4. REFRIGERATION PROCESS CONTROL: SIMULATION MODEL

In this chapter, the development of the simulation model for the two-stage refrigeration system is presented. The model is based on the mathematical model developed in Chapter 3, and is written using the programming language ACSL.

4.1. ACSL:

ACSL (**A**dvanced **C**ontinuous **S**imulation **L**anguage) is a commercial high level computer language. It has been developed for the purpose of simulating systems described by time dependent, non-linear differential equations and transfer functions. Typical application areas are control system design, chemical process representation, heat transfer analysis and missile and aircraft simulation. The use of ACSL has increased in dynamic simulation applications, e.g. Fararooy *et al.* (1993), Havranek (1996), Gränfors *et al.*, 1997; Dixon, 1997; Dixon *et al.*, (1998). In process control, it has been suggested by several authors (Coughanowr, 1991; Ogunnaike and Ray, 1994) as one of the high level computer-aided analysis packages to be used. For a comprehensive list of research conducted using ACSL, see MGA (1998).

ACSL program code can be developed from block diagrams, mathematical equations or conventional FORTRAN statements. The language code is based on FORTRAN, and many of its commands have a similar format. Once the model code is written, it is

translated into FORTRAN or C++ and is then compiled using a standard compiler. One of the main features of ACSL is its capability of sorting the continuous model equations, in contrast to conventional programming languages such as FORTRAN, which depend on statement order (MGA, 1996).

In general ACSL programs are written in a flexible explicit structure. This structure is illustrated in Figure 4.1 below.

In practice the DERIVATIVE section includes all the equations that describe the model. ACSL sorts all equations automatically inside this section. Hence, it should be noted that for programs whose nature needs a reinforced imposed definite structure (e.g. using Do-loops), the code should be included in a PROCEDURAL block, inside which the code is not sorted automatically by ACSL.

```
PROGRAM

        INITIAL
                Statements executed before the run begins.
                State variables do not contain the initial conditions yet.
        END

        DYNAMIC
            DERIVATIVE
                Statements to be integrated continuously.
            END
            DISCRETE
                Statements executed at discrete points in time.
            END
                Statements executed at each communication interval.
         END

         TERMINAL
                Statements executed after the run terminates.
         END

END
```

Figure 4.1: Outline of explicitly structured programs in ACSL

ACSL has two important powerful features: it can handle integration by its build-in integration algorithms, and it can solve for any variable defined implicitly in an expression. ACSL handles integration by a centralised integration routine. In performing integration, the integration algorithms utilise two intervals: the integration step size and the communication interval. The integration step is the fundamental interval over which the state variables are updated in the course of calculation. Several integration algorithms (both fixed step and variable step) are available in ACSL. In addition the program allows the user to define other integration algorithms if necessary. The second feature is that ACSL is capable of solving for a variable defined implicitly (i.e. via trial and error) in an expression. It does so by evaluating a defined expression as a residual and keeping its value at or close to zero.

For executable models (i.e. code translated, compiled and linked), ACSL has the capability to perform a linear analysis for the models with its command ANALYZE. It can evaluate the Jacobian, calculate eigenvalues and their associated eigenvectors, and generate frequency response analysis and plots.

4.2. MODEL IMPLEMENTATION:

The model has been written using double precision calculations. This was necessary to achieve better accuracy in the calculation, especially when extracting the linearised model. The structure of the program follows the explicit structure of ACSL model, and hence is divided into two main sections. First, an INITIAL section that contains all the constant statements that define the values of the variables, as well as any equations needed to evaluate the initial conditions of the state variables. Second, a DYNAMIC section in which all the calculations are conducted. This section includes both a DERIVATIVE section in which all the model equations are stated including both the integration and the implicit statements, and a DISCRETE section where the load disturbances are introduced to the model. This model does not include a TERMINAL section. A complete listing of ACSL program is found in Appendix A.

The model includes 6 state variables (the total refrigerant hold-up in the first

evaporator W1, the total refrigerant hold-up in the second evaporator W2, the refrigerant vapour hold-up in the condenser W4, the total enthalpy of the refrigerant in the first evaporator H1, the total enthalpy of the refrigerant in the second evaporator H2 and the temperature of the refrigerant in the receiver T3) defined by 6 differential equations (Equations 3.34-3.39). These equations are solved numerically by ACSL using its embedded integration methods. The integration routines consist of the integration method and the integration step size. The choice of the integration step is discussed in the next section.

The implicit solver in ACSL is used to solve for 10 variables, using the embedded solver IMPLC. These include two variables for each compressor stage discharge temperature TD_i and head h_i; and two for each evaporator to solve for its temperature T_i and vapour refrigerant hold-up WV_i. A ninth implicit statement is used to solve for the saturation temperature in the condenser T3 (referred to as T_c in the mathematical model), and the last is to solve for the inter-stage mixing node pressure PA (referred to as PD_1 in the mathematical model). The latter proved to be the critical point of the program and its main problem, as its convergence needed a fairly close initial guess to converge.

The control algorithms are also included in the DERIVATIVE section. Each PI controller adds one differential equation to the model to solve for the integral action. Another integral needs to be solved to evaluate the ISE for each controller as well. Any load disturbance or set point change is introduced in a DISCRETE section in the DYNAMIC section, at the time specified.

The ANALYZE feature of ACSL is used during the runtime to perform the linear analysis. The Jacobian function is invoked to obtain the Jacobian matrix **A** defined in Equation 2.1. Obtaining the other system matrices **B**, **C** and **D** (in Equations 2.1 and 2.2) requires the definition of two vectors of variables: the CONTROL vector includes the required manipulated variables and disturbances, and the OBSERVE vector includes the controlled variables. In this analysis the OBSERVE vector consists of: the levels in both evaporators (L1,L2) and the receiver (L3), the pressure in all three drums (LP – P1, IP – P2, HP – P3) and the outlet process stream temperature TP1o. Several versions of CONTROL vectors are used depending on the

case analysed. For the basic case, the variables included are the liquid valve opening XV2, the liquid valve opening XV3, the compressor speed N, the gas valve opening XV1, the condenser cooling FCP3. Other cases require the substitution of the liquid flowrate into the first evaporator FL2, the liquid flowrate into the second evaporator FL3 for XV2 and XV3, the gas flowrate from the second evaporator FG2 for XV1 and the pressure in the first evaporator P1 for N. In the latter case P1 must be removed from the OBSERVE vector.

The variables in the CONTROL list must not be calculated (i.e. they must be defined in CONSTANT statements). However, in most models (including this) the manipulated variables are calculated quantities as the controllers are built into the models. So to obtain the open loop state matrices, it is necessary either to use the model without controllers, or to modify the model's code and to include extra variables that can be used. This is done by setting the gain factors to zero and then defining a CONSTANT variable, associated with the manipulated variable, setting its value to zero and using it in the CONTROL vector. As an example, the valve opening XV2 is calculated in the model as follows:

$$XV2 = K1 * (ER1 + INTEG(ER1,0)/TI1) + XV2ic \tag{4.1}$$

where K1 is the controller gain, TI1 is the integral time, ER1 is the error in the controlled variable, and XV2ic is the initial value of XV2. The code is modified so

```
CONSTANT  XV2lin=0
XV2 = K1 * (ER1 + INTEG(ER1,0)/TI1) + XV2ic + XV2lin
```
(4.2)

So XV2lin is used in the CONTROL vector instead of XV2 during the linearisation.

A FREEZE function is used in the analysis to eliminate some variables from the state vector. Its purpose here is to eliminate the controller integrations and the ISE calculations.

4.2.1. Choosing the integration method and step size:

The choice of the integration step size is critical in determining the stability of the

numerical solution and the convergence of the model. Thus it is also very important to choose the appropriate integration method, and optimising the integration step size and the algorithm chosen generally worth the effort.

In ACSL, as mentioned earlier, the integration routines consist of the integration method and the integration step size. Also, the integration algorithms can be classified into two broad methods: fixed-step integration algorithm and variable-step integration algorithm and thus ACSL has two different ways of determining the integration step size depending on the method. In fixed-step algorithms, the integration step HS is calculated in Equation 4.3, where MAXTERVAL is the upper bound on the integration step size for both variable step and fixed step algorithms, CINT is the communication interval at which the DYNAMIC section is executed and NSTEP is the number of integration steps in a communication interval.

$$HS = \min (MAXTERVAL, CINT/NSTEP) \tag{4.3}$$

As a general rule, ACSL manual recommends always keeping NSTEP equal to 1, so the integration step size is determined solely by MAXTERVAL.

In variable integration algorithms, the integration step HS means the lower bound on the step, and is determined in Equation 4.4, where MINTERVAL is the lower bound on the integration step. Its default value is 1.0E-10.

$$HS = \max (MINTERVAL, \min (MAXTERVAL, CINT/NSTEP)) \tag{4.4}$$

A trade-off between the efficiency (computation time) and the accuracy of the algorithm should be put into consideration. For most cases the ACSL manual recommends using the Runge-Kutta Second Order Method. This needs a smaller integration step size than Runge-Kutta Fourth Order but it is much faster. On the other hand the Euler Method is generally faster but it needs even smaller integration step size to achieve acceptable accuracy. Using a variable integration step algorithm (such as Gear's Method) may be beneficial for stiff systems only if the range of time constants is as high as three or four decades, then these techniques may be significantly faster than the others.

For the refrigeration model, ACSL recommendation to use Runge-Kutta Second Order Method was tested and compared to both Euler Method and Runge-Kutta Forth Order Method. The accuracy (represented by absolute error) of the solution was compared to the CPU time and the integration step size. The CPU time was obtained by using SPARE command in ACSL which determines the central processor time, both accumulated from the start of the run and elapsed since the previous use of the command. Hence to determine the elapsed CPU time, the following sequence in ACSL should be used: SPARE, START, SPARE. All computations were carried out on a PC with a Pentium 133 processor, 32 MB RAM and Windows 95.

A preliminary guess to determine the integration step size using the eigenvalues of the model recommended the value of 1, and showed that the use of variable integration step methods is not beneficial. A comparison between three integration algorithms (Euler, Runge-Kutta Second Order, Runge-Kutta Forth Order) was carried out, and the Runge-Kutta Forth Order Method was chosen to achieve more accuracy which is needed in the derivation of the linear model.

4.3. LINEAR MODEL:

The ANALYZE command described previously was used to obtain a linearised model for the two-stage refrigeration system model. The state-space model has the form described in Equations 2.1 and 2.2.

$$\frac{d\mathbf{x}}{dt} = \mathbf{A}\,\mathbf{x} + \mathbf{B}\,\mathbf{u} \tag{2.1}$$

$$\mathbf{y} = \mathbf{C}\,\mathbf{x} + \mathbf{D}\,\mathbf{u} \tag{2.2}$$

where x is the vector of state variables, u is the vector of inputs (manipulated variables and load disturbances), y is the vector of outputs (controlled / measured variables), and **A**, **B**, **C**, and **D** are constant matrices of appropriate dimensions.

When applying the ANALYZE function, the measured variables are called “observe” variables, and the manipulated variables “control” variables. In deriving

the linearised model, all candidate measurements are included in the y-vector, and all candidate manipulated variables and disturbances are included in the u-vector.

A generalised linearised model composed of 7 candidate measurements, 5 candidate manipulated variables and one disturbance has been derived using the ANALYZE command. The resulting system matrices **A**, **B**, **C**, and **D** are found in Figure 4.2. In this figure, the order of the matrices corresponds to the following **x**, **u**, and **y** vectors.

- x = {H1, H2, T3, W1, W2, W4}
- u = {XV2, XV3, N, XV1, FCP3, TP1i}
- y = {L1, L2, L3, P1, P2, P3, TP1o}

A

-0.07429	0.087509	0	43.2531	-61.4813	10.208
0.02175	-0.21926	10.3165	-14.173	150.702	15.5889
-7.37E-13	-3.19E-12	-0.00125	-1.33E-05	-1.33E-05	1.63E-04
-5.79E-05	9.28E-05	0	0.033571	-0.06509	0.009708
2.10E-05	-2.11E-04	0	-0.01379	0.14503	0.016418
3.34E-05	8.35E-05	0	-0.0135	-0.05154	-0.07788

B

2838.32	0	-31981.4	536.783	0	81.3754
-2838.32	5209.06	-2771.39	-2016.15	0	0
0	0	-3.06E-07	-5.79E-08	-2.62E-09	0
4.05719	0	-30.3864	0.509234	0	0
-4.05719	6.3425	-2.533	-1.84407	0	0
0	0	31.483	1.06368	-0.01234	0

C

8.81E-07	0	0	0.001166	0	0
4.08E-09	4.16E-07	0	-2.83E-06	0.001587	5.11E-06
0	0	0	-0.00242	-0.00242	0.00135
7.11E-06	0	0	-0.00422	0	0
8.37E-08	1.05E-04	0	-5.82E-05	-0.07268	1.05E-04
0	0	0	-0.00373	-0.00373	0.0459
1.24E-04	0	0	-0.07327	0	0

D

0	0	0	0	0	0
0	0	-0.00143	-3.89E-04	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	-0.02942	-0.00798	0	0
0	0	0	0	0	0
0	0	0	0	0	0.269482

Figure 4.2: System matrices

Other models using the flowrates FL2, FL3, FG2, and the pressure P1 as manipulated variables were also derived for the use in control analysis in Chapter 6.

4.4. ACSL MODEL VALIDATION:

The ACSL linearised model was validated by comparing its performance to the ACSL non-linear model, and to the FORTRAN stand alone model used by Wilson and Jones (1994).

Table 4.1 shows a comparison of the eigenvalues obtained from ACSL ANALYZE with the values of Wilson and Jones (1994).

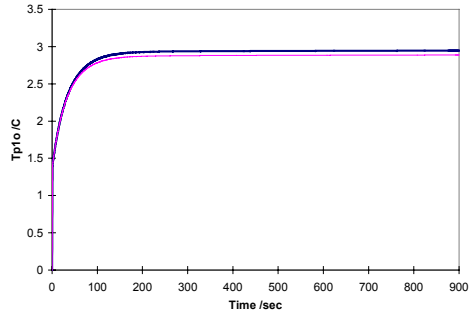
Table 4.1: Comparison of the eigenvalues of the system

Wilson and Jones (1994)	Current Model
0	0.000000
0	0.000000
-0.0012	-0.001198
-0.0272	-0.026586
-0.0685	-0.069902
-0.0938	-0.096939

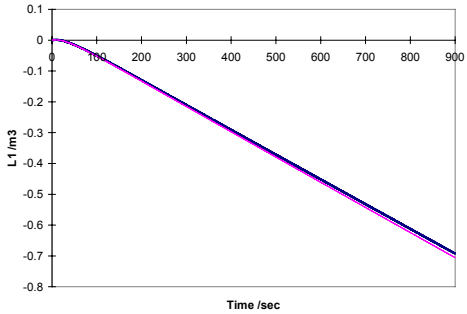
The agreement shown in the eigenvalues is satisfactory, and the small differences are due to the lower precision used by the previous workers.

A step change of $+5^{o}C$ in the process stream inlet temperature TP1i is introduced, and the transient response in all candidate measured variables is compared with all controllers left on manual.

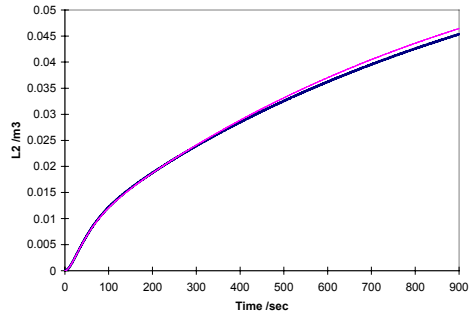
Figure 4.3 shows the results of comparing the non-linear and the linearised model. As shown in the figure, the agreement between the models is satisfactory. Consequently the linearised model can be used in further control analysis.



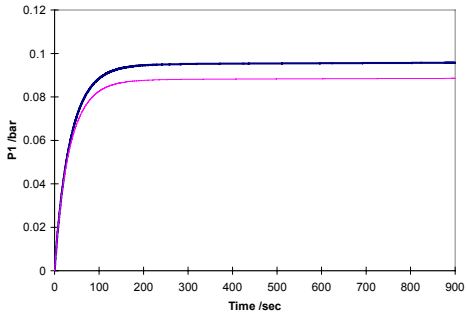
(a) TP1o Perturbation



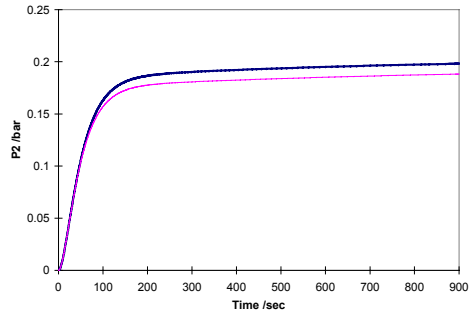
(b) L1 Perturbation



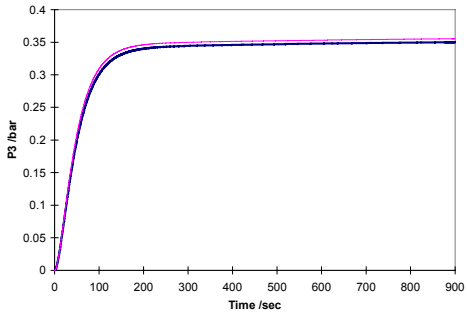
(c) L2 Perturbation



(d) P1 Perturbation



(e) P2 Perturbation



(f) P3 Perturbation

non linear model linear model

Figure 4.3: Validation of the linearised model
(transient uncontrolled responses to a step change of +5^{o}C in the process stream inlet temperature TP1i)

4.5. MODEL LIMITATIONS:

The model was developed bearing in mind the possibility of future expansion. As a result, the coding for each unit was treated as a separate object to allow its replacement or development if needed. However, in its current state, the model has several limitations that should be noted.

- The compressor model used is simplified and is based on real data for a specific case. So, when applying the model for other cases, this part needs to be replaced.

- The model is sensitive to initial conditions when solving the implicit loops, especially to solve for intermediate pressure. Thus a close guess is required to achieve convergence.