cond
VtubeDistE	vertical tube spacing in the evaporator	0.083	[ft]	Evap
LRtrnBndC	length of a condenser return bend	0.13	[ft]	Cond

XK name	Definition	Typical Values	Units	Com-ponent
LRtrnBndE	length of an evaporator return bend	0.13	[ft]	Evap
NumRtbC	number of return bends in the condenser	29	[-]	Cond
NumRtbE	number of return bends in the evaporator	20.5	[-]	Evap
FinPtchC	fin pitch of the fins on the condenser	192	[fins/ft]	Cond
FinPtchE	fin pitch of the fins on the evaporator	192	[fins/ft]	Evap
FinThC	thickness of the condenser fins	0.00035	[ft]	Cond
FinThE	thickness of the evaporator fins	0.00035	[ft]	Evap
rough	tubing roughness for calculating pressure drop in various components	0.000005	[in]	All
KfinC	thermal conductivity of the fins in the condenser	128.3	[Btu/hr-ft-°F]	Cond
KtubeC	thermal conductivity of the tubing in the condenser	196	[Btu/hr-ft-°F]	Cond
KfinE	thermal conductivity of the fins in the evaporator	128.3	[Btu/hr-ft-°F]	Evap
KtubeE	thermal conductivity of the tubing in the evaporator	196	[Btu/hr-ft-°F]	Evap
Vaccum	volume of the accumulator	0.0024	[ft^3]	Charge
Vcomp	volume of the compressor	0.0405	[ft^3]	Charge
VheaderE	volume of the evaporator header	0.001	[ft^3]	Charge
Wgcaptube	void fraction in the capillary tube	0.5	[-]	Charge
Icomp	irreversibility generated in the compressor, pseudo-constant	1180	[Btu/hr]	Irrev
Icond	irreversibility generated in the condenser, pseudo-constant	1316	[Btu/hr]	Irrev
Ievap	irreversibility generated in the evaporator, pseudo-constant	1099	[Btu/hr]	Irrev
Ipipes	irreversibility generated in connecting lines and capillary tube, pseudo-constant	386	[Btu/hr]	Irrev
Itot	total irreversibility generated in the system, pseudo-constant	3981	[Btu/hr]	Irrev
pdsupC	pressure drop in the superheated zone of the condenser, pseudo-constant	0.5	[psia]	Cond
pd2phC	pressure drop in the two-phase zone of the condenser, pseudo-constant	5.68	[psia]	Cond
pdsubC	pressure drop in the subcooled zone of the condenser, pseudo-constant	0.05	[psia]	Cond
pd2phE	pressure drop in the two-phase zone of the evaporator, pseudo-constant	2.61	[psia]	Evap
pdsupE	pressure drop in the superheated zone of the evaporator, pseudo-constant	0.21	[psia]	Evap
pdsuctline	pressure drop in the suction line, pseudo-constant	0.2	[psia]	Suclin
pddisline	pressure drop in the discharge line, pseudo-constant	0.6	[psia]	Disclin
pdliqline	pressure drop in the liquid line, pseudo-constant	0.01	[psia]	Liqlin
fsensible	fraction of the 2ph heat transfer in the evaporator that is sensible, pseudo-constant	1	[-]	Evap
TACI	bulk air temperature in evaporator at which water begins to condense, pseudo-constant	-3.7	[F]	Evap
wwallo	humidity ratio of air at wall leaving evaporator coil, pseudo-constant	0.002	[lbm H2O/lbm air]	Evap
Kcntact	contact conductance, pseudo-constant	85913	[hr-ft-°F/Btu]	Evap

Appendix G: RACMOD Subroutine and Function Descriptions

RACMOD is organized into several files which are compiled using a "make" file that automatically compiles and links the subroutines in the various files along with the subroutines and files from the ACRC solver. The use of "include" files within various subroutines is discussed in Section B.2.1.

Table G.1 RACMOD file structure and subroutine descriptions

File or Subroutine Name	Description
INITMOD.f	This file contains subroutines used in the initialization of RACMOD
InitializeModel	Reads XK initialization file, generates Xmap, asks if NonZeroList should be rebuilt and calls CreateNonZeroList or ReadNonZeroList accordingly
CreateNonZeroList	Calculates the Jacobian, builds the NonZeroList, and saves it to a file
ReadNonZeroList	Reads the NonZeroList from the file "NONZEROL.IST"
EQNS.f	This file contains CalcR for RACMOD
CalcR	Calculates residual values of the governing equations using sparse-matrix Gaussian elimination
EQNSUBS.f	This file contains several subroutines that are used by the governing equations of RACMOD
CaptubeAshrae	Calculates mass flow through the capillary tube based on the ASHRAE curve-fit model
CaptubeACRC	This subroutine is used to solve the ACRC finite difference capillary tube model (See Section ?.5)
FindWet	This subroutine is used to solve the evaporator dehumidification analysis, based on the analysis in the ORNL code
Pwrmap	Compressor power map
Wmap	Compressor mass flow map
Wact	Calculates compressor mass flow based on Wmap and the suction line superheat correction from the ORNL code
Pwract	Calculates compressor power based on Pwrmap and the suction line superheat correction from the ORNL code
sfunct	function for finding temperature as a function of entropy and pressure, equals zero when the temperature has been found. Used by Pwract
zero	A single variable Newton-Raphson solver used by Pwract
KH	Returns parameter used by Hughmark void fraction correlation in GLIntegrate
GLIntegrate	Returns the refrigerant gas density weighting factor (Wg) using Hughmark void fraction correlation and Gaussian-Legendre quadrature integration
FUNCTION.f	This file contains numerous functions used by the governing equations of RACMOD
pi	Returns the value for p
FK	Converts Fahrenheit to Kelvin
iff	A function that replicates a one-line if-then structure found in EES
e2p	Returns the effectiveness of a two-phase heat exchanger
ecrossflow	Returns cross flow heat exchanger effectiveness, taken from Incropera and DeWitt
ha	Returns the enthalpy of air given the temperature (Btu/lbm)
va	Returns the specific volume of air given the temperature and pressure (ft^3/lbm)
cpa	Returns the specific heat of air given the temperature (Btu/(lbm-R))
mua	Returns the viscosity of air given the temperature (lbm/(ft-hr))

File or Subroutine Name	Description
ka	Returns the thermal conductivity of air given the temperature (Btu/(hr-ft-F))
ReAir	Returns the Reynolds number of air given the temperature, mass flux, and diameter
PrAir	Returns the Prandtl number of air given the temperature given the temperature
Cpl	Returns the specific heat of saturated liquid refrigerant given the temperature (Btu/(lbm*R))
Cpv	Returns the specific heat of saturated vapor refrigerant given the temperature (Btu/(lbm-R))
kl	Returns the thermal conductivity of saturated liquid refrigerant given the temperature (Btu/(hr-ft-F))
kv	Returns the thermal conductivity of saturated vapor refrigerant given the temperature (Btu/(hr-ft-F))
mul	Returns the viscosity of saturated liquid refrigerant given the temperature (lbm/hr-ft)
muv	Returns the viscosity of saturated vapor refrigerant given the temperature (lbm/hr-ft)
Rel	Returns the Reynolds number of saturated liquid refrigerant given the temperature, mass flux, and diameter
Rev	Returns the Reynolds number of saturated vapor refrigerant given the temperature, mass flux, and diameter
Prl	Returns the Prandtl number of saturated liquid refrigerant given the temperature given the temperature
Prv	Returns the Prandtl number of saturated vapor refrigerant given the temperature given the temperature
hdbw	Returns the enthalpy of moist air given dry-bulb temperature and humidity ratio (Btu/lbm)
Humrat	Returns the humidity ratio given temperature, pressure, and relative humidity.
Tdp	Returns the dew-point temperature of moist air given temperature, humidity ratio, and pressure (F)
PVSF	Returns the partial pressure of water vapor in saturated air (psi)
VolMoist	Returns the specific volume of moist air given temperature, pressure, and relative humidity
UsCond	This subroutine calculates the superheated, two-phase, and subcooled overall heat transfer coefficients for the condenser, based on geometric parameters specified in the input file
UsEvap	This subroutine calculates the superheated and two-phase overall heat transfer coefficients for the evaporator, based on geometric parameters specified in the input file
hliq	Returns the refrigerant side liquid heat transfer coefficient (using Dittus Boelter correlation)
hvap	Returns the refrigerant side vapor heat transfer coefficient (using Dittus Boelter correlation)
SPHTC	This subroutine determine singles phase heat transfer coefficients and reynolds numbers in laminar, transition, or turbulent gas flow from an abrupt contraction entrance (from ORNL code)
h2phcondACRC	Returns the two-phase heat transfer coefficient for the condenser (based on correlation by Chato/Dobson at ACRC)
h2phevapACRC	Returns the two-phase heat transfer coefficient for the evaporator (based on correlation by Chato/Wattelet at ACRC)
h2phcond	Returns the two-phase condensation heat transfer coefficient (from the ORNL code)
CHTC	This subroutine determines the forced convection condensation two-phase heat transfer coefficient for flow in tubes (from ORNL code)

File or Subroutine Name	Description
h2phevap	Returns the two-phase evaporation heat transfer coefficient, (from the ORNL code)
SurfEff	Returns surface effectiveness for given geometry, (from ORNL code)
hair	Returns air side heat transfer coefficient in (Btu/hr-ft^2-F) for a fin and tube heat exchanger, (from the ORNL code)
ERROR	This subroutine is used by CHTC
pd2phACRC	Returns the pressure drop for a two-phase zone of a heat exchanger (developed by ACRC)
Xtt	Returns the Lockhart-Martinelli parameter
dptpHX	Returns the pressure drop for a two-phase zone of a heat exchanger (taken from ORNL code)
Z	Returns the frictional multiplier for calculating pressure drop (based on data taken from ORNL code)
ZZ	Returns the accelerational multiplier for calculating pressure drop (based on data taken from ORNL code)
dpspHX	Returns the pressure drop for a single-phase zone of a heat exchanger (taken from ORNL code)
dpsuction	Returns the pressure drop for the suction line (taken from ORNL code)
Moody	Returns the Moody friction factor (explicit form of correlation taken from ORNL code)
CHECKMOD.f	This file contains the subroutines used for initial, boundary, and final checking for RACMOD
IC	Checks for two-phase condenser and evaporator exits and capillary tube entrance and sets the flags Cond2phX, Evap2phX, and Cap2phIn accordingly
BC	Checks to see if the phase status at the condenser and evaporator exits and capillary tube entrance have changed and sets the flags Cond2phX, Evap2phX, and Cap2phIn accordingly
FC	Calculates irreversibilities that are output as pseudo-constants
PROPFNCT.f	This file contains functions that interface with REFPROP
Initial	This subroutine initializes the property routines and selects a refrigerant
xht	Returns the quality of saturated refrigerant given enthalpy and temperature
xhp	Returns the quality of saturated refrigerant given enthalpy and pressure
htpsub	Returns the enthalpy of subcooled liquid given temperature and pressure (Btu/lbm)
PsatT	Return the refrigerant saturation pressure given the temperature (psi)
TsatP	Return the refrigerant saturation temperature given the pressure (F)
hpt	Return the enthalpy of subcooled or superheated refrigerant given pressure and temperature (Btu/lbm)
htx	Returns the enthalpy of saturated refrigerant given temperature and quality (Btu/lbm)
hpx	Returns the enthalpy of saturated refrigerant given pressure and quality (Btu/lbm)
vpx	Returns the specific volume of saturated refrigerant given pressure and quality (ft^3/lbm)
vtx	Returns the specific volume of saturated refrigerant given temperature and quality (ft^3/lbm)
vpt	Returns the specific volume of saturated refrigerant given pressure and temperature (ft^3/lbm)
saturation	This subroutine calculates the saturation properties given the pressure.
xph	Returns the quality given the pressure and enthalpy

File or Subroutine Name	Description
CvPT	This subroutine finds the constant volume and pressure specific heat and the velocity of sound given the temperature and pressure, only for the superheated and subcooled regions
CvPsat	This subroutine finds the constant volume and pressure specific heat and the velocity of sound of saturated vapor and liquid given the saturation pressure
spt	Returns the entropy for subcooled liquid or superheated vapor given the temperature and pressure (Btu/lbm-R)
spx	Returns the entropy for subcooled liquid or superheated vapor given the quality and pressure (Btu/lbm-R)

Appendix H: ASME and Monte Carlo Uncertainty Analysis

H.1 Introduction

In the closure to ASME's 1983 Symposium on Uncertainty Analysis, S. J. Kline (1985a) stated the primary conclusion of the symposium: "Uncertainty analysis is an essential ingredient in planning, controlling, and reporting experiments." Uncertainty analysis, if performed accurately, allows meaningful comparisons between experiments performed at different times and locations to be made (Moffat, 1991). Meaningful model validation requires uncertainty analysis to determine whether discrepancies between modeled and measured results may be due to uncertainties in measured input parameters rather than mis-modeling of the system in question (Kline, 1985b).

Estimating the uncertainty of a mathematical model requires specification of uncertainties for all important input parameters and propagation of those uncertainties through the modeling equations to yield an uncertainty in the result. This appendix discusses two common uncertainty analysis methods, ASME uncertainty analysis and Monte Carlo uncertainty analysis, and describes the implementation of both methods into the ACRC solver.

H.2 ASME Uncertainty Analysis

Much debate over the years has addressed the validity of various methodoligies for uncertainty analysis, but a degree of consensus has been achieved in recent years and has led to a standard method of analytical uncertainty analysis, which is well represented by the ASME Measurement Uncertainty standard (ASME, 1985). The ASME method draws from the work of various authors, including Kline and McClintock's seminal paper (1953) and is based on standard statistical principles. Abernethy (1985) provides a concise summary of the standard, and the reader is referred to it or the actual standard for the details of the ASME method, though its main features are discussed below.

Precision error is characterized by the precision index (S), which is the expected standard deviation of random errors in the measurement, and the bias error (B), or fixed error in the measurement. The precision index can be determined through statistical analysis of repeated measurements at steady state. Bias error can be reduced by careful experimental calibration, but may be impossible to measure statistically and in general must be estimated based on judgment (Abernethy, 1985).

Much of the debate regarding uncertainty analysis has dealt with the method for combining multiple bias or precision errors involved in a measurement. There is a finite, but very small, probability that multiple precision or bias errors will simultaneously have the same sign and a near-maximum value, which justifies summing the errors, i.e.

$$x = x_1 + x_2 + \ldots + x_k , \tag{H.1}$$

where x represents either B or S, the bias or precision error in a parameter, and k is the number of sources of uncertainty. Because of the low probability of simultaneous, like-signed, near-maximum errors, a root-sum-square (RSS) method "is widely used and is recommended" in the ASME standard (1985) because it accounts for some error cancellation. Thus bias or precision errors in a parameter are given by

$$x = \sqrt{x_1^2 + x_2^2 + \ldots + x_k^2} . \tag{H.2}$$

Once the precision and bias errors are determined, they can be combined to yield an overall uncertainty interval (U), which is analogous to a confidence interval, but not rigorously defined as such, since bias uncertainties are often estimated rather than statistically determined. The ASME standard resolves the debate over combining precision and bias uncertainties into an overall uncertainty by allowing either an additive (ADD) or RSS approach to be used (Abernethy, 1985).

$$U_{ADD} = B + tS \tag{H.3}$$

$$U_{RSS} = \sqrt{B^2 + (tS)^2} \tag{H.4}$$

In the above equations, t is the 95% confidence Student t statistic associated with S. Monte Carlo simulation was used to determine that when bias and precision errors are of the same magnitude, U_{ADD} covers 99% of the uncertainty, while U_{RSS} provides 95% coverage. If either the bias or precision error is negligible relative to the other, both U_{ADD} and U_{RSS} provide approximately 95% coverage (ASME, 1985).

The ASME standard justifies allowing two combination methods by specifying that the method (ADD or RSS) must be clearly stated in reported uncertainties and that bias and precision errors must be propagated through data reduction calculations separately, with combination being the last step so that it can be easily undone and converted to the alternate method.

As mentioned, bias and precision uncertainties in individual parameters must be propagated through data reduction calculations or modeling equations in order to determine their effect on the desired result. The ASME standard recommends a Taylor series method for properly accounting for the "linked" relationships between various parameters in calculating a result, although for large systems it may be impractical (ASME, 1985).

ASME also recommends a simpler approach for uncertainty propagation using sensitivity factors (θ_i). If the result (r) is dependent on J parameters, i.e.

$$r = f(P_1, P_2, ..., P_J), \tag{H.5}$$

then the sensitivity coefficient of r with respect to the ith parameter is given by

$$\theta_i = \frac{\partial r}{\partial P_i} \tag{H.6}$$

This partial derivative may be evaluated analytically, but a numerical estimation may be necessary if the data reduction calculations or model are too complex.

The precision and bias uncertainties of the result can then be calculated by

$$S_r = \sqrt{\sum_{i=1}^{J} (\theta_i S_i)^2} \tag{H.7}$$

$$B_r = \sqrt{\sum_{i=1}^{J} (\theta_i B_i)^2}. \tag{H.8}$$

The bias and precision uncertainties are combined by the RSS or ADD method to yield the overall uncertainty in the result (Abernethy, 1985).

H.3 Monte Carlo Uncertainty Analysis

As mentioned previously, the Monte Carlo method was used in the development of the ASME standard to evaluate the accuracy of the ADD and RSS error combination methods. In general, Monte Carlo uncertainty analysis is a more flexible and accurate technique of propagating the uncertainty in parameters through a model without the complexity of the Taylor series approach.

While the ASME standard uncertainty analysis method is statistically based on normal error distributions of the input parameters, the Monte Carlo technique involves evaluating the actual data reduction or modeling equations repeatedly with computer-generated distributions for the input parameters. The distribution of the results then represents uncertainty in the results due to the uncertainty in input parameters. Any input parameter distribution may be used, allowing improved accuracy for cases when a normal distribution is not the best representation of the uncertainty in an input parameter (Shannon, 1975). The Monte Carlo method is also more convenient for handling cases where uncertainty intervals are asymmetrical, e.g. due to radiation error in a thermocouple. The large number of model solutions required necessitates computer implementation, thus execution time may be a limiting factor for large systems.

Because all of the input parameters may be varied simultaneously and the actual propagating equations are used, the Monte Carlo method realistically accounts for error cancellation in the model. The ASME method accounts statistically for error cancellation, but may not accurately reflect the cancellation through a particular set of non-linear equations.

H.4 Applications to Thermal Systems

Both the ASME standard method and Monte Carlo analysis are useful for the uncertainty analysis of thermal systems. Cho (1987) used a RSS method that was fundamentally similar to the ASME method to evaluate the uncertainty in required area for a heat exchanger design. Badar (1993) reexamined the thermal model presented by Cho and performed a Monte Carlo analysis with normal, log-normal, and Weibull distributions for heat transfer coefficient errors and normal distributions for other parameters. The magnitudes of the uncertainties in input constants were held constant.

For the case where all input parameters had a normal distribution, Cho predicted 6.3 and 20% uncertainties in area for 80 and 99% confidence levels, respectively, while Badar found 6.6 and 29.5% uncertainties. Badar offered no rationalization for the larger discrepancy at higher confidence intervals. Badar also noted that the uncertainty in area was best represented by a Weibull distribution, even when the input distributions were all normal. Using different input distributions had only minor effects on the area uncertainties, which are increased to 6.8 and 30.3% when all the heat transfer coefficients are characterized by Weibull distributions.

H.5 Implementation with the ACRC Equation Solver

Both of the ASME uncertainty analysis methods (ADD and RSS) as well as a Monte Carlo uncertainty analysis (MC) have been implemented in the ACRC equation solver. Appendix å explains how to use the ACRC

solver and Table H.1 contains an example instructions file for an uncertainty analysis. The # of MC runs and R/N (rectangular/normal distribution) specifiers may be omitted for ADD and RSS runs. As indicated by the descriptions in Table H.1, the user selects the number of parameters for which uncertainties will be given and the number of variables for which distributions will be reported. XK#'s are used to indicate specifically which parameters and variables are to be used. Bias and precision values, as defined in Section H.2, are specified for each parameter on either a percentage or an absolute basis. Either a rectangular or normal parameter distribution can be selected. Although Badar (1993) indicated that other parameter distributions may be appropriate, they have not yet been implemented.

Table H.1 Sample instructions file for "UNCERTAINTY" analysis

Example	**Description**
`UNCERTAINTY`	Type of analysis (perform uncertainty analysis)
`XK`	Name of initialization file
`mcout`	Extender to append to output file names
`MC`	Specify type of uncertainty analysis: ADD, RSS, MC, or MCF
`25,14,1000`	# of parameters, # of variables, # of MC runs
`23 20 0 % N`	Parameter List:
`25 50 0 % N`	XK#, bias, precision, A/% (absolute or % basis),
`34 0.05 0 A N`	R/N (rectangular/normal distribution)
`37 1.0 0 % N`	"
`43 1.0 0 A N`	" (Note that tabs may be used instead of commas)
`45 0.3 0 A N`	"
`48 0.5 0 % N`	"
`94 1. 0 A N`	"
`95 1. 0 % N`	"
`98 0.025 0 A N`	"
`71`	Variable list: XK#'s
`72`	"
`77`	"
`82`	"
`89`	"

A very important assumption made in all of the analyses is that precision values are always based on sample sizes = 30, so that according to the ASME standard, the Student t statistic is approximated by a value of 2 (ASME, 1985).

H.5.1 Implementation of ASME Methods

For both the ADD and RSS methods, the first step after reading the input information is to calculate the sensitivity coefficients. Moffat (1985) suggests a "jitter" algorithm to numerically calculate the sensitivity coefficients by solving the model repeatedly for slightly altered parameter values. The fact that the Newton-Raphson algorithm provides a Jacobian matrix ($\mathbf{J} = \frac{\partial \mathbf{R}}{\partial \mathbf{x}}$, where **R** and **x** represent the residual function and variable vectors,

respectively), suggests a different approach. A "jitter"-type algorithm is used to calculate $\frac{\partial \mathbf{R}}{\partial k_i}$, and then the linear system $\mathbf{J} \bullet \frac{\partial \mathbf{x}}{\partial k_i} = \frac{\partial \mathbf{R}}{\partial k_i}$ is solved for $\frac{\partial \mathbf{x}}{\partial k_i}$, the vector of sensitivity coefficients with respect to the ith parameter. The linear system is solved with the same Gaussian elimination routine used by the Newton-Raphson solver.

Once the sensitivity coefficients are found, equations H.7 and H.8 are used to propagate the uncertainties, and equation H.3 or H.4 is used to calculate an overall uncertainty, labeled U_{normal} in the ADD and RSS output.

An example output file for a RSS analysis is given in Table H.2 for a sample model that will be discussed in Section H.6. Note that with the ASME analysis, the distribution specification is ignored. Input parameter specifications are reported in the output, as well as the bias and precision that was propagated to the variables. All values are reported on an absolute basis, even if they were input on a percentage basis. U_{actual} is only applicable to the Monte Carlo analysis.

Table H.2 Sample ASME RSS output file

RSS uncertainty analysis						
XK#	Name	Nom. value	Bias/Offset	Precision	Unormal	Uactual
23	PFrC	1	0.2	0		
24	PFrE	1	0.2	0		
25	qlossLL	50	25	0		
26	qlossDL	150	75	0		
27	qgainSL	220	110	0		
34	FanPercent	0.33	5.00E-02	0		
35	GSB	85	8	0		
36	GSN	85	8	0		
37	qE	10170	101.7	0		
38	t0	183.1	1	0		
39	t1	178.8	1	0		
40	t2avg	125.2	1	0		
41	t3	112.3	1	0		
42	t7i	45.5	1	0		
43	t9	60.6	1	0		
45	Tindoor	80	0.3	0		
46	Toutdoor	95	0.3	0		
47	FanPwr	227	1.135	0		
48	RacPwr	1260	6.3	0		
93	UAwall	7.9	1	0		
94	UAcab	8	1	0		
95	A1	82.875	0.82875	0		
96	A2	52	0.52	0		
97	A3	102	1.02	0		
98	Ftrans	1	2.50E-02	0		
53	QCab	120	15.379	0	15.379	0
54	QevapEB	10603	290.2	0	290.2	0

RSS uncertainty analysis						
XK#	Name	Nom. value	Bias/Offset	Precision	Unormal	Uactual
55	QFan	255.59	38.747	0	38.747	0
56	QWall	57.655	43.005	0	43.005	0
70	pwrmp	3530.6	35.561	0	35.561	0
71	pwrmapc	3051.7	105.35	0	105.35	0
72	CompPwr	3524.6	21.842	0	21.842	0
77	tsatincomp	43.918	1.088	0	1.088	0
78	tsatoutcomp	125.74	0.99721	0	0.99721	0
80	wmap	169.51	3.4009	0	3.4009	0
81	wmapc	176.62	3.7962	0	3.7962	0
82	wEB	153.38	4.2966	0	4.2966	0
85	wACRC	178.83	3.635	0	3.635	0
89	wASHRAE	166	4.3954	0	4.3954	0

H.5.2 Implementation of Monte Carlo Analysis

Section C.5 contains pseudocode for the uncertainty analysis. For each Monte Carlo run, input parameters are randomly generated according to the specified distributions, and the full model is solved. Solution values for the selected variables are written to the file "MC.*extender*"

If a particular Monte Carlo model run does not solve, a warning will be given, the parameter values will be written to the "MC.*extender*" file, and the error flag will be set, causing program execution to halt once all Monte Carlo runs have been attempted. The user may then attempt to solve those runs individually and edit "MC.*extender*" accordingly or remove the bad runs from "MC.*extender*." The statistical analysis can then be completed by running the MCF analysis, which loads a pre-existing "MC.*extender*" rather than generating a new one. The MCF option can also be used to combine the results of two smaller MC runs into one larger file for statistical analysis.

After all of the Monte Carlo runs have been completed (or loaded from a file in the case of MCF) , the "MC.*extender*" file is used to generate a histogram of values for each variable, which is saved to the file "H.*extender*." The offset ($x_{nominal} - x_{avg}$) and precision of the data around x_{avg} are then calculated, and an overall uncertainty (U_{normal}) is calculated according to equation H.4, which assumes a normal distribution.

In addition, the actual 95% coverage intervals (U_{actual}) based on the generated data are found via a symmetrical search, increasing the symmetrical U interval until 95% of the generated values lie within it.

An output file for a Monte Carlo analysis is given in Table H.3 for the sample model that will be discussed in Section H.6. Note that for this case the ASME RSS and the Monte Carlo analysis give similar predictions. This is often the case and the RSS method can be used as a good approximator for a Monte Carlo analysis.

Table H.3 Sample Monte Carlo analysis output file

MCF uncertainty analysis: 431 runs						
XK#	Name	Nom. value	Bias/Offset	Precision	Unormal	Uactual
23	PFrC	1	0.2	0		
24	PFrE	1	0.2	0		
25	qlossLL	50	25	0		
26	qlossDL	150	75	0		
27	qgainSL	220	110	0		
34	FanPercent	0.33	5.00E-02	0		
35	GSB	85	8	0		
36	GSN	85	8	0		
37	qE	10170	101.7	0		
38	t0	183.1	1	0		
39	t1	178.8	1	0		
40	t2avg	125.2	1	0		
41	t3	112.3	1	0		
42	t7i	45.5	1	0		
43	t9	60.6	1	0		
45	Tindoor	80	0.3	0		
46	Toutdoor	95	0.3	0		
47	FanPwr	227	1.135	0		
48	RacPwr	1260	6.3	0		
93	UAwall	7.9	1	0		
94	UAcab	8	1	0		
95	A1	82.875	0.82875	0		
96	A2	52	0.52	0		
97	A3	102	1.02	0		
98	Ftrans	1	2.50E-02	0		
53	QCab	120	0.69586	7.8057	15.627	16.235
54	QevapEB	10603	-1.2486	147.09	294.18	296.12
55	QFan	255.59	0.28265	18.873	37.748	36.649
56	QWall	57.655	0.24874	21.866	43.732	41.142
70	pwrmp	3530.6	0.585	19.283	38.57	38.46
71	pwrmapc	3051.7	-2.5733	56.544	113.12	111.82
72	CompPwr	3524.6	-0.16165	10.189	20.378	20.064
77	tsatincomp	43.918	5.64E-02	0.62732	1.2559	1.2429
78	tsatoutcomp	125.74	-7.31E-3	0.53082	1.0617	1.0849
80	wmp	169.51	0.17068	1.9559	3.9155	3.8408
81	wmapc	176.62	0.18166	2.1502	4.3043	4.2692
82	wEB	153.38	1.47E-02	2.1827	4.3654	4.1024
85	wACRC	178.83	-9.29E-2	1.8614	3.724	3.5603
89	wASHRAE	166	-0.27252	2.0804	4.1696	4.0465

H.6 Analysis of Refrigerant Mass Flow Component Models

A set of equations that includes the calculation of a refrigerant mass flow via an energy balance on an evaporator, two capillary tube models, and compressor maps for mass flow and power consumption was used as a

sample model in this Appendix. The results from the uncertainty analysis were used in the model validations in Section 4.

The calculation of refrigerant mass flow from the energy balance involves numerous parameters describing the heat transfer characteristics of the room air conditioner calorimeter. These parameters and their uncertainties were estimated by Rugg and Dunn (1994), and will not be described here.

The most critical part of any uncertainty analysis is the determination of uncertainties for the input parameters. Often, only educated guesses about the uncertainties of certain parameters are available. Uncertainties on thermocouples were specified as ±1 °F to account for thermocouple error as well as the fact that surface thermocouples will not measure the true refrigerant temperature. The power input to the calorimeter was assumed to have an accuracy of 1% and fan power measurement were assumed accurate to 0.5%. Pressure drop calculations were assumed accurate to within 20%. The instructions file in Table H.1 is an abreviated version of the one used to perform the uncertainty analyses.

The 95% uncertainty interval of w_{eb}, the mass flow calculated from the energy balance, is 4.3 lbm/hr for a nominal value of 153.4, or 2.8%, which represents an absolute uncertainty bound on the mass flow, assuming that the specified uncertainties and the equations used to calculate w_{eb} are correct.

The uncertainties for the capillary tube and compressor models do not represent absolute bounds on the values that they predict, but rather indicate the uncertainty that will be propagated through those models due to uncertainties in their input parameters. This information is useful to judge whether mispredictions of the models may be due to errors in the input parameters rather than errors in the models themselves.

H.7 Conclusions

Both the ASME uncertainty analysis methods and the Monte Carlo method can be helpful for evaluating thermal systems model predictions. The predictions of the RSS and Monte Carlo analyses were very similar for all of these models, as will often be the case. The Monte Carlo is ultimately more accurate, and was used by ASME to validate their method.

The ASME methods are fast, simple approximators of uncertainty and are thus well-suited for a screening analysis. A Monte Carlo analysis gives the most accurate uncertainty predictions, especially when modeling equations that are highly nonlinear, though it may require large amounts of computer time for large models.

Determination of the magnitudes of individual parameter uncertainties is the most important step in any uncertainty analysis, and will naturally be the controlling factor in the results. The overall uncertainty in a result may also be dominated by the uncertainties in only a few of its input parameters. An analysis of sensitivity coefficients may help identify which input parameters introduce the most uncertainty in the result.

References

Abernathy, R. B., R.P. Benedict, and R.B. Dowdell. “ASME Measurement Uncertainty.” Journal of Fluids Engineering 107 (June 1985): 161-164.

ASME. Measurement Uncertainty Part 1, Instruments and Uncertainty. ASME, 1985. American National Standard PTC 19.1-1985.

Badar, M Affan, Syed M Zubair, and Anwar K Sheikh. “Uncertainty Analysis of Heat-Exchanger Thermal Designs Using the Monte Carlo Simulation Technique.” Energy 18 (8 1993): 859-866.

Cho, S. M. “Uncertainty Analysis of Heat Exchanger Thermal-Hydraulic Designs.” Heat Transfer Engineering 8 (2 1987): 63-74.

Coleman, Hugh W. and Jr W. Glenn Steele. Experimentation and Uncertainty Analysis for Engineers. New York: John Wiley and Sons, 1989.

Coleman, Hugh W., M. H. Hosni, Robert P. Taylor, and Glenn B. Brown. “Using Uncertainty Analysis in the Debugging and Qualification of a Turbulent Heat Transfer Test Facility.” Experimental Thermal and Fluid Science 4 (1991): 673-683.

Dilks, David, Raymond Canale, and Peter Meier. “Development of Bayesian Monte Carlo Techniques for Water Quality Model Uncertainty.” Ecological Modeling 62 (1992): 149-162.

Kline, S. J. “The Purposes of Uncertainty Analysis.” Journal of Fluids Engineering 107 (June 1985b): 153-160.

Kline, S.J. “1983 Symposium on Uncertainty Analysis Closure.” Journal of Fluids Engineering 107 (June 1985a): 181-182.

Lassahn, G.D. “Uncertainty Definition.” Journal of Fluids Engineering 107 (June 1985): 179.

Moffat, R. J. “Establishing the Credibility of Experimental Work.” Experimental Thermal and Fluid Science 4 (5 1991): Back cover.

Moffat, R. J. “Using Uncertainty Analysis in the Planning of an Experiment.” Journal of Fluids Engineering 107 (June 1985): 173-178.

Moffat, R.J. “Contributions to the Theory of Single-Sample Uncertainty Analysis.” Transactions of the ASME 140 (June 1982): 250-120.

Moffat, Robert J. “Describing the Uncertainties in Experimental Results.” Experimental Thermal and Fluid Science 1 (1988): 3-17.

Rugg, Steve and W. E. Dunn. "Design, Testing, and Validation of a Room Air Conditioner Test Facility" University of Illinois at Urbana-Champaign, ACRC TR-59, 1994.

Shannon, Robert E. Systems Simulation: The Art and Science. Englewood Cliffs, N.J.: Prentice-Hall, 1975.

Appendix I: Condensate Spray Analysis

I.1 Introduction

When the room air conditioner evaporator removes moisture from the indoor air, condensate drains from the evaporator and travels through two channels to a pan in the outdoor side of the air conditioner. The flow of condensate represents an evaporative cooling potential equal to the mass flow rate of the water times the heat of vaporization of water. This cooling potential may be used in one of two ways to enhance condenser performance.

The water could be used to evaporatively cool the air before it passes over the heat exchanger, increasing the heat transfer due to the higher temperature difference between the air and the refrigerant. This is an inefficient use of the water's cooling potential, however, because it only indirectly cools the refrigerant stream, with the air serving as an intermediary.

A sample calculation was performed using an effectiveness-NTU based model of a condenser for a 12,000 Btu/hr room air conditioner at the highest fan speed with condenser inlet conditions set to experimentally measured values. By evaporating 5 lbm/hr of water into the air stream (a potential 5200 Btu/hr of evaporative cooling) the air temperature entering the condenser was reduced from 95°F to 87°F, causing an additional 700 Btu/hr heat removal from the coil, increasing condenser capacity by only 5% and representing a 13% utilization of the water's cooling potential.

The more efficient approach of spraying the condensate directly onto the condenser is commonly used in room air conditioners. The spraying is accomplished by the use of a plastic "sling ring" that is attached to the perimeter of the condenser fan blade. The "sling ring" dips down into the pool of condensate and slings the water up onto the condenser. Using the model and conditions described previously and assuming 100% utilization of the cooling potential of 5 lbm/hr water, the condenser capacity could be increased by 39% over a dry condenser case.

I.2 Theoretical Analysis and Literature Review

In reality, not all of the water coming from the evaporator ($\dot{m}_{total}$) will be utilized to enhance the condenser performance. A portion of $\dot{m}_{total}$ will be evaporated from the condensate pan ($\dot{m}_{pan}$) or from the "sling ring" or fan blade itself ($\dot{m}_{sling}$). Some water ($\dot{m}_{miss}$) may be slung onto the housing surrounding the fan and condenser rather than onto the condenser. Hopefully, the majority of the water ($\dot{m}_{eff}$, effective mass flow) will actually land on the coil and directly enhance heat transfer performance, though a portion of the water that does make it to the condenser could be blown completely through ($\dot{m}_{drift}$). Thus a mass balance on the water stream gives

$$\dot{m}_{eff} = \dot{m}_{total} - \dot{m}_{pan} - \dot{m}_{sling} - \dot{m}_{miss} - \dot{m}_{drift} . \qquad \text{(I.1)}$$

Order of magnitude calculations can estimate the importance of the various terms in the mass balance. According to Stoecker (1986) and ASHRAE Fundamentals (1993) using the Lewis relation and assuming that the Lewis number is unity gives

$$h_m = \frac{h_c}{C_{pm}} , \qquad \text{(I.2)}$$

where h_m is the mass transfer coefficient, C_{pm} is the specific heat of moist air, and h_c is the convective heat transfer coefficient. Using a heat transfer correlation for laminar flow over a flat plate as given in Incropera and Dewitt (1990), potential rates of water evaporation were estimated. These calculations indicated that evaporation from the condensate pan ($\dot{m}_{pan}$) would be around 0.05 lbm/hr and evaporation from the "sling ring" ($\dot{m}_{sling}$) would generally be less than 0.1 lbm/hr. These numbers are quite small compared to a $\dot{m}_{total}$ of 1 to 6 lbm/hr, which is typical of a room air conditioner. Observations of Tree and Goldschmidt (1978) and observations of air conditioners investigated in the current project indicate that $\dot{m}_{drift}$ is virtually zero, as no water is observed to be blown through the condenser. The quantity $\dot{m}_{miss}$ is difficult to analytically estimate because of lack of knowledge of the spray pattern, though it may prove to represent a significant mass flow of water evaporating into the air upstream of the condenser.

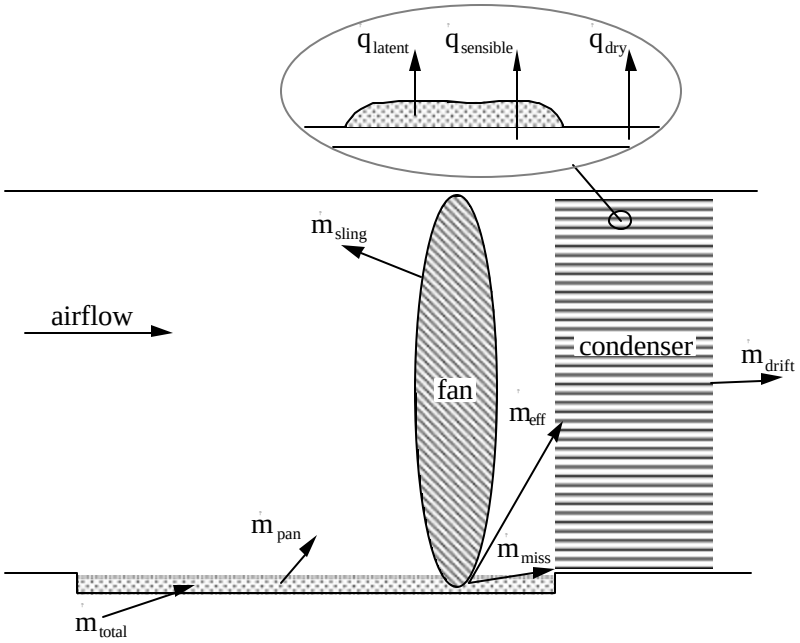

Figure I.1 Condensate spray schematic.

Also, since each water droplet may initially be cooler than the air, it can draw a portion of its heat of vaporization from the air rather than the condenser fins, reducing the utilization of $\dot{m}_{eff}$'s cooling potential. However, the temperature difference between the water and the air would be significantly smaller than that between the water and the fin, and as the water becomes warmer than the air, it will begin heating the air rather than being heated by it. The conduction resistance between the water and the fin is also much smaller than the convection resistance between the water and the air. Thus the majority of the evaporative cooling potential should be drawn directly from the condenser fins.

In a typical condenser utilizing condensate spray, only a portion of the condenser surface will be wetted. Both latent and sensible heat transfer will occur on the wetted portion of the condenser. The latent transfer will be simply $\dot{m}_{eff}$ times the heat of vaporization of water.

The sensible heat transfer is less straightforward to determine. The thickness of water introduces a small conduction resistance while simultaneously increasing surface area and roughness. The presence of the water also increases the degree of air flow constriction in the condenser, increasing the air velocity while possibly reducing total air flow due to pressure drop. Myers's work (1967), reported by Threlkeld (1970), indicates a net effect of a slight increase in the air-side sensible heat transfer for a wetted fin, compared to an equivalent dry fin. The sensible heat transfer coefficient is ~12% higher in an example given by Threlkeld, which would typically translate into an ~ 9% increase in overall UA, assuming that the refrigerant side resistance is about half of the air-side resistance. Strictly speaking, one would not expect the heat transfer coefficient between the water and air to be any higher than that between a bare fin and air, but because our practical definition of the heat transfer coefficient is based on the dry fin area, the surface area increase due to droplet area gets incorporated into the heat transfer coefficient

Based on an empirical study by Myers, the wetted sensible heat transfer coefficient can be related to a dry coefficient by

$$h_{s,w} = h_{s,d} \cdot 0.626V^{0.101}, \tag{I.3}$$

where V is the standard air face velocity in ft/min. Equation I.3 predicts wet air-side sensible heat transfer coefficients to be 10 to 25% higher than the equivalent dry coefficients, corresponding to approximately 7 to 20% increases in overall UA, not accounting for conduction resistance between the tube wall and the water surface. The increase in the sensible heat transfer convection coefficient may be due to the surface roughness and increased surface area of the water and the increased core air velocity caused by restriction of the condenser's free flow area.

Myers work was also used by Fischer and Rice (1983) in the development of a dehumidification heat transfer analysis for the evaporator model of the DOE heat pump model. Running this evaporator model yielded 7.7% higher sensible heat transfer in wetted regions of the evaporator for one case, or about an 11% increase in the sensible heat transfer coefficient on the wetted fraction of the evaporator.

The above indicates that sensible heat transfer in the wetted regions will be enhanced rather that reduced, and Threlkeld indicates that Myers's equation may be used as an approximation of the sensible heat transfer coefficient.

Using Equation I.2, h_m for the wetted condenser can be determined based on a heat transfer coefficient. Then, given the $\dot{m}_{eff}$ applied to the coil, h_m can be used to determine an effective wetted area, A_e, for calculation of increased sensible heat transfer:

$$\dot{m}_{eff} = h_m A_e (\omega_o - \omega_\infty), \tag{I.4}$$

where ω_o and ω_∞ are the humidity ratios of saturated air at the water temperature and of the ambient air.

Tree and Goldschmidt (1978) assert that the heat transfer for a wet coil is approximately the heat transfer of an equivalent dry condenser plus the latent heat of an effective water mass flow, $\dot{m}_{eff}$. They give an expression for $\dot{m}_{eff}$

$$\dot{m}_{eff} = 1 - c_3 \frac{\dot{m}_{total} d^{n-1}}{\rho_w A_e}, \tag{I.5}$$

where d is water droplet diameter, n is empirically determined, A_e is the wetted area of the condenser, ρ_w is the density of water, and c_3 is empirically determined based on droplet evaporation rate. Tree and Goldschmidt indicate that the form of predictions based on Equation I.5 agree with their experimental data, although they do not show any predicted versus experimental results. They also assumed a wetted area equal to 1/8 of the total surface, rather than calculating an area based on $\dot{m}_{eff}$.

The experimental results of Tree and Goldschmidt indicate that heat transfer enhancements were most strongly affected by the total mass flow of water spray and were not consistently affected by drop diameter. They obtained condenser capacity enhancements of up to 40% for the highest water flow rates.

Grissom (1981) discusses spray cooling in general, focusing on determination of the point at which flooding of a surface will occur, and calculation of the maximum rate of evaporation possible, given a particular surface temperature and ambient pressure. He also examines the onset of the Leidenfrost state.

His discussion of dry-wall, transition, and flooded modes of evaporation suggest that due to the particular geometry of the condenser and spray mechanism, the front edge of the condenser may be flooded with more water than it can evaporate, with excess water being blow farther back in the condenser. The actual conditions are not readily observable due to the difficulty of seeing inside the running air conditioner. Tree and Goldschmidt reported observing dropwise evaporation and indicated that only 1/4 of the depth of the condenser was wetted. They reported a droplet evaporation time of about .5 seconds after impact on the fin.

Given these considerations, it can be assumed that the mass flow of water to the condenser in a typical room air conditioner is not in excess of the maximum amount that can be evaporated. This conclusion is also supported by the absence of drift of water droplets through the condenser exit on two units that we have tested.

McQuiston (1978) has done extensive work on combined heat and mass transfer for dehumidification coils, using a mass transfer analogy to the Colburn j-factor analogy. McQuiston's analysis may be valid for evaporation as well, but the j-factor method is not easily amenable to our current modeling methods.

As stated earlier, $\dot{m}_{eff}$ cannot be evaluated analytically using our known information, and its reasonable determination will require experimental analysis. A proposed method of evaluating an approximate expression for the $\dot{m}_{eff}$ is given below.

I.3 Experimental Analysis

Using a room air conditioner testing facility, we can experimentally determine condenser capacity and the water removal rate by the evaporator, as well as air and refrigerant-side condenser inlet conditions. Experimental data for a case with no dehumidification and thus no water spray can be taken to establish a base capacity of Q_{dry}.

A condenser model can be calibrated to give accurate prediction of Q_{dry} based on the measured inlet conditions. Another data point can then be taken with water supplied to the condenser by dehumidification condensate, where $\dot{m}_{total}$ can be considered equal to the measured water removal rate by the evaporator. The condenser heat transfer for the wet case will be denoted Q_{wet}. The condenser inlet conditions for this second point will be different from those for Q_{dry}, so the condenser model must be used to predict what the dry heat transfer, Q'_{dry}, would have been in the absence of the water spray.

Thus the heat transfer increase due to the condensate spray is given by

$$\Delta Q = Q_{wet} - Q'_{dry} = \dot{m}_{eff} * h_{fg} + Q_{se}, \tag{I.6}$$

where ΔQ is the heat transfer enhancement and Q_{se} is the sensible heat transfer enhancement over the area A_e due to the increased sensible heat transfer coefficient. Equation I.6 can then be solved for $\dot{m}_{eff}$:

$$\dot{m}_{eff} = \frac{Q_{wet} - Q'_{dry} - Q_{se}}{h_{fg}}. \tag{I.7}$$

According to Tree and Goldschmidt's observations, generally less than 25% of the condenser is wetted by the spray, thus if the sensible heat transfer enhancement were 10%, the overall effect of the enhanced sensible heat transfer would be about 2.5%. Considering the much larger uncertainty in evaporative losses prior to reaching the coil and the limits of our experimental accuracy, the sensible enhancement can reasonably be ignored, as Tree and Goldschmidt did. Given this assumption, the Equation I.7 can be rearranged to give

$$\dot{m}_{eff} = \frac{Q_{wet} - Q'_{dry}}{h_{fg}}. \tag{I.8}$$

Using several experimental determinations of $\dot{m}_{eff}$, an expression can be found to relate $\dot{m}_{eff}$ to $\dot{m}_{total}$.

Based on observations by Tree and Goldschmidt (1978), we expect that the enhancement due to the condensate spray would be most strongly affected by the mass flow of water supplied to the condenser pan. Geometric factors that determine how much of the sprayed water directly hits the condenser would also have a significant affect. The coil temperature and the humidity ratio of the ambient air should also affect the evaporative flux on a given area of the heat exchanger, but since excess water tends to be blown on to dry areas in actual operation, the overall heat transfer rate should not be strongly affected. However, the changes in the rate of evaporation before the water actually reaches the coil could have a significant effect on $\dot{m}_{eff}$.

Thus a proposed form for a relationship between $\dot{m}_{eff}$ and $\dot{m}_{total}$ is

$$\dot{m}_{eff} = \dot{m}_{total} - h_m A * (\omega_o - \omega_\infty) - C_1 * \dot{m}_{total}, \tag{I.9}$$

where h_mA represents an effective mass transfer coefficient times area (analogous to UA) for evaporation from the condensate pan and other constant evaporations.

Predictions of heat transfer in the condenser can be made with the equation

$$Q_{wet} = Q'_{dry} + \dot{m}_{eff} * h_{fg}. \tag{I.10}$$

The coefficient C_1 accounts for the physical efficiency of the fan in delivering water to the evaporator coil. Observations indicate that a significant portion of condensate is sprayed onto the condenser fan motor and the air conditioner cabinet, where it will draw heat from the those surfaces and thus not effect an heat transfer enhancement.

The coefficients h_mA and C_1 were evaluated based on experimental data taken with a room air conditioner testing facility. A least squares minimization indicated that the term containing h_mA had a negligible effect, and its estimated value was negative, so h_mA was set to zero. The estimated value for C_1 was 0.73, which, in the newly

simplified correlation, must account for the physical efficiency of the sling ring as well as for small amounts of evaporation.

I.4 Conclusion

Equation I.10 can be easily implemented in a mathematical model of a room air conditioner. Because the air conditioner geometry and the design of the "sling ring" can vary significantly between different units, this method is strictly only valid when experimental data is available, although the general behavior of different air conditioners should remain the same, and the equation should provide a reasonable estimate of condensate spray performance for a variety of air conditioners.

References

ASHRAE 1993. Handbook of Fundamentals. ASHRAE, Atlanta.

Fischer, S.K. and C.K. Rice 1983. "The Oak Ridge Heat Pump Models." ORNL/CON-80/R1. Oak Ridge National Laboratory.

Grissom, W.M. and F.A. Wierum 1981. "Liquid Spray Cooling of a Heated Surface." International Journal of Heat and Mass Transfer., 24, 261-271.

Incropera, F.P. and D.P. DeWitt 1990. Introduction to Heat Transfer. 2nd ed. Wiley and Sons, New York.

McQuiston, F.C. 1978. "Correlation of Heat, Mass, and Momentum Transport Coefficients for Plate-Fin-Tube Heat Transfer Surfaces with Staggered Tubes." ASHRAE Transactions., 84(1).

Myers, R.J. 1967. "The Effect of Dehumidification on the Air Side Heat Transfer Coefficient for a Finned-tube Coil". Master's Thesis, University of Minnesota.

Stoecker, W.F. and J.W. Jones 1986. Refrigeration and Air Conditioning. 2nd ed. McGraw Hill, New York.

Threlkeld, J.L. 1970. Thermal Environmental Engineering. 2nd ed. Prentice Hall, Englewood Cliffs, New Jersey.

Tree, D.R., V.W. Goldschmidt, *et al*. 1978. "Effect of Water Sprays on Heat Transfer of a Fin and Tube Heat Exchanger." Proceedings of Sixth International Heat Transfer Conference, Vol. 4.