MONITORING AND DIAGNOSIS OF MULTI-TANK FLOWS USING QUALITATIVE REASONING

APPROVED:

Supervisor: ____

Benjamin Kuipers

____

Bruce Porter

MONITORING AND DIAGNOSIS OF MULTI-TANK FLOWS USING QUALITATIVE REASONING

by

HERBERT KAY, B.S.

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF ARTS

THE UNIVERSITY OF TEXAS AT AUSTIN

May, 1991

Acknowledgments

This work has taken place in the Qualitative Reasoning Group at the Artificial Intelligence Laboratory, The University of Texas at Austin. Research of the Qualitative Reasoning Group is supported in part by NSF grants IRI-8905494 and IRI-8904454, by NASA grant NAG 2-507, by the Texas Advanced Research Program under grant no. 003658175, and by the Jet Propulsion Laboratory.

Herbert Kay

The University of Texas at Austin

May, 1991

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

The tasks of monitoring and diagnosis are necessary to maintain the proper operation of any dynamic system. As systems become more complex, these tasks become correspondingly more complex, and computer-based techniques are often used to make the analysis of the system easier. To be effective, a monitoring system must detect deviations from the normal operating state of its underlying process. One method to detect deviations is to construct a model of the system based on a theoretical or empirical understanding of the process and to compare the predictions of the model to the values of the process variables as read from sensors. Often, these models are described by a system of differential equations and can be solved numerically by differential equation solvers. The difficulty with this approach is that 1) sufficient understanding of the process may not exist to construct an accurate numerical model, and 2) the actual process variables and functions may vary across process runs. In these cases, an approximate model can sometimes be constructed which makes a best guess at the behavior of the system based on an "average" process run. While this approach is appropriate for some processes, it may not be possible for large systems with with complex process dynamics[Cheung and Stephanopoulos, 1989].

Qualitative reasoning addresses the problem of simulating with inexact models. A qualitative model is an accurate model of a system, but does not contain the same level of detail as does a numerical model. In cases where an exact model is unknown, the qualitative model nonetheless permits simulation of the system, unlike conventional simulation. Uncertainty in a model translates into uncertainty in the simulation output. In a qualitative simulation, this uncertainty is captured by producing multiple behaviors from a single model. Because of this ability to represent inexact models and simulation results, qualitative modeling and simulation provide a useful method to reason about inexact systems.

One of the well-known qualitative reasoning systems is the QSIM system. [Kuipers, 1986]. A QSIM model is a set of constraints which is an abstraction of an ordinary differential equations (ODEs). Because of this similarity, the constraint set is referred to as a qualitative differential equation or QDE. The QSIM simulation algorithm generates a set of behaviors for each QDE. These behaviors are guaranteed to the solutions of any ODE system that the QDE abstracts. Recently, the QSIM algorithm has been augmented to use numerical information in addition to the more abstract qualitative constraints. This new system, called Q2 [Kuipers and Berleant, 1988] uses a range interval propagation method to deduce numerical ranges for the variables of a system over the course of a simulation. Q2 has been used as part of MIMIC [Dvorak and Kuipers, 1989], a qualitative process monitoring system. MIMIC takes a set of Q2 models representing the working state of a system as well as specific fault models of the system and uses them to track the health of a system by comparing process sensor readings to the predictions of the models. MIMIC is capable of monitoring systems for which exact numerical models do not exist; however, its predictions are often very loose due to the inexactness of the underlying range propagation methods.

1.2 Goals of this Research

This research considers the task of process simulation and monitoring as applied to systems which can be modeled as systems of tanks. The class of systems that can be modeled in this manner is large, as it includes any system which can be modeled by substances and flows. Additionally, this research defines a method for combining qualitative and numerical reasoning that uses the precision of numerical simulation to improve the bounds on the simulation results while retaining the ability to reason about inexact models. This combined method is then used to solve a process monitoring problem in the area of vacuum technology – the monitoring of a vacuum system during pumpdown.

1.3 Outline of this Thesis

Chapter two discusses the construction of multi-tank models using QSIM with special attention to the modeling of vacuum systems. Chapter three discusses a method for improving the predictions of the Q2 system over time intervals by using a method called "dynamic envelopes." Examples of the systems in the previous chapter are reexamined using this new simulation method. Chapter four discusses the process monitoring problem and evaluates the value of the dynamic envelope method by comparing a version of the MIMIC system augmented with dynamic envelope simulation to one that does not use it. Chapter five discusses the results of this research and suggests future areas to explore.

Chapter 2

Qualitative Modeling of Multi-compartment Systems

2.1 Introduction

Many physical processes can be modeled by multi-tank systems. Such processes involve substances that flow about or change into other substances at rates based solely on the amount of each process substance that is present in the system at any given time. In the jargon of simulation, these models are called compartmental models and are found in fields as diverse as chemistry (where the substances are chemical concentrations and the flows represent reaction rates), economics (where the substances represent money and the flows represent sales rates, payment rates, etc.), and planning (where the substances are resources and the flows represent production and consumption rates). This chapter introduces the concepts of qualitative reasoning and the QSIM system and then defines qualitative models for simple tank systems. It then uses these methods to model an important process in the field of vacuum engineering.

2.2 The QSIM System

Mathematically, compartmental models can be described by systems of first order autonomous ordinary differential equations. In this section the models will be represented qualitatively using the QSIM system [Kuipers, 1986]. QSIM models and mathematical models are similar in that they both represent a physical system in terms of variables and constraints. In the mathematical model, the constraints are described by equations and the variables can take on any value over the real numbers. In the qualitative model, the constraints are described by a more abstract set of arithmetic, differential, and monotonic function relations, and the variables range over an ordered set of landmark values (and intervals) which represent qualitatively meaningful points on the extended real number line. The qualitative model is therefore an abstraction of the mathematical model and can be seen as representing a set of differential equation systems. This many-to-one abstraction is the key to the ability of a qualitative model to represent incomplete knowledge.

2.2.1 Elements of a QSIM model

A QSIM model is composed of the following elements :

- A list of model variables and their quantity spaces. A variable may be either a state variable, a constant, or a variable derived from state variables and constants. The quantity space is an ordered list of qualitatively meaningful points on the real number line. For instance, in a model of a container, the meaningful points for the variable *level* might be zero (marking where the container is empty) and TOP (marking where the container is full). At any time during the simulation, a variable will either be at a landmark or between two landmarks. The location of the variable in its qspace is referred to as the qualitative magnitude or *qmag* of the variable. In addition, the direction of change (or *qdir*) of each variable is also tracked. The qdir can take on the values increasing, decreasing, or steady. Together, the qmag and qdir form a structure called a *qval* which describes the state of a variable and its derivative at a given time.

- A list of the constraints on model variables. These constraints correspond to the equations in a mathematical model. Some express simple mathematical operators such as *mult(mass, accel, force)*, *add(netflow, inflow, outflow)*, and *d/dt(position, velocity)*. Others represent abstract functional relationships such as monotonically increasing ($M^+(depth, waterpressure)$) or decreasing ($M^-(altitude, airpressure)$). Constraints also optionally contain sets of *corresponding values* which further constrain the allowable values of the variables in the constraint. For instance, in any container filled with liquid, the pressure at the bottom of the container is a monotonically increasing function of liquid level. Also, a liquid level of zero corresponds to a pressure of zero. In QSIM, this is modeled as $M^+(level, pressure)$ with corresponding value $(0, 0)$.

- An initial state description. Normally, this will specify the initial value of all the state variables of the system. If the initial state is underspecified, however, QSIM will simply generate all possible initial states that are consistent with the given initial state description.

The qspace and constraint lists together form a structure called a qualitative differential equation or *QDE*.

2.2.2 The QSIM algorithm

The QSIM algorithm begins by generating a complete[1] initial state (or states) for the given QDE and initial state description and then uses a set of transition rules and filters to determine the subsequent states. The filters eliminate inconsistent states as well as detect quiescent states and cycles in a behavior. The end result of the simulation is a set of behaviors for that model. QSIM is guaranteed to find all real behaviors of a system, although it may generate impossible ones as well [Kuipers, 1986]. Additionally, each behavior description has the property that the signs of all model variables are uniform over any time interval. Each behavior can be graphically displayed using tools within the QSIM system. The plots show the qvals of the variables of the model at each distinguished timepoint[2] and at the intervals between the timepoints.

2.3 Simple Tank Models

2.3.1 The Bathtub

The simplest compartmental system is the one-tank system (or "bathtub"). In compartmental terms, the process substance is the water in the tub and the flows are the inflow from a tap and the outflow through the drain (see Figure 2.1). The differential equation that determines the amount of water in the tub is :

$$A' = inflow - outflow(A)$$

[1]A state is *complete* if every variable in the QDE has a qval (i.e., qmag and qdir).

[2]Time is considered to be an implicit variable of every QDE whose qspace consists of a set of *distinguished timepoints*. These points are generated by QSIM based on the activity of the model variables.

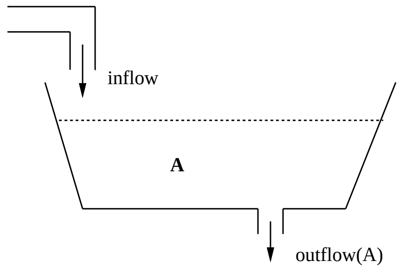
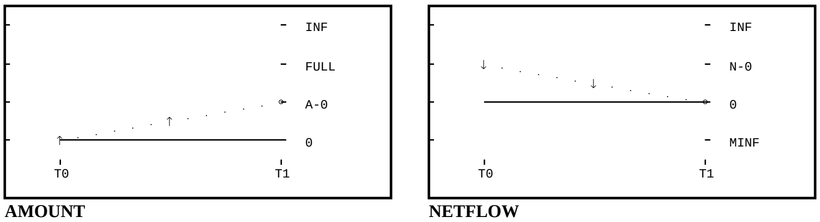


Figure 2.1: The bathtub.

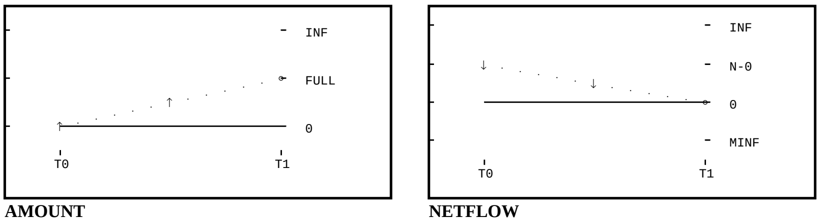
```
(define-QDE Bathtub
  (text "Simple bathtub model.")
  (independent inflow)
  (quantity-spaces
    (amount    (0 FULL inf))
    (outflow   (0 inf))
    (inflow    (0 if* inf))
    (netflow   (minf 0 inf)))
  (constraints
    ((M+ amount outflow)    (0 0) (inf inf))
    ((ADD netflow outflow inflow))
    ((D/DT amount netflow))))
```

Figure 2.2: Qualitative model of the bathtub.

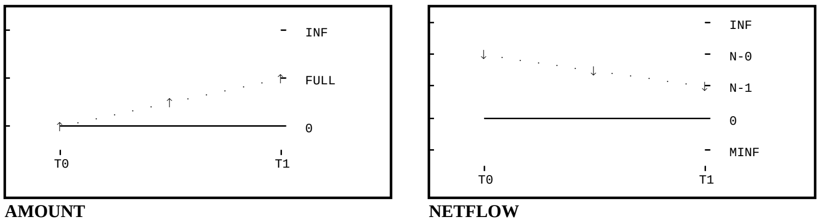
where $outflow(A)$ is a function of the amount of water in the tank and $inflow$ is a constant. In cases where the function between amount and outflow is actually known, this equation can be solved analytically. The qualitative model for the bathtub is shown in Figure 2.2. Note that the outflow is described as a monotonically increasing function of the amount of water in the tank. This constraint stands for a set of possible monotonically increasing functions of A (including non-linear ones). This model therefore describes *any* bathtub regardless of the shape of the tub or the size of the drain. It is therefore more general than a numerical model which must pick a specific function for $outflow$ and a specific value for $inflow$.



(a) $netflow = 0$ before tub fills



(b) $netflow = 0$ exactly when tub is full



(c) Tub overflows

Figure 2.3: The three behaviors of the bathtub.

Figure 2.3 shows the result of simulating the bathtub model with the initial condition $A = 0$ and $inflow = IF*$. The model predicts three behaviors corresponding to $netflow$ reaching equilibrium either before the tank becomes full, exactly when it becomes full, or, never (i.e. the tank overflows). Multiple behaviors result due to uncertainty in the function $outflow$ and its relation to the qmag $IF*$.

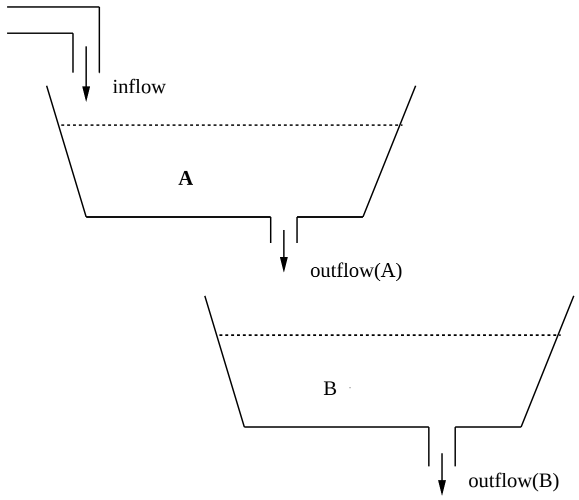


Figure 2.4: Two cascaded tanks.

2.3.2 The Two-tank Cascade

A more complex system of tanks is the cascade. Here, one tank is a bathtub and the second tank is placed under the first one so that its inflow is the outflow of the first one (see Figure 2.4). The equations for this system are

$$\begin{aligned} A' &= inflow - outflowA(A) \\ B' &= outflowA(A) - outflowB(B) \end{aligned}$$

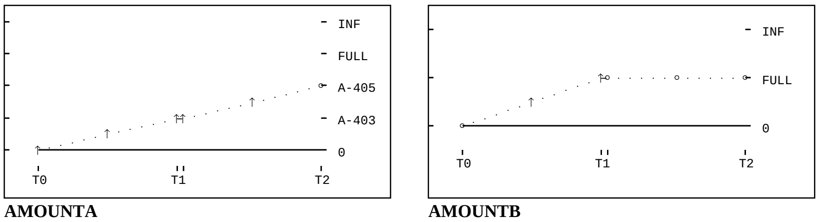
In general, these equations cannot be solved analytically and numerical simulation is needed to determine the state of the system over time. Since we would like the qualitative model to predict all possible behaviors of the system, it must include the cases where either tank overflows. It is assumed that when either tank overflows, the amount of water in the tank remains constant at the maximum amount of the tank (the remainder dripping over the side). Therefore, when one tank overflows, the two-tank cascade becomes a simple bathtub and the model of the system must change. QSIM permits such model

```
(define-QDE Cascaded-tanks
  (text "Two cascaded tanks")
  (quantity-spaces
    (inflowa    (0 ifa* inf))
    (amounta    (0 full inf))
    (outflowa   (0 inf))
    (netflowa   (minf 0 inf))
    (amountb    (0 full inf))
    (outflowb   (0 inf))
    (netflowb   (minf 0 inf)))
  (constraints
    ((M+ amounta outflowa)            (0 0) (inf inf)   )
    ((ADD outflowa netflowa inflowa)                    )
    ((d/dt amounta netflowa))
    ((m+ amountb outflowb)            (0 0) (inf inf)   )
    ((add outflowb netflowb outflowa)                   )
    ((d/dt amountb netflowb)))
  (independent inflowa)
  (transitions
    ((amounta (FULL inc)) a-overflow)
    ((amountb (FULL inc)) b-overflow))
  (other
    (ignore-qdirs netflowb)))
```

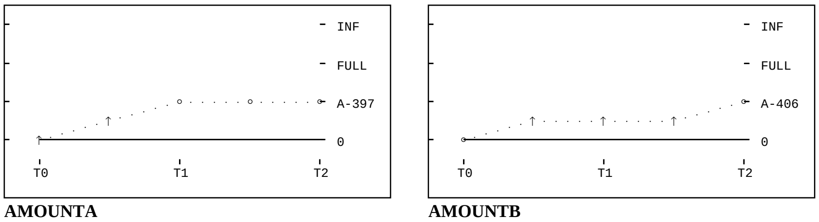
Figure 2.5: Qualitative model of the two-tank cascade.

transitions via *operating region transitions*. A region transition states when to switch to a new QDE based on the qvals of variables in the current QDE. It is defined by a `transitions` clause in the QDE which is a list of a variable, a qval, and a function that evaluates to the initial state of the QDE that governs the new operating region of the system. Figure 2.5 describes the initial QDE for the system. In this case, the function `a-overflow` creates the first state of the operating region where the system may be described as a constantly full A tank and a non-full B tank that acts like a bathtub. The function `b-overflow` is defined similarly with respect to the B tank. For the initial conditions $A = 0$, $B = 0$, and $inflow = IF*$, QSIM predicts a huge number of behaviors. The problem is that under the rules of qualitative arithmetic, the qdir for variable $netflowB$ is unconstrained by the QDE when both $outflowA$ and $outflowB$ are changing in the same direction. In this case, the QSIM algorithm will generate behaviors for all possible qdirs of the variable, a situation known as "chatter." There are two classes of methods for eliminating chatter in QSIM. One is to use information about the higher order derivatives of the model by algebraically computing an expression for the qdirs of the chattering variables [Kuipers and Chiu, 1987] [Dalle Molle, 1989]. With the second method, the qdir of certain variables are ignored, resulting in multiple behaviors collapsing into a single one [Kuipers and Chiu, 1987]. The former method is more desirable since qdirs are maintained for all variables. Unfortunately, it is not the case that the necessary algebraic information can be extracted from all models. In these cases, only the method of ignoring qdirs will work.

Using the "ignore-qdirs" method on $netflowB$, the model still predicts twenty behaviors. This large number of behaviors is due to uncertainty in the relative outflow rates and capacities of the two tanks. QSIM generates behaviors for all possible relations between these quantities, thereby guaranteeing that all possible behaviors of the system are simulated. Figure 2.6 shows two of these behaviors. The first shows tank A reaching equilibrium before tank B and the second shows the opposite case, with B overflowing. As more and more tanks are added to the model, the qualitative simulation becomes increasingly intractable and requires more sophisticated methods for eliminating spurious behaviors [Dalle Molle, 1989].



(a) B fills completely before A reaches equilibrium



(b) A reaches equilibrium before B.

Figure 2.6: Two of the twenty behaviors of the cascaded tanks.

2.3.3 The U-tube

A variation of the two-tank cascade is the u-tube. In this system, the two tanks are connected in parallel so that there is a crossflow between them (see Figure 2.7). The equations for this system are

$$\begin{aligned} A' &= inflowA - flow(A, B) \\ B' &= flow(A, B) - outflowB(B) \end{aligned}$$

The qualitative model for this system is shown in Figure 2.8. This model exhibits chatter on *netflowA, netflowB, amtdiff*, and *flowAB*. After ignoring qdirs on these variables, the model predicts one behavior for the initial condition $A = 0$, $B = 0$, and $inflow = IF*$ (see Figure 2.9).

The compartmental models defined in this section can also be represented as in Figure 2.10 where each flow is described by a fractional *transfer*

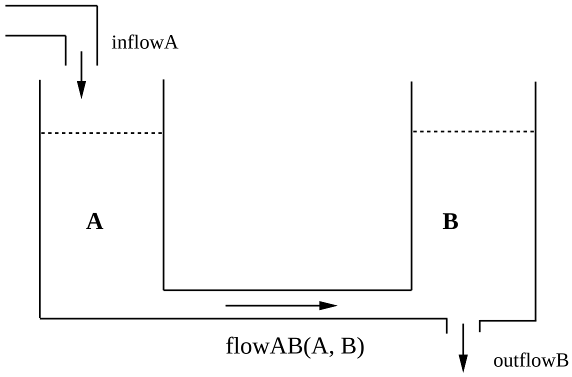


Figure 2.7: The u-tube.

```
(define-QDE U-tube
  (text "Simple U-tube")
  (quantity-spaces
    (inflowa    (0 ifa* inf))
    (amounta    (0 inf))
    (netflowa   (minf 0 inf))
    (amountb    (0  inf))
    (netflowb   (minf 0 inf))
    (outflowb   (0 inf))
    (amtdiff    (minf 0 inf))
    (flowAB     (minf 0 inf)))
  (constraints
    ((D/DT amounta netflowa))
    ((D/DT amountb netflowb))
    ((ADD amountb amtdiff amounta))
    ((M+ amtdiff flowAB)            (0 0) (inf inf) )
    ((M+ amountB outflowb)          (0 0) (inf inf) )
    ((ADD netflowa flowAB inflowa))
    ((ADD netflowb outflowb flowAB) (0 0 0)))
  (independent inflowa)
  (other
    (ignore-qdirs netflowa netflowb amtdiff flowAB)))
```

Figure 2.8: Qualitative model of the u-tube.

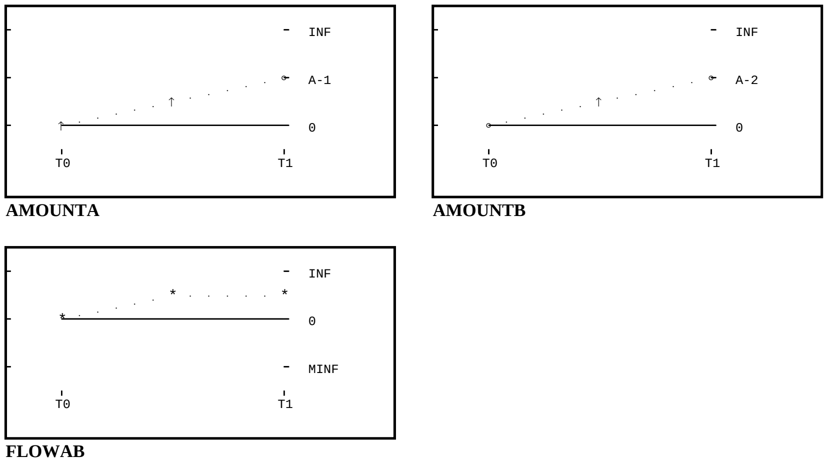


Figure 2.9: The one behavior of the u-tube.

coefficient that represents the portion of the compartment that is transferred per unit time [Jacquez, 1985]. This approach is used in traditional compartmental analysis. Note that these flows are represented with monotonic function constraints in QSIM.

In the next section, we shall see that these apparently simple compartmental models can be used to model industrially significant systems.

2.4 Vacuum Systems

2.4.1 Motivation

The production of high vacuum is of great importance to semiconductor fabrication as many of the steps (such as sputtering and molecular beam epitaxy) cannot be performed if there are foreign particles in the process chamber.

Unfortunately, creating such ultra-high vacua can be expensive and

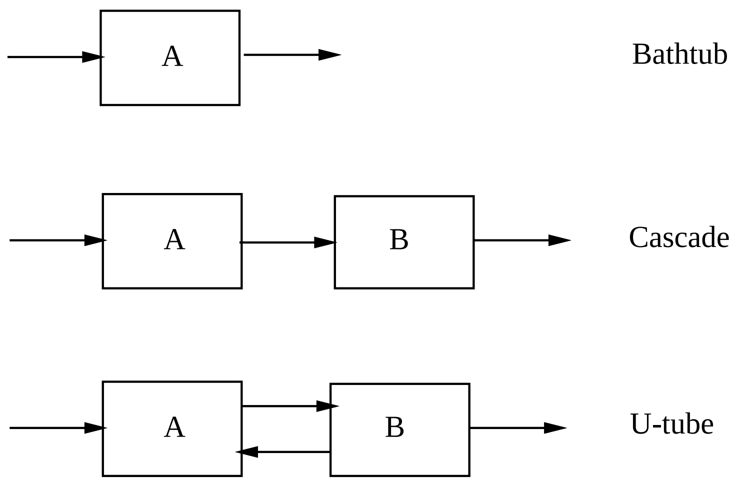


Figure 2.10: An alternate representation for the models in this section.

time-consuming. To reach ultimate pressures of 10^{-9} torr[3] can take several hours and something as innocuous as a fingerprint left on the chamber during servicing can cause a huge performance loss.

Because of this risk, it is important to service vacuum equipment only when there is a problem. This suggests a need for a monitoring system that can detect when the system goes out of tolerance. Of particular importance is the time during which the chamber is being pumped down from atmospheric pressure. If failures during this fairly short period (around 15 to 30 minutes) can be detected, much time and expense can be avoided. The initial pumpdown phase of operation was chosen as the focus of this research.

The difficulty in modeling this process numerically is that there is no practical theory for the sorption of gases. Therefore, any useful model must deal with uncertainties in the underlying modeling assumptions. Qualitative

[3] 1 Torr is equal to 1000 microns. See Appendix A for pressure unit definitions

modeling permits reasoning with these types of uncertainties.

2.4.2 Models

The pumpdown process is intuitively very simple. A chamber at room pressure initially contains some amount of gas. A pump, which can displace a certain amount of gas per unit time and pressure, removes gas from the chamber, hence lowering the pressure. For a simple vacuum pump, this process will continue until the pump reaches its cutoff pressure at which point the minimum pressure within the pump is the same as the pressure within the chamber. This simple system is similar to a full bathtub with no inflow whose drain is mounted on the side of the tub so that some water will always remain.

For pumps that operate in the high vacuum range (between 1 and 0.01 microns), there are additional effects to consider. The most significant of these is that of "outgassing" – a process where gas initially present in the walls of the chamber desorbs[4] and thereby increases the chamber pressure. This effect becomes more pronounced at pressures below 70 microns [LaPelle, 1972].

At the very-high (10^{-3} to 10^{-5} microns) and ultra-high (below 10^{-6} microns) vacuum ranges, there are also nontrivial pressure effects due to the release of gas dissolved in the chamber walls as well as the permeation of gas through the walls from the outside environment.

The actual vacuum system that was modeled in this thesis is de-

[4] *Desorption* is the process by which gases trapped in a substance are released. The reverse process is called *adsorption*. Adsorption is different from absorption in that the gases do not dissolve into the substance; they simply "stick" to its surface.

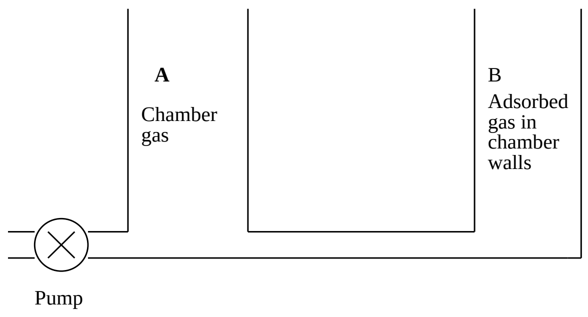


Figure 2.11: Basic pumpdown model of a vacuum system.

scribed in [Duval, 1988]. This system was chosen because there existed real pressure data for it and this data was needed to perform the process monitoring tests in Chapter 4. Unfortunately, there was no data for faulted versions of the system, so numerical models of both the healthy and faulted models were constructed to provide this data. As a check, the numerical model of the healthy system was compared to the real data from [Duval, 1988]. The details of this comparison are discussed in the next section.

2.4.3 The Basic Model

For the modeling problem analyzed here, only outgassing effects were considered since this is the most prevalent effect during the pumpdown of a high-vacuum system. Because there are two locations where gas is present in the sorption model, the process was modeled as a u-tube (see Figure 2.11). The 'A' side of the tube represents the gas in the chamber. The 'B' side represents the gas adsorbed in the walls.

The Sorption Model The process of desorption is not very well understood. There have been a number of studies of its physical nature, the most well-known of which is that of Langmuir [Dushman and Lafferty, 1962]

[Roth, 1976] [Lewen, 1965]. In this theory, it is assumed that the adsorbed gas forms a single layer on the chamber walls. The number of gas molecules adsorbed per unit area is

$$\frac{adsorbrate}{area} = f(p) \cdot \alpha \tag{2.1}$$

$$f(p) = 3.5 \times 10^{19}[P/\sqrt{MT}] \tag{2.2}$$

where $f(p)$ is the number of molecules hitting a unit area of chamber surface, p is the chamber pressure in microns, M is the molecular mass of the gas, T is the temperature of the gas in degrees Kelvin, and *adsorbrate* is in units of molecules/(cm^2·sec). The variable α is the "sticking coefficient" or probability that an incident molecule will be adsorbed. It is a function of another variable θ which measures the fraction of molecular sites on the surface that contain adsorbed molecules. There does not exist an exact equation for α, although empirical studies have shown that its value depends on both the type of gas and the composition of the surface and that it is a monotonically decreasing function of θ [Roth, 1976].

The rate at which molecules desorb from the surface is given by

$$desorbrate = B/t_s \tag{2.3}$$

where t_s is the average amount of time that an adsorbed molecule spends at a particular molecular site on the wall (this is also known as the *sojourn time*). This time has been shown to be [Roth, 1976]

$$t_s = t' \exp E_D/(R_0 T)$$

where t' is the period of oscillation of the molecule normal to the surface (approximately 10^{-13} seconds), E_D is the energy of desorption in kg-cal/mole and R_0 is the gas constant in kg-cal/degree (approximately .002). At atmospheric

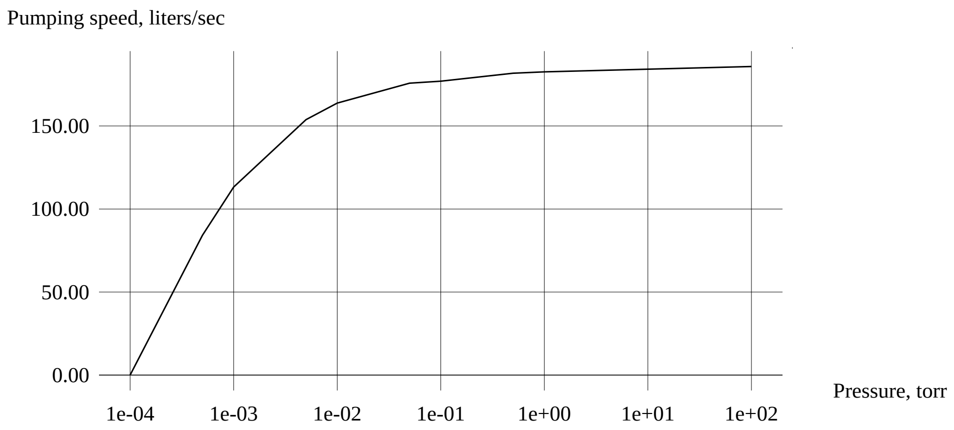


Figure 2.12: Typical pump speed curve (from [Van Atta, 1965] p. 176).

pressure, the adsorption rate of the gas is equal to the desorption rate, however, as the pressure is lowered, there is a net flow of gas out of the chamber walls.

The Pump Model Vacuum systems can use a variety of pump types to evacuate the gas from the chamber. We have included a two-stage mechanical pump because this pump was used in the "benchmark" system from which our raw pressure data was gathered. Vacuum pump performance is measured by speed expressed in liters/minute and is a function of chamber pressure. A representative speed curve is shown in Figure 2.12. The pressure where the speed drops to zero is called the *ultimate pressure* of the pump and represents the point at which internal leaks due to mechanical clearances within the pump prevent it from further reducing the chamber pressure. The throughput of the pump is given by

$$pumpthroughput = pumpspeed \times pressure \times c \tag{2.4}$$

where *pumpspeed* is in liters/minute, *pressure* is in microns, and *pumpthroughput* is in molecules/minute. The constant c converts micron-liters/-minute into molecules and is equal to $N_a/(RT)$ where N_a is Avagadro's number. Putting these equations together gives the basic model

$$
\begin{aligned}
A' &= -flowAB(A,B) - pumpthroughput(A) \\
B' &= flowAB(A,B) \\
flowAB &= adsorbrate(A,B) - desorbrate(B)
\end{aligned}
$$

where *pumpthroughput* is defined in Equation 2.4, *adsorbrate* is described by Equation 2.1, and *desorbrate* is defined in Equation 2.3.

The Qualitative Model Figure 2.13 shows the qualitative model for the basic vacuum system. Note the use of monotonic function constraints for α, f, pumpspeed, and desorbrate[5]. The qdirs of dA, $flowAB$, $mflowAB$, and *f*alpha* were ignored to eliminate chatter[6]. Figure 2.14 shows the qualitative values of A, B, and $flowAB$ for this system. There is a single behavior corresponding to the system reaching the ultimate pressure *poff* of the pump.

The Numerical Model A numerical model for the basic system was developed as both a check against the qualitative model and as a method for generating test data for the process monitoring test (see Chapter 4). The model was constructed using the equations described above with "reason-

[5]Note that f is defined as $adsorbrate/\alpha$.

[6]By ignoring the qdirs on these variables, our simulation no longer gives detailed predictions for dA, $flowAB$, $mflowAB$, and *f*alpha*. However, since we are primarily interested with the detailed behavior of A and B only, this loss of information is of no consequence.

```
(define-QDE vac-sys
  (text "A vacuum system.")
  (quantity-spaces
      (p         (0 poff proom))  ; chamber pressure
      (A         (0 Aoff Aroom))  ; # adsorbed molec in chamber
      (dA        (minf 0 inf))    ; netflow out of chamber
      (B         (0 bmax))        ; # molec in chamber walls
      (area      (0 area0))       ; chamber surface area
      (theta     (0 one))         ; fraction of molec sites filled
      (pumptp    (0 inf))         ; pump throughput
      (pumptp/mq (0 inf))         ; throughput / mq
      (mq        (0 mq0))         ; # molec / micron-liter
      (speed     (0 speed0))      ; pump speed
      (flowAB    (minf 0 inf))    ; gas flow A->B
      (mflowAB   (minf 0 inf))    ; gas flow B->A
      (alpha     (0 alpha0 alphamax)) ; sticking function
      (f*alpha   (0 f0*alpha0 inf))   ; adsorbrate/unit area
      (f         (0 f0))          ; # molec incident on walls
      (desorbrt  (0 g0 inf))      ; desorbrate
      (adsorbrt  (0 f0*alpha0*a inf)) ; adsorbrate
      (m         (0 m0))          ; # molecules / micron
      (bmax      (0 bmax)))       ; initial # molec adsorbed
  (constraints
      ((D/DT A dA))
      ((D/DT B flowAB))
      ((MULT theta bmax B) (one bmax bmax))
      ((MULT speed p pumptp/Mq))
      ((MULT pumptp/Mq Mq pumptp))
      ((M+ p speed) (poff 0) (proom speed0))
      ((ADD pumptp dA mflowAB))
      ((MINUS mflowAB flowAB))
      ((MULT m A p) (m0 Aoff poff) (m0 Aroom Proom))
      ((M+ p f) (0 0) (proom f0))
      ((MULT f alpha f*alpha) (f0 alpha0 f0*alpha0))
      ((mult f*alpha area adsorbrt) (f0*alpha0 area0 f0*alpha0*a))
      ((add desorbrt flowAB adsorbrt) (g0 0 f0*alpha0*a))
      ((M+ B desorbrt) (0 0) (Bmax g0))
      ((M- theta alpha) (0 alphamax) (one alpha0)))
  (independent m mq bmax area)
  (other (ignore-qdirs dA flowAB mflowAB f*alpha adsorbrt)))
```

Figure 2.13: Qualitative model for the basic vacuum system.

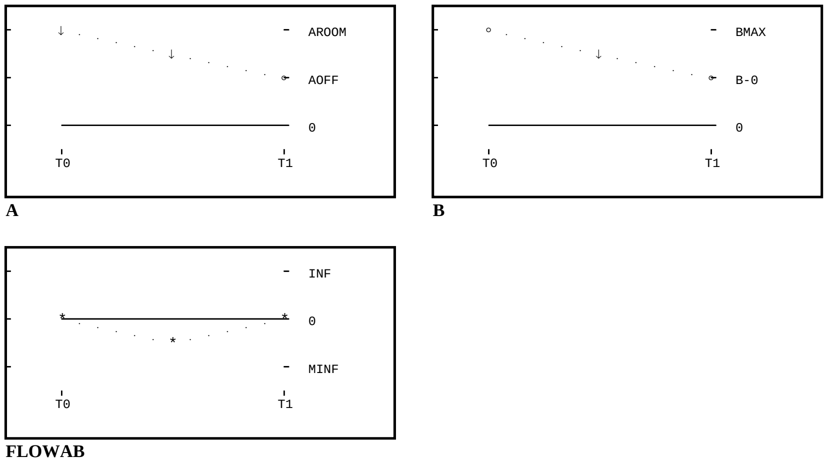


Figure 2.14: The one behavior of the basic vacuum system.

able" approximation for α, *pumpspeed*, and the initial value of B. It was compared against a set of real data described in [Duval, 1988] for accuracy. The real data was collected from a system with a 90 liter chamber and a 250 liters/minute maximum speed pump. From this, a piecewise linear approximation of the pump speed curve was made. A chamber surface area of 13800 cm^2 was calculated by assuming the 90 liter chamber was a cube roughly $100 \times 30 \times 30$ cm. The initial value of B was empirically chosen to be approximately 10^{21}. This value assumes that there is a concentration of approximately 10^{17} molecules per unit area which is approximately 2 orders of magnitude above the number of molecules that could normally fit in a monolayer. While this violates the monolayer assumption, it is more in line with physical evidence [Dushman and Lafferty, 1962]. The effect of this violation on the monolayer theory is that the sticking function α must be adjusted so that at atmospheric pressure, the adsorption rate balances the desorbtion rate. The equation for α was therefore defined to be an exponential function of chamber area and θ such that this equilibrium balance was maintained.

The equation was

$$
\begin{aligned}
\alpha &= \exp\left(\ln\left(c_a\right)\theta\right) \\
c_a &= B(0) \cdot \frac{\sqrt{MT}}{3.5 \times 10^{19} \cdot t_s \cdot area \cdot proom}
\end{aligned}
$$

The complete model is described in Appendix C. Figure 2.15 shows the results of the numerical simulation. Overall, it predicts the same general trends as the qualitative model, with pressure decreasing over time and flow decreasing to a point and then increasing back to zero. This qualitative similarity is an indication that the numerical model is a member of the class of ODEs that the QDE model represents.

One could ask why the qualitative model is needed if the numerical model gives similar results. To answer this, note that while the numerical model behaves similarly to the real vacuum system, it does not behave identically. In the modeling process, uncertainty has crept into the model by way of simplifying assumptions about functional relationships and parameters. This uncertainty is not represented in the numerical model and so its predictions, while very precise, may not correctly portray the behavior of the real system. The qualitative model also contains uncertainty, but it is explicitly represented in the model and its predictions. The qualitative model is therefore more accurate than the numerical model in that the true behavior of the system is predicted, although with less precision. Since our monitoring and diagnosis strategy relies primarily on accurate predictions, the qualitative model is preferred over the numerical model.

Figure 2.16 shows the correlation between this model and the raw data described in [Duval, 1988]. The numerical model matches the data quite well for both low pressures, and, when extrapolated back to the initial pressure, high pressures. The middle pressure range is where the correlation is

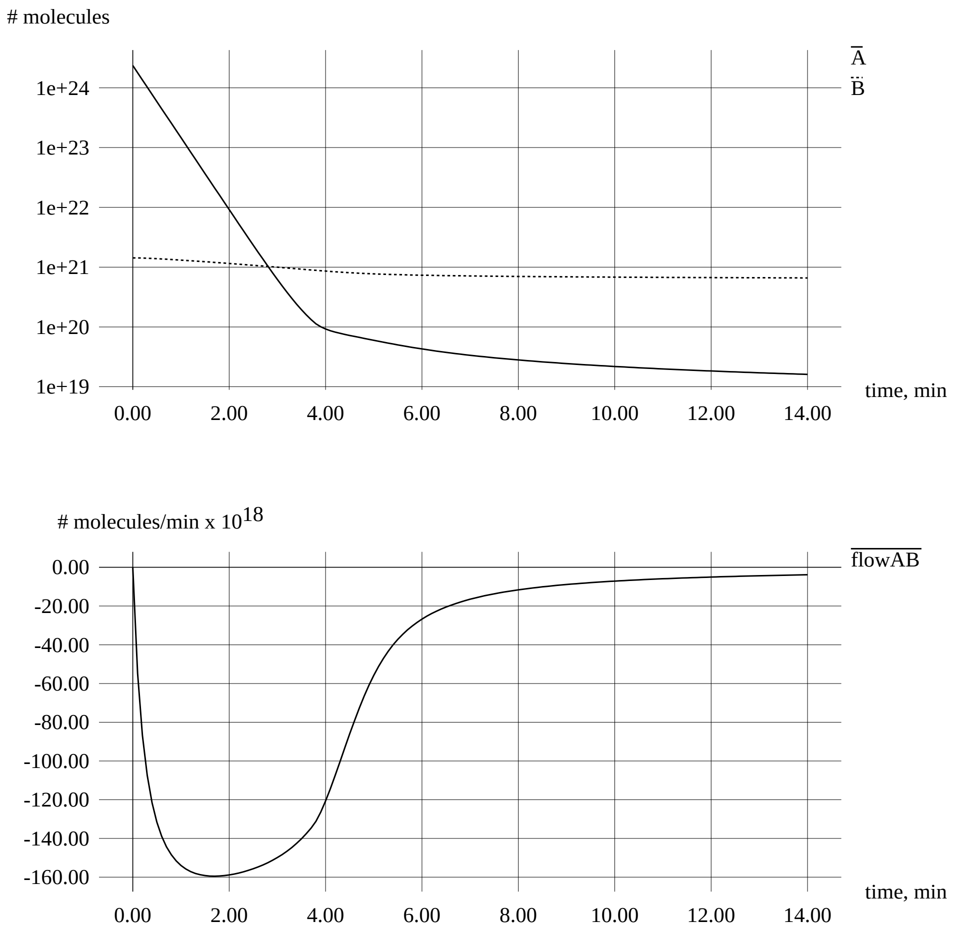


Figure 2.15: The numerical simulation of the basic vacuum system. The number of molecules in the chamber and wall decreases continuously with time. The flow of gas is always out of the chamber walls and is initially very fast. After two seconds, it begins to slow down until equilibrium is reached.

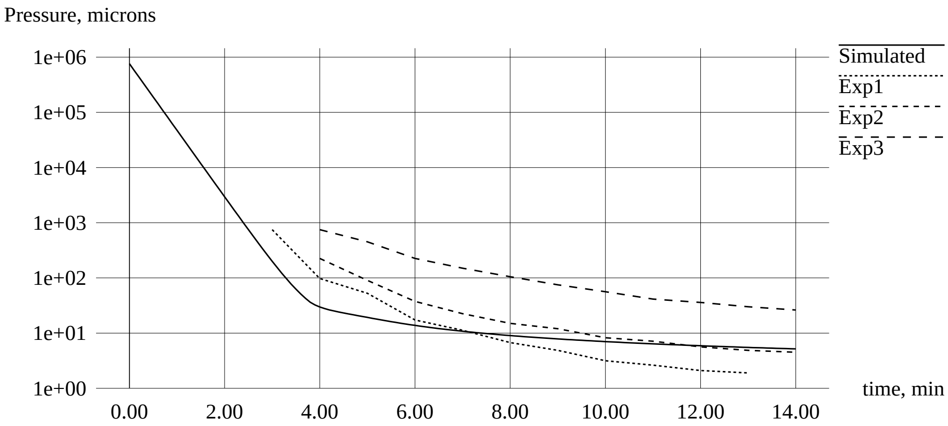


Figure 2.16: Comparison of simulated and experimental data (from [Duval, 1988] p. 203).

weakest, in part because of the simplified pump speed curve which is overly high for midrange pressures.

2.4.4 Fault Models

Real Leak There are a number of different failures that can affect the pumpdown curves. The most important of these is a leak in the chamber. The leak was modeled as a gasket leak as described in [Roth, 1976]. The leak is caused by channels that form between the two surfaces that form the seal. Each leak path can be viewed as a prismatic channel of height A_R and length w that runs through the gasket material (see Figure 2.17). The actual height of A_R depends on the roughness of the surfaces. Values of A_R for typical surfaces used in vacuum systems range from 10^{-3} cm for roughly ground seals up to 10^{-6} for highly polished surfaces. Seal widths also vary from 0.1 to 1 cm depending on the type of piping used. Assuming that the number of leaks per unit length is a constant described by the surface roughness (as is the

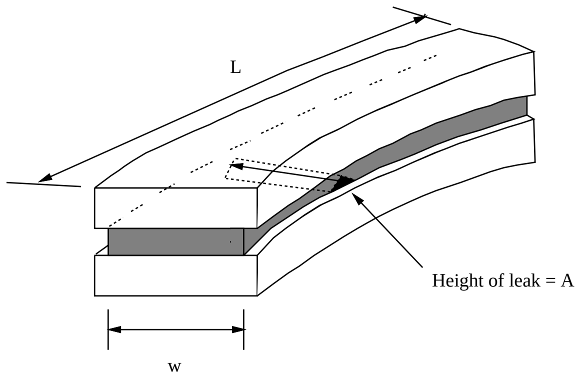


Figure 2.17: Leak model derivation (from [Roth, 1976], p. 386).

case with any machined surface), the total leak conductance[7] around a seal of width w cm and total perimeter L cm is

$$C_{leak} = [34A_R^2/w]L$$

The leak rate is then

$$leakrate = 60 \cdot C_{leak} \cdot (p_{room} - p) \cdot c$$

where $proom$ is atmospheric pressure in microns, p is chamber pressure in microns, c is a constant that converts from micron-liters to molecules and $leakrate$ is in molecules/second. This is then added to the equation for A to yield

$$A' = -flowAB(A, B) - pumpthroughput(A) + leakrate(A)$$

Qualitatively, $leakrate$ is modeled as a monotonically increasing function of the pressure difference between the chamber pressure and atmospheric pressure. Figure 2.19b shows the qualitative behavior for this system.

[7] *Conductance* measures the ability of a pipe or orifice to permit gas flow. Conductance is measured in liters/sec.

Once again, there is only one behavior corresponding to the case where the system reaches equilibrium, although now this point is above the ultimate pressure of the pump.

For the numerical model, a value of 10^{-3} was chosen for A (corresponding to a roughly ground seal), 5π was chosen for L (corresponding to a 5 cm diameter pipe), and 1 mm was chosen for w. Figure 2.20b shows the numerical plot for this model. Notice that the shape of the "A" curve is the same as in Figure 2.15 except that the curve is settling at a higher equilibrium value.

Virtual Leak A second important fault is the virtual leak. A virtual leak is an excess gas load caused by the presence of air or liquid trapped in the walls of the chamber that slowly escapes as the chamber pressure drops. This foreign material may be present because of improper maintenance (not using screws with a pressure-release mechanism in the chamber walls) or contamination (a fingerprint, for example). A virtual leak differs from a real leak in that there is a finite amount of gas which can leak into the chamber, so theoretically it can all be pumped out. However, it could take so long to pump it all out that it would be impractical to wait so long. The virtual leak can be modeled as another finite gas load in series with the chamber gas load and the adsorbed gas load. This configuration can be modeled by adding another tube to the basic model that represents the amount of gas contained in the leak (see Figure 2.18). This leak can also be modeled as a gasket leak in which case the equation for *leakrate* (see page 27) holds except that the pressure difference is now between the chamber and the leak volume. This change, and the fact that the leak has a finite volume, changes the overall structure of the equations. A new state variable must be added for the gas contained in the virtual leak.

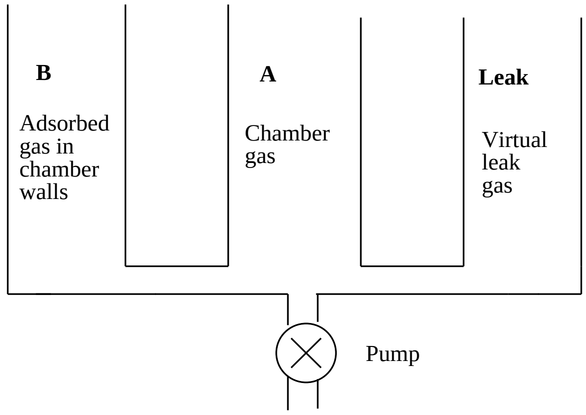


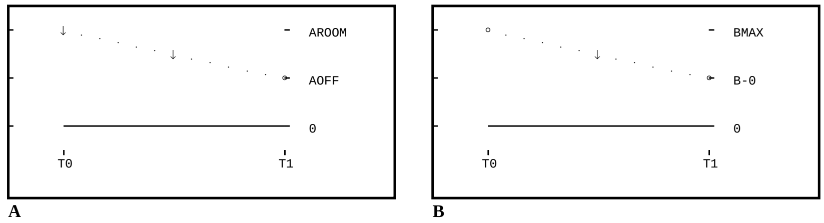
Figure 2.18: Virtual leak model.

The set of equations becomes

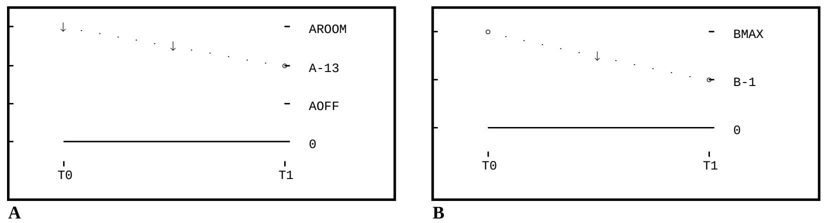
$$\begin{aligned} A' &= -flowAB(A,B) - pumpthroughput(A) + Leak' \\ B' &= flowAB(A,B) \\ Leak' &= leakrate(A, Leak) \\ flowAB &= adsorbrate(A,B) - desorbrate(B) \end{aligned}$$

Qualitatively, *leakrate* is modeled as a monotonically increasing function of the pressure difference between the chamber and the leak volume. Figure 2.19c shows the qualitative behavior of this system. Once again, there is only one behavior corresponding to the case where the system reaches equilibrium at the ultimate pressure of the pump. Note that this single behavior is qualitatively identical to the behavior of the basic model. This result is reasonable since at $t = \infty$ both systems reach the ultimate pressure of the pump.

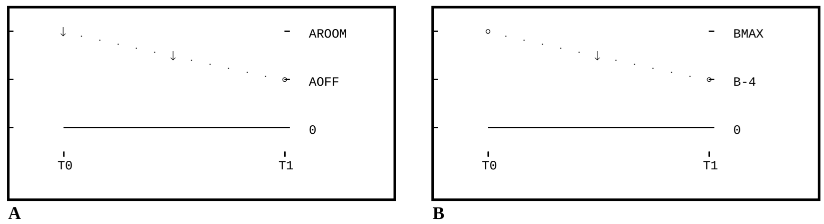
For the numerical model, the case where the leak forms behind a screw in the chamber wall was considered and a value of 10^{-4} was chosen



(a) Basic model



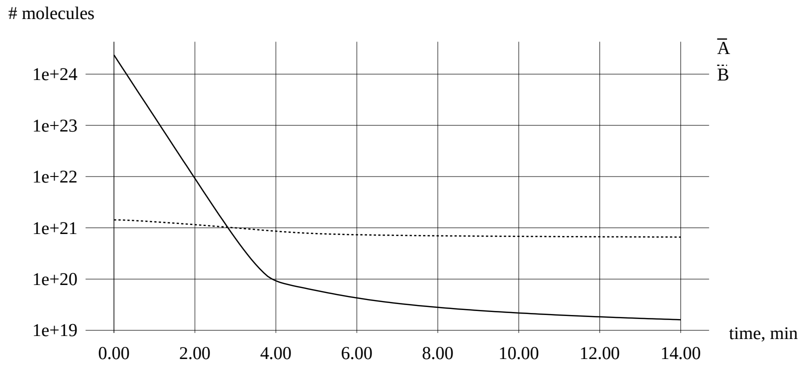
(b) Real leak



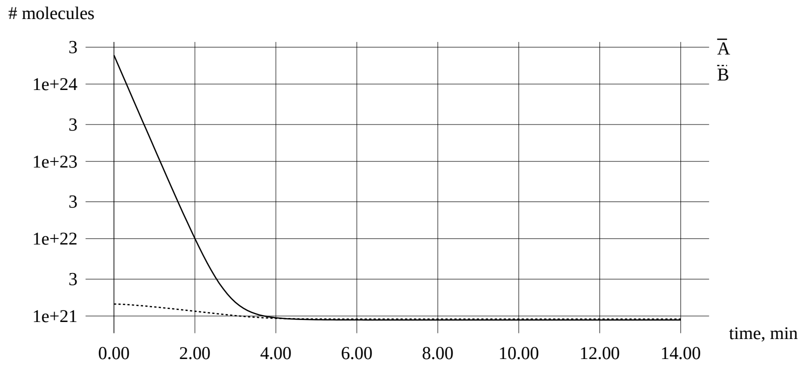
(c) Virtual leak

Figure 2.19: Qualitative simulation results for the vacuum models.

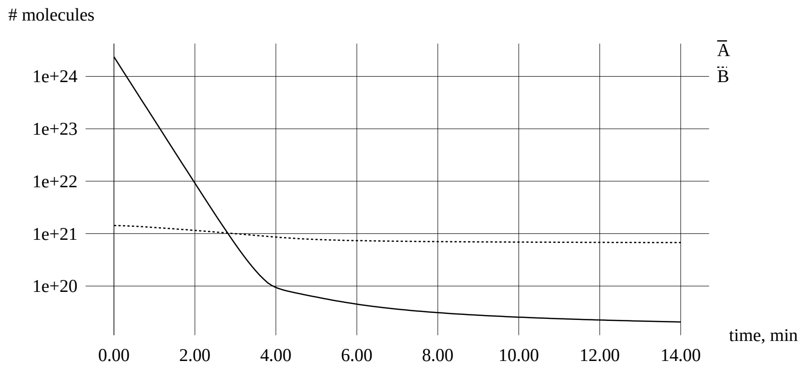
for A_R (corresponding to a well-ground seal that the screw forms with the chamber wall), 1.5π cm was chosen for L, and 1 mm was chosen for w. A volume of 10×0.001 liters was chosen, corresponding to the volume of 10 screws with the virtual leak problem. Figure 2.20c shows the numerical plot for this model. Figures 2.19 and 2.20 contrast the three models. Notice that the magnitude of the "A" variable is almost identical to the "A" variable in the basic system. This is because at the pressures being considered, the virtual leak has a very small effect on the chamber pressure of the system.



(a) Basic model



(b) Real leak



(c) Virtual leak

Figure 2.20: Numerical simulation results for the vacuum models.

2.4.5 Assumptions affecting the models

Several assumptions and simplifications were made both because of incomplete data as well as to simplify simulation. Some of these have been stated earlier and are repeated here for clarity.

- Because of the "stiffness" of the problem and the use of a fourth-order Runge-Kutta algorithm [Forsythe *et al.*, 1977] for numerical simulation, a simulation of pumpdown to the ultra-high vacuum range proved problematic. Instead, the pumpdown of a vessel to the high-vacuum range was modeled. Data for this process came from [Duval, 1988]. Figure 2.16 is a plot of the actual data as well as the numerical results for the basic model using Runge-Kutta. The divergence of the simulated data from the real data at the midrange can be explained by the simplified model of the pump speed curve used (see the next assumption).

- The model assumes a 2-stage mechanical pump. This model was further simplified so that a two-piece linear function could represent the speed curve with respect to pressure. This type of pump was used to generate the experimental data.

- The models do not differentiate between gas types. This simplification was made because desorption of water vapor is the major pressure load in the high-vacuum range. For the ultra-high range, the partial pressures of individual gases play a larger role.

- The only independent variable that was allowed to vary was the chamber surface area. In particular, it was assumed that chamber volume, particle mass, and energy of desorption are accurately known constants.

Chamber surface area is probably the most unknown (and widely ranging) variable because it depends on the surface condition of the chamber [Roth, 1976] which changes over time. Chamber volume can be accurately estimated, and average molecular mass can be used to approximate the true molecular mass without effecting the equations greatly. The same is true for values of energy of desorption [Lewen, 1965], so the key contributor to uncertainty is the chamber surface area.

2.5 Using Partial Quantitative Knowledge in Qualitative Models

The qualitative models in sections 2.3 and 2.4 correctly predict the behaviors of processes that they describe. Their predictions, however, are rather weak. By abstracting the bathtub model so that outflow is simply a monotonic function of amount, some rather unlikely shaped bathtubs are considered. Suppose that it is known, at least approximately, how outflow varies with amount for the particular tub of interest. Suppose also that we know the inflow rate and the capacity of the tub. It would be desirable to use this information to further constrain the simulation. Such an extension to QSIM has been defined and is called Q2 [Kuipers and Berleant, 1988]. With Q2, numerical ranges are associated with each landmark and numerical envelopes are associated with each monotonic function constraint. A range takes the form $x = [low, high]$ and states the relation $lo \leq x \leq high$. The envelope functions are assumed to bracket the true function being modeled. A range propagation algorithm is used in parallel with the normal QSIM algorithm to further reduce the possible number of behaviors based on the quantitative values. The QDE is augmented with the following information :

- A set of envelope functions for the monotonic function constraints. These functions define an upper bound and a lower bound for the function as well as the inverses of these functions[8].

- A set of initial ranges for landmarks. Typically, these ranges will be defined for the landmarks of all independent variables as well as for the initial qmag landmarks of the state variables.

Note that it is not necessary to specify this information for every function and landmark. Q2 will use as much or as little information as is available.

Figure 2.21 augments the bathtub model with quantitative bounds on the $M^+(amount, outflow)$ constraint and with a numeric range for the constant $IF*$. The resulting behavior set is shown in Figure 2.22. Note that the additional information has led to the elimination of two of the behaviors which are not possible under these conditions. Figure 2.23 shows the behaviors of the two tank cascade when the same envelopes are used for the $M^+(amountA, outflowA)$ and $M^+(amountB, outflowB)$ constraints. Previously, this model produced twenty behaviors due to the uncertainty in the relative capacities and outflow rates of the two tanks. With the addition of the envelope functions, the number of behaviors is reduced to two because the outflow rates can now be compared. In addition, the predicted behaviors are more precise as they are now bounded by numerical values.

[8]The inverses are needed when propagating range values of the function back to the domain values.

```
(define-QDE Bathtub-w-ranges
  (text "Simple bathtub model with interval range information")
  (independent inflow)
  (quantity-spaces
    (amount    (0 FULL inf))
    (outflow   (0 inf))
    (inflow    (0 if* inf))
    (netflow   (minf 0 inf)))
  (constraints
    ((M+ amount outflow)    (0 0) (inf inf))
    ((ADD netflow outflow inflow))
    ((D/DT amount netflow)))
  (transitions
    ((amount (FULL inc))))
  (other
    (m-envelopes
      ((M+ amount outflow)
          (upper-envelope (lambda (x) x))
          (upper-inverse  (lambda (y) y))
          (lower-envelope (lambda (x) (if (< x 20) (/ x 2)
                                          (+ (/ x 5) 6))))
          (lower-inverse  (lambda (y) (if (< y 10) (* y 2)
                                          (* 5 (- y 6)))))))
    (initial-ranges ((inflow if*)  (5 10))
                    ((amount full) (50 100)))))
```

Figure 2.21: Qualitative model of the bathtub.

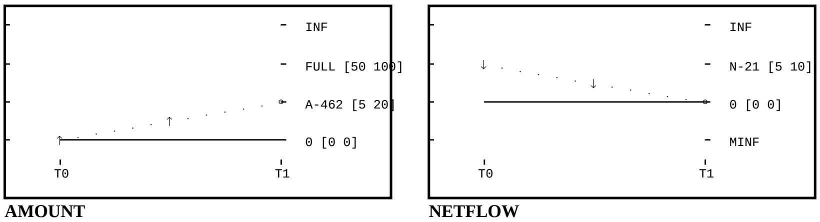
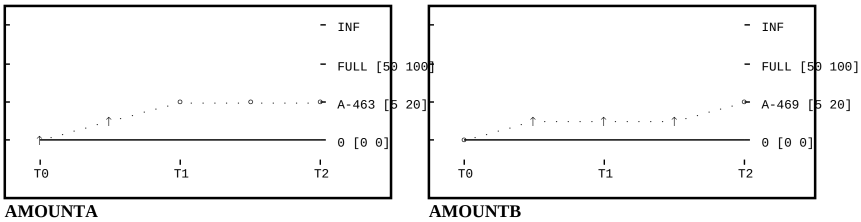
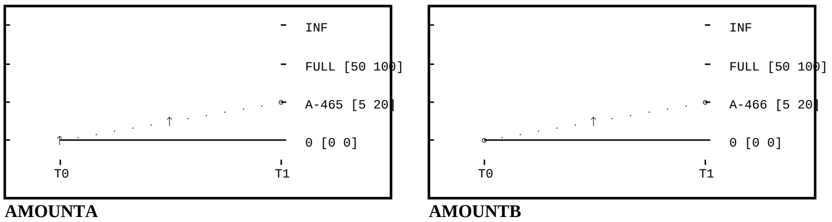


Figure 2.22: The single behavior of the bathtub using Q2.



(a) Tank A reaches equilibrium before B



(b) Both tanks reach equilibrium simultaneously

Figure 2.23: The two behaviors of the cascaded tanks using Q2.

2.5.1 Q2 Models of the Vacuum System

Given the qualitative models for the vacuum system, it is relatively easy to add the quantitative information needed for Q2. There are four monotonic function constraints in the basic model (see Figure 2.13). They are

$$\begin{array}{l} M^+(p, speed) \\ M^+(p, f) \\ M^+(B, desorbrt) \\ M^-(\theta, \alpha) \end{array}$$

Given that the maximum speed of the pump is 250 liters/minute, an envelope for the $M^+(pressure, speed)$ relation can be drawn as shown in Figure 2.24. The actual speed curve will fall between these two envelopes. The constraint $M^+(p, f)$ is precisely defined by Equation 2.2 and therefore one function serves as both envelopes. A similar case holds for $M^+(B, desorbrate)$ (it is equal to B/t_s) and so both of its envelopes are the same.

The envelopes for $M^-(\theta, \alpha)$ are constructed to fall above and below

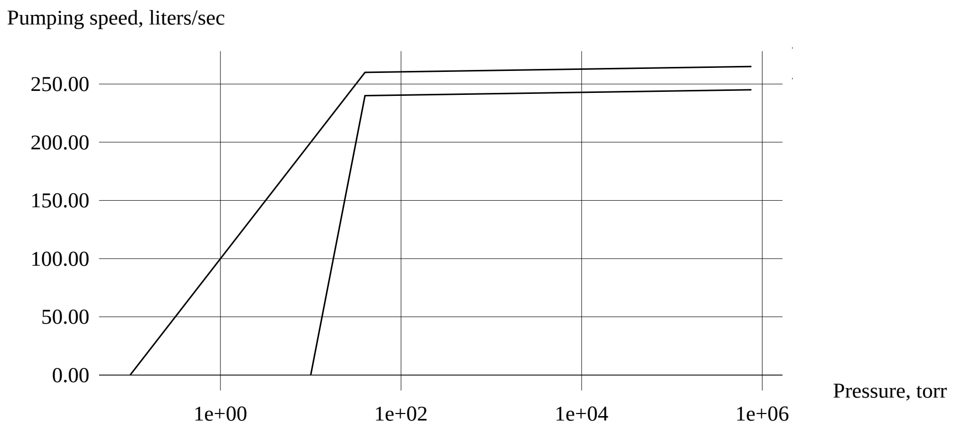


Figure 2.24: Pump speed envelope.

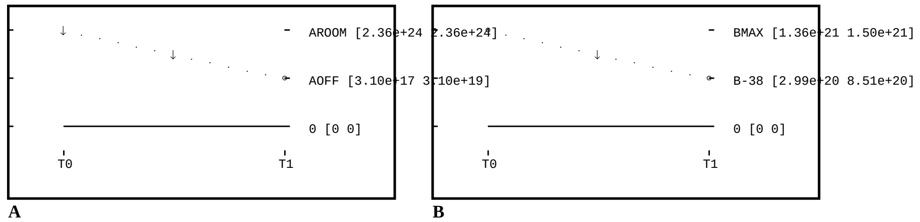
the numerical model's equation for α. The equations for these envelopes are

$$\begin{aligned} \alpha_{min} &= \frac{\exp(\ln(c_a)\theta) + c_a}{2} \\ \alpha_{max} &= 2\exp(\ln(c_a)\theta) - c_a \end{aligned}$$

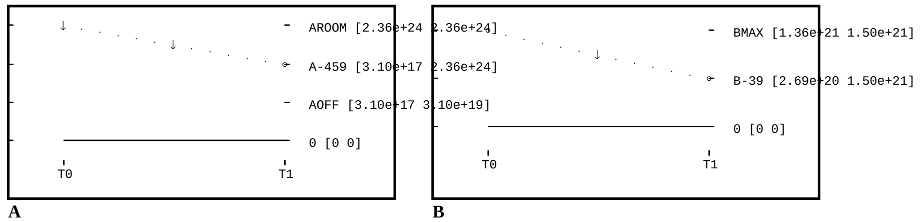
These equations are designed to be equal when $\theta = 1$, which is the initial condition of the system. At all other times $\alpha_{min} < \alpha_{max}$.

The initial value for the number of molecules in "A" is assumed to be known precisely and the initial number of molecules adsorbed in the walls is a range that depends on the chamber wall surface area. Figure 2.25a shows the result of simulating the basic model using Q2. Q2 adds numerical information that can be compared against sensor data from the real process.

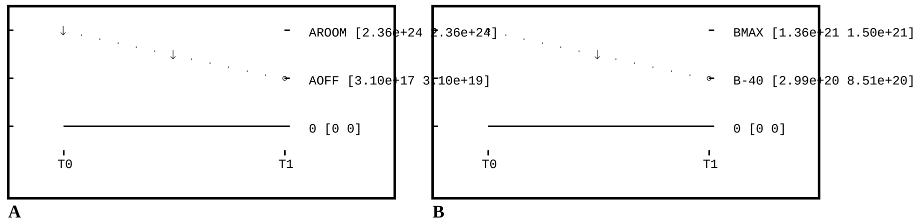
In the real leak model, a leak conductance range of [0.001, 0.01] is assumed and the corresponding envelopes for $M^+(proom - p, dleak)$ are defined as for $leakrate$ (see page 27), but with the upper and lower conductance ranges substituted for C_{leak}. Similarly, a leak conductance range of [10^{-7}, 10^{-5}] is assumed for the virtual leak model and the corresponding envelopes



(a) Basic model



(b) Real leak



(c) Virtual leak

Figure 2.25: Q2 behaviors of the real leak and virtual leak models.

are defined using the same *leakrate* equation. The plots of A and B for these two models are shown in Figures 2.25b and c. Table 2.1 compares the Q2 ranges for each of the landmarks in the three models. There are two things to note about these ranges. First, the ranges for the working and virtual leak systems are identical.

As previously discussed, this is not surprising since the effects of a virtual leak at these pressures is inconsequential. Second, and more disturbing, is that the ranges for the real leak, while different from the working system, do not really provide much help in fault recognition because they

Landmark	Basic model	Real leak	Virtual leak
AROOM	[2.36e24, 2.36e24]	[2.36e24, 2.36e24]	[2.36e24, 2.36e24]
AOFF	[3.10e17, 3.10e19]	[3.10e17, 3.10e19]	[3.10e17, 3.10e19]
A(T1)	[3.10e17, 3.10e19]	[3.10e17, 2.36e24]	[3.10e17, 3.10e19]
BMAX	[1.36e21, 1.50e21]	[1.36e21, 1.50e21]	[1.36e21, 1.50e21]
B(T1)	[2.99e20, 8.51e20]	[2.69e20, 1.50e21]	[2.99e20, 8.51e20]

Table 2.1: Q2 ranges for landmarks of the three vacuum models.

are so coarse[9]. It is only the qualitative value at A(T1) which differentiates the models, and even this value cannot differentiate the basic model from the virtual leak.

The coarseness of the interval predictions is due to the weakness of the interval propagation method used by the Q2 quantitative range reasoner. Although it does provide bounding values for system variables at distinguished time points, it cannot predict how the system will behave *between* these time points. In the next chapter, we investigate a method that can make stronger predictions about variable trajectories over time intervals.

[9]In fact, the range for A(T1) for the working system is a proper subset of the range for the real leak.

Chapter 3

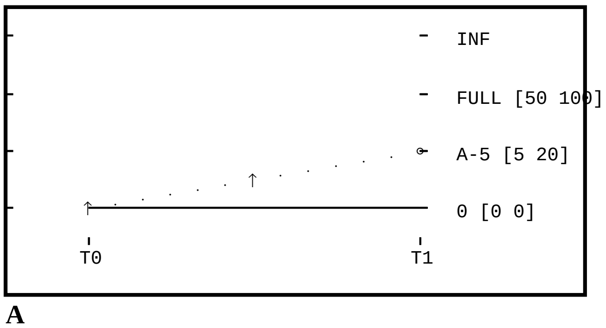
Improving the Predictions of the Qualitative Models

3.1 Introduction

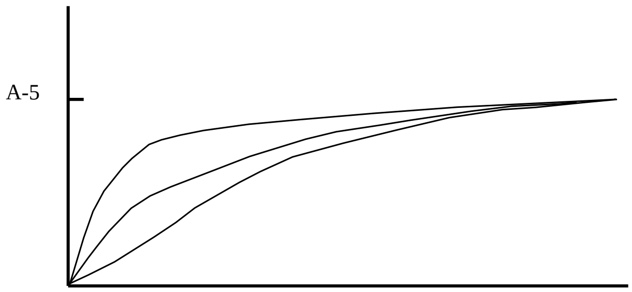
The QSIM system can simulate a wide variety of dynamic systems and make qualitative predictions about them. Combined with the Q2 quantitative range reasoner, the system can make limited numerical predictions from the models. Unfortunately, these predictions are fairly weak over time intervals. This weakness is especially disadvantageous for process monitoring tasks where early detection of deviations is vital. This chapter explores the nature of this weakness and suggests an improvement that greatly increases the accuracy of the simulation over time intervals.

3.2 The Dynamic Envelope Method

The dynamic envelope method attempts to eliminate the weaknesses of range interval propagation. As an example, consider the plot of the bathtub using Q2 repeated here as Figure 3.1a. Over the time interval $[t_0, t_1]$, A falls in the range [0, 20] and is continually increasing. Figure 3.1b shows some possible trajectories that fall in [0, 20] and are continually increasing. Even though Q2 has an exact numerical definition of the envelopes, it is unable to use them more effectively because of limitations in range propagation across derivative constraints. The propagation method uses the mean value theorem to constrain the ranges of a variable at two adjacent time points. For the



(a) Q2 behavior of the bathtub.



(b) Possible curves corresponding to the behavior.

Figure 3.1: The single behavior of the bathtub using Q2.

variable x with derivative y, this amounts to solving the equation

$$x(t_{n+1}) = x(t_n) + (t_{n+1} - t_n)y(t^*)$$

where $y(t^*)$ is in the interval $[y(t_n), y(t_{n+1})]$. Since this interval can be formed from the values of y at the two surrounding timepoints, Q2 does not need to reason about the behavior of the system during $[t_n, t_{n+1}]$ and so its prediction over the interval is simply that x falls somewhere between the minimum and maximum range values of $x(t_n)$ and $x(t_{n+1})$. This forms a box in the x-t plane. The *dynamic envelope method* attempts to shrink this box by replacing the use of the mean value theorem with explicit integration by using the equation

$$x(t_{n+1}) = x(t_n) + \int_{t_n}^{t_{n+1}} y dt.$$

Specifically, bounding ordinary differential equation (ODE) systems derived from the QDE are numerically evaluated using the Q2 envelopes and the initial ranges in parallel with the normal Q2 algorithm. This approach captures the benefits of both qualitative and quantitative reasoning as all possible behaviors of the system are simulated [Kuipers, 1986], and tighter numerical bounds are deduced yielding more accurate predictions of each behavior.

3.3 Details of the Method

The goal of dynamic envelope simulation is to determine the upper and lower bounds of every variable of interest in the QDE at every point in time. This is done by forming a system of ODEs from the QDE and simulating them using standard numerical techniques. To do this, the set of model variables is divided into three groups; state variables, constant (or independent) parameters, and "other" intermediate variables (which can be determined from the state and constant variables). The information required to simulate the envelopes is

- The set of state variables.
- A set of equations for the derivatives of the state variables, and an equation for each "other" variable.
- The envelopes for the monotonic functions.
- The initial values for all state variables and constants.

Note that a Q2-augmented QDE contains all of this information.

Since a QDE, even a Q2-augmented one, represents a set of ODEs, some method is needed to construct a tractable set of equations that bound the set of possible ODEs. The method described here derives extremal systems from the Q2-augmented QDE.

To illustrate the method on a simple case, consider the single-variable system

$$A' = f(A) \tag{3.1}$$

where f is an unknown monotonically increasing function. Let $\underline{f}$ and $\overline{f}$ be known, computable functions such that $\underline{f}(x) < f(x) < \overline{f}(x)$ for all x (i.e., envelopes for f). Let $A(t)$ be any particular solution to Equation 3.1. Define a "minimum" equation

$$\underline{A}' = \underline{f}(\underline{A}) \tag{3.2}$$

and a "maximum" equation

$$\overline{A}' = \overline{f}(\overline{A}) \tag{3.3}$$

and let their solutions be $\underline{A}(t)$ and $\overline{A}(t)$ respectively with

$$\underline{A}(0) \leq A(0) \leq \overline{A}(0) \tag{3.4}$$

We can show that for all t, $\underline{A}(t) < A(t) < \overline{A}(t)$ as follows. Consider the relation between $\underline{A}$ and A. At any time t we have $\underline{A} < A$ (in which case $\underline{A}$ is clearly a lower bound at t), $\underline{A} > A$, or $\underline{A} = A$. If $\underline{A} > A$ then by (3.4) there must be some time $t_1 < t$ where $\underline{A} = A$. We therefore consider only the case where $\underline{A}(t_1) = A(t_1)$ and we seek to show that $\underline{A}$ does not cross over A at t_1. Since $\underline{f}(A(t_1)) < f(A(t_1))$, at $t_1 + \varepsilon$ we must have that $\underline{A} < A$ for an arbitrarily small ε and thus $\underline{A}$ does not cross A at t_1. A similar argument shows that $\overline{A}(t_1 + \varepsilon) > A(t_1 + \varepsilon)$ and so $\overline{A}$ does not cross A at t_1.

The effect of this method can be seen on the bathtub model of Section 2.3. This model is of the same form as Equation 3.1 where the monotonic function is $c - f(x)$.[1] Figure 3.2 plots the envelopes defined by $\overline{A}$ and $\underline{A}$ superimposed with the Q2 range predictions for the bathtub. Note that the bounds on the system have been tightened, especially for values near $t = 0$ and as $t \to \infty$.

For systems of coupled equations, the above method can be expanded by noting that to guarantee that the state variable $\underline{A}$ in Equation 3.1 was a lower bound of A we must find an expression for the derivative of $\underline{A}$ that is less than $f(A)$ whenever $\underline{A} = A$. For systems of ODEs, this may mean that to minimize some variable A we may need to maximize some other variable B.[2] These considerations are now formalized.

[1] Note that in this example, the bounding ODEs (Equations 3.2 and 3.3) are in fact, instances of the QDE (Equation 3.1). In general, however, this is not the case.

[2] For instance, if $A' = A - B$ then $\underline{A}' = \underline{A} - \overline{B}$.

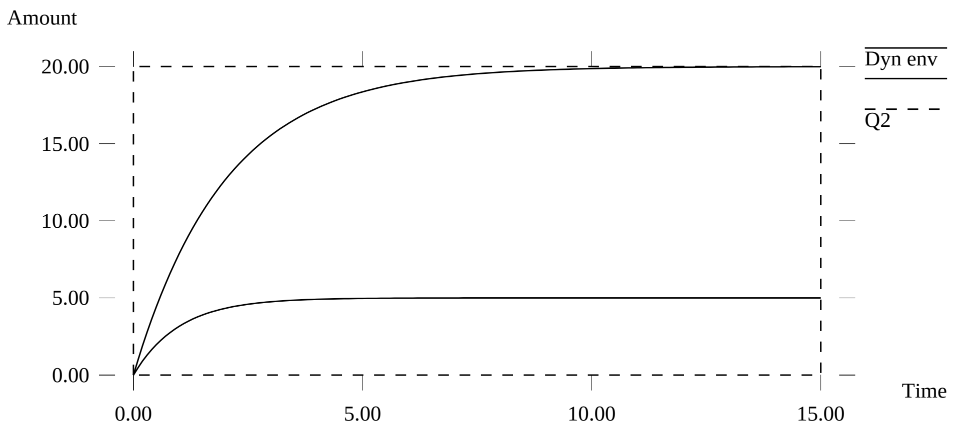


Figure 3.2: Dynamic envelopes for the bathtub with Q2 range prediction superimposed.

y	y $\mathbf{R}$ x	y $\mathbf{R}_=$ x	y $\neg\mathbf{R}$ x	y $\neg\mathbf{R}_=$ x
$\underline{x}$	$<$	$\leq$	$\geq$	$>$
$\overline{x}$	$>$	$\geq$	$\leq$	$<$

Table 3.1: The definition of the $\mathbf{R}$ operator.

3.3.1 Bounded State Variables

Definition 1 *Let x be a state variable of some ODE system. Then $\underline{x}$ and $\overline{x}$ represent the lower and upper bounds on x and are called the* bounded state variables *of x.*

Bounded Variable Relations In what follows, y refers to either the upper or lower bound of x. The function $bound(y)$ equals L if $y \equiv \underline{x}$ and U if $y \equiv \overline{x}$. The relations $\mathbf{R}$, $\mathbf{R}_=$, $\neg\mathbf{R}$, and $\neg\mathbf{R}_=$ represent inequalities between y and x as defined in Table 3.1. Note that y $\mathbf{R}$ x implies $-y$ $\neg\mathbf{R}_=$ $-x$. The $\mathbf{R}$ relation also applies to derivative functions such that $\mathbf{R} \equiv <$ in g_i $\mathbf{R}$ f_i if g_i is the derivative function for y_i and $bound(y_i) = L$. Normally, these relations are determined by the bound of y. However, under some conditions, it is necessary to compare expressions in x_j and y_j using the comparison relation determined by x_i and y_i. In this case, the notation $\mathbf{R_{y_i}}$ is used. For example, in the expression $g(y_j)\ \mathbf{R_{y_i}}\ f(x_j)$, $\mathbf{R_{y_i}} \equiv <$ if $y_i \equiv \underline{x}_i$. The $\mathbf{R}$ relation also applies to a vector in which case it stands for a vector of relations such that for each element $y_i \in \mathbf{y}$, $y_i\ \mathbf{R}\ x_i$. The following lemma relates to bound notation.

Lemma 1 *If $bound(y_j) \neq bound(y_i)$ then $(g_j\ \mathbf{R}\ f_j) \equiv (g_j\ \neg\mathbf{R_{=,y_i}}\ f_j)$.*

Proof We prove the lemma for the case $bound(y_j) = L$. A similar proof holds for $bound(y_j) = U$. Assume that $bound(y_i) = U$. This means that $(g_j\ \mathbf{R_{y_j}}\ f_j) \equiv (g_j < f_j)$ and $\mathbf{R_{y_i}}$ is $>$. From Table 3.1, if $\mathbf{R_{y_i}} \equiv >$ then $\neg\mathbf{R_{=,y_i}} \equiv <$ and so $(g_j < f_j) \equiv (g_j\ \neg\mathbf{R_{=,y_i}}\ f_j)$.

□

3.3.2 Extremal Equations and Systems

Definition 2 *Let $A : \mathbf{x}' = \mathbf{f}(\mathbf{x})$ be an ODE system with state vector $\mathbf{x}$. For each $x_i \in \mathbf{x}$, let $x_i' = f_i(\mathbf{x_i})$ be the equation for the derivative of x_i where $\mathbf{x_i} \subseteq \mathbf{x}$ is the set of state variables that f_i directly depends upon. Then $y_i' = g_i(\mathbf{y_i})$ is an* extremal equation *for x_i where*

1. *For all $x_h \in \mathbf{x_i}$, y_h is a bounded state variable of x_h such that*

$$\frac{\partial g_i}{\partial y_h} \begin{cases} \geq 0 & \text{if } bound(y_i) = bound(y_h) \\ \leq 0 & \text{if } bound(y_i) \neq bound(y_h) \end{cases}$$

2. *If*

   - $\mathbf{y_i}\ \mathbf{R_=}\ \mathbf{x_i}$, *and*

- *$y_i = x_i$, and*
- *$\exists_{x_k \in \mathbf{x_i}}\ y_k \neq x_k$*

then g_i $\mathbf{R}$ f_i,

3. *Either $\forall_{\mathbf{x}}\ g_i(\mathbf{x})$ $\mathbf{R}$ $f_i(\mathbf{x})$ or $\forall_{\mathbf{x}}\ g_i(\mathbf{x}) = f_i(\mathbf{x})$.*

We call g_i an *extremal expression* for f_i with respect to x_i.

Definition 2 states that g_i will be an extremal expression for f_i if 1) each bounded state variable monotonically influences g_i, 2) when $y_i = x_i$ and $\mathbf{y}$ is on the "correct side" of $\mathbf{x}$, g_i will also be on the correct side of f_i, and 3) g_i is on the correct side of f_i at every point in the state space of the system. Note that if $g_i = f_i$ at some point in the state space then $g_i \equiv f_i$ by Part 3 of the definition.

Definition 3 *Let $A : \mathbf{x}' = \mathbf{f}(\mathbf{x})$ be an ODE system with state vector $\mathbf{x}$. Then $\alpha : \mathbf{y}' = \mathbf{g}(\mathbf{y})$ is an* extremal system *for A if, for all i, $y_i' = g_i(\mathbf{y_i})$ is an extremal equation for x_i and in all equations y_i represents the same bound on x_i.*

An extremal system α is composed of a set of maximum and minimum equations such that each state variable of A is represented by either an upper or a lower bounded state variable, but not both.

With these definitions, the following general result can be proven :

Theorem 1 *Let $A : \mathbf{x}' = \mathbf{f}(\mathbf{x})$ be an ODE system. Let $\alpha : \mathbf{y}' = \mathbf{g}(\mathbf{y})$ be an extremal system for A. Assume that for all i, y_i $\mathbf{R}_=$ x_i at $t = 0$. Then for all t, $y_i(t)$ $\mathbf{R}_=$ $x_i(t)$.*

In order to prove this theorem, we first show that y_i cannot cross x_i given that y_i is on the correct side of x_i at some time t.

Lemma 2 *Assume that systems A and α are defined as in Theorem 1. For all t, if $\forall_j$, y_j $\mathbf{R}_=$ x_j at time t then y_i does not cross x_i at t.*

Proof outline We are attempting to prove that y_i remains on its correct side of x_i shortly after time t if it is on the correct side of x_i at time t. This is clearly true if $y_i(t)$ $\mathbf{R}$ $x_i(t)$. If $y_i(t) = x_i(t)$ then we wish to show that $y_i(t + \varepsilon)$ $\mathbf{R}$ $x_i(t+\varepsilon)$. We therefore seek to identify the discrepancies between the two ODE systems and to determine how they affect x_i and y_i. Discrepancies may occur because of differences in the state variables or the derivative functions of the two systems. We can determine the effect of the discrepancies by forming Taylor expansions about t for x_i and y_i and comparing successive terms. We can view each ODE system as a digraph whose nodes are the derivative functions of the system and whose edges point from state variables to their derivative functions (see Figure 3.3). If we choose node x_i' (or y_i') as the root of the graph, we can describe the effect of a discrepancy between A and α as a difference in the value of one of the time derivatives of x_i and y_i. It will turn out that for a discrepancy N steps away from x_i' (or y_i') in the graph, $y_i^{(m)} = x_i^{(m)}$ for $1 \leq m \leq N$ and $y_i^{(N)}$ $\mathbf{R}$ $x_i^{(N)}$ for some N and so, by virtue of their Taylor expansions, $y_i(t + \varepsilon)$ $\mathbf{R}$ $x_i(t + \varepsilon)$ thus proving the lemma.

Notation The definitions and lemmas in this section use the notation described in Table 3.2. The notation works equally well for system α by replacing instances of x and f with y and g and so all lemmas proved for A also hold for α.

x_i	The state variable of interest in Lemma 2. We assume that $y_i = x_i$ at t.
x_i'	The first time derivative of x_i. Note that $x_i' \equiv f_i$ and $y_i' \equiv g_i$.
$\mathbf{x_i}$	The set of state variables that directly affect x_i' (i.e., $x_i' = f_i(\mathbf{x_i})$).
$x_i^{(m)}$	The mth time derivative of x_i.
N	The graph distance from x_i' to the closest discrepancy between A and α (see Section 3.3.3). Note that $N \leq n$ where n is the number of equations in the system.

Table 3.2: Notation for the lemmas and definitions in this section.

The next section proves some useful properties of ODE systems that obey the definition of extremal systems. In particular, the notion of a distance of a state variable or derivative function from x_i' is defined and the following properties are proved

- Variables or derivative functions further away than N don't affect the Nth time derivative of x_i or y_i.
- The relative values of $bound(y_i)$ and $bound(y_j)$ affect the sign of the Nth time derivative of y_i.

Given these properties of systems and distances, the next section proves the following properties :

- If the closest discrepancy between x_i' and y_i' is at N then the first N time derivatives of x_i and y_i are identical.
- If the closest discrepancy between x_i' and y_i' is at N then $y_i^{(N+1)}\ \mathbf{R}\ x_i^{(N+1)}$.

$$
\begin{aligned}
X_1' &= f_1(X_2, X_3)\\
X_2' &= f_2(X_2, X_4)\\
X_3' &= f_3(X_4)\\
X_4' &= f_4(X_1, X_3)
\end{aligned}
$$

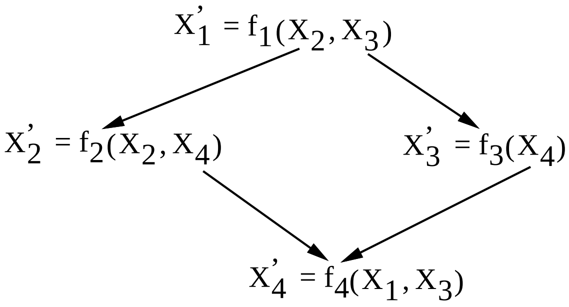


Figure 3.3: An ODE system and its digraph rooted at x_1.

3.3.3 Distances in ODE Systems

We can view the ODE system as a digraph whose nodes are the derivative functions and whose links point from state variables to their derivative functions (see Figure 3.3). We assume that the ODE system is not decomposable into separate systems, so for any x_i', there is a path to every state variable in the system. We can represent the "distance" from x_i' to any state variable or derivative function as the minimum graph distance between them.

Definition 4 *The* distance *from the derivative equation for x_i' to a state variable x_j is*

$$
d(x_i', x_j) = \begin{cases} 0 & \text{if } x_j \in \mathbf{x_i} \\ n & \text{if } min(d(x_k', x_j)) = n - 1 \text{ and } x_j \in \mathbf{x_i} \end{cases}
$$

The distance from the derivative equation for x_i' to the derivative function for x_j' is

$$d(x_i', x_j') = \begin{cases} 0 & \textit{if } j = i \\ 1 + d(x_i', x_j) & \textit{otherwise} \end{cases}$$

Note that (except for x_i itself) a state variable is always one step closer to x_i' than its derivative function. Note also that for any derivative function f_j, $d(f_j, x_k) = 0$ if and only if $x_k \in \mathbf{x_j}$. This observation leads to the following lemma.

Lemma 3 *For all j, if $d(f_j, x_k) \neq 0$ then $\frac{\partial f_j}{\partial x_k} = 0$.*

We now show that state variables further away than N will have no effect on the Nth time derivative of x_i (Lemma 4) and that the bound of x_k will affect the sign of the $\frac{\partial x_i^{(N)}}{\partial x_k}$ (Lemma 5).

Lemma 4 *For any $N \geq 1$ and any j, if $d(x_i', x_j) \geq N$ then $\frac{\partial x_i^{(N)}}{\partial x_j} = 0$.*

Proof This result is proved by induction. If $N = 1$ then $d(x_i', x_j) \geq 1$ implies $\frac{\partial f_i}{\partial x_j} \equiv \frac{\partial x_i^{(1)}}{\partial x_j} = 0$ by Lemma 3.

Assume that the lemma is true for N. We seek to prove that if $d(x_i', x_j) \geq N + 1$ then $\frac{\partial x_i^{(N+1)}}{\partial x_j} = 0$. Expanding this partial derivative gives

$$\frac{\partial x_i^{(N+1)}}{\partial x_j} = \sum_{k=1}^{n} \left(\frac{\partial x_i^{(N)}}{\partial x_j \partial x_k} f_k + \frac{\partial x_i^{(N)}}{\partial x_k} \frac{\partial f_k}{\partial x_j} \right)$$

Since $d(x_i', x_j) \geq N + 1$ we know that $d(x_i', x_j) \geq N$ and so $\frac{\partial x_i^{(N)}}{\partial x_j} = 0$ by our assumption that the lemma is true for N. This implies $\frac{\partial x_i^{(N)}}{\partial x_j \partial x_k} = 0$ and so the first term of the expression in parentheses is zero. To determine the value of the second term of the expression in parentheses, we consider two cases

- $d(x_i', x_k) \geq N$

  By the assumption that the lemma is true for N, $\frac{\partial x_i^{(N)}}{\partial x_k} = 0$, so this term must be zero.

- $d(x_i', x_k) < N$

  In this case, $d(x_i', f_k) < N+1$ by Definition 4. However, since $d(x_i', x_j) \geq N+1$ we have $d(f_k, x_j) > 0$ and so by Lemma 3, $\frac{\partial f_k}{\partial x_j} = 0$.

Therefore, $\frac{\partial x_i^{(N+1)}}{\partial x_j} = 0$.

□

Lemma 5 *Let $d(x_i', x_j) \geq N$. If bound(x_i) = bound(x_j) then $\frac{\partial x_i^{(N)}}{\partial x_j} \geq 0$. If bound$(x_i) \neq$ bound(x_j) then $\frac{\partial x_i^{(N)}}{\partial x_j} \leq 0$.*

Proof This result is proved by induction. If $N = 1$ then the result follows directly from Definition 2, Part 1.

Now assume that the lemma is true for N. We seek to prove that if $bound(x_i) = bound(x_j)$ then $\frac{\partial x_i^{(N+1)}}{\partial x_j} \geq 0$ and if $bound(x_i) \neq bound(x_j)$ then $\frac{\partial x_i^{(N+1)}}{\partial x_j} \leq 0$, given that $d(x_i', x_j) \geq N + 1$. Expanding $\frac{\partial x_i^{(N+1)}}{\partial x_j}$ gives

$$\frac{\partial x_i^{(N+1)}}{\partial x_j} = \sum_{k=1}^{n} \left(\frac{\partial x_i^{(N)}}{\partial x_j \partial x_k} f_k + \frac{\partial x_i^{(N)}}{\partial x_k} \frac{\partial f_k}{\partial x_j}\right)$$

By Lemma 4 $\frac{\partial x_i^{(N)}}{\partial x_j \partial x_k} = 0$ and so the first term in parentheses is zero.

Assume that $bound(x_i) = bound(x_j)$. Now, for each x_k such that $bound(x_k) = bound(x_j)$, we have $\frac{\partial f_k}{\partial x_j} \geq 0$ by Definition 2, Part 1 and $\frac{\partial x_i^{(N)}}{\partial x_k} \geq 0$ by the assumption that the lemma is true for N. Hence, $\frac{\partial x_i^{(N)}}{\partial x_k} \frac{\partial f_k}{\partial x_j} \geq 0$. For each x_k such that $bound(x_k) \neq bound(x_j)$ then $\frac{\partial g_k}{\partial x_j} \leq 0$ and $\frac{\partial x_i^{(N)}}{\partial x_k} \leq 0$ and so once again $\frac{\partial x_i^{(N)}}{\partial x_k} \frac{\partial f_k}{\partial x_j} \geq 0$. Therefore, all terms of the sum are greater than or equal to zero and so $\frac{\partial x_i^{(N+1)}}{\partial x_j} \geq 0$. A similar result holds if $bound(x_i) \neq bound(x_j)$.

$\square$

3.3.4 Discrepancies between A and α

A discrepancy can occur at any node in the graph as either a difference in state variable values between A and α or a difference in derivative function values. By Definition 2, Part 3 we know that if two derivative values are equal then the derivative functions are identical. Therefore, for a discrepancy at N, $f_k \equiv g_k$ if f_k is less than N away from x'_i.

Lemma 6 *Let the closest discrepancy to x'_i be at N. For all $k \neq i$, if $d(x'_i, x_k) < N$ then $y_k = x_k$. For all k, if $d(x'_i, x'_k) < N$ then $g_k \equiv f_k$.*

We now prove that a discrepancy N units from x'_i will first affect the $N+1$st time derivative of x_i. First, we show that time derivatives less than N will be equivalent in A and α (Lemma 7). Then we show that the $N+1$st derivative will cause y_i to turn towards the correct side of x_i (Lemma 8).

Lemma 7 *If the closest discrepancy between A and α is N units away from x'_i and y'_i then*
$\forall_{1 \leq m \leq N, \mathbf{y}}\ y_i^{(m)} \equiv x_i^{(m)}$.

Proof We prove the result by induction on N. If $N = 1$ then $d(x'_i, x'_i) < 1$ which implies $y_i^{(1)} \equiv x_i^{(1)}$ by Lemma 6.

Now assume that the lemma is true for N and prove it for $N+1$. Assume that the closest discrepancy is at $N+1$. From our assumption, we know that $y_i^{(m)} \equiv x_i^{(m)}$ for $1 \leq m \leq N$. All that remains is to show that $y_i^{(N+1)} \equiv x_i^{(N+1)}$. Expanding these terms gives

$$
\begin{aligned}
x_i^{(N+1)} &= \sum_{j=1}^{n} \frac{\partial x_i^{(N)}}{\partial x_j} f_j \\
y_i^{(N+1)} &= \sum_{j=1}^{n} \frac{\partial y_i^{(N)}}{\partial y_j} g_j
\end{aligned}
$$

By assumption $y_i^{(N)} \equiv x_i^{(N)}$ and so $\frac{\partial y_i^{(N)}}{\partial y_j} \equiv \frac{\partial x_i^{(N)}}{\partial x_j}$ for all j. Since the closest discrepancy is at $N+1$, by Lemma 6 we have $g_j \equiv f_j$ for all x_j such that $d(x_i', x_j) < N+1$. By Lemma 4 we have that $\frac{\partial x_i^{(N)}}{\partial x_j} = 0$ for $d(x_i', x_j) \geq N$. Therefore, if $\frac{\partial x_i^{(N)}}{\partial x_j} \neq 0$ we have that $f_j \equiv g_j$ and so $y_i^{(N+1)} \equiv x_i^{(N+1)}$.

$\square$

Lemma 8 *Under the assumptions of Lemma 2, if the closest discrepancy to x_i' is at N then $y_i^{(N+1)}$ $\mathbf{R}$ $x_i^{(N+1)}$.*

Proof Expanding both terms gives

$$
\begin{aligned}
x_i^{(N+1)} &= \sum_{j=1}^{n} \frac{\partial x_i^{(N)}}{\partial x_j} f_j && (3.5) \\
y_i^{(N+1)} &= \sum_{j=1}^{n} \frac{\partial y_i^{(N)}}{\partial y_j} g_j && (3.6)
\end{aligned}
$$

From Lemma 7 we have $\forall_j \ \frac{\partial y_i^{(N)}}{\partial y_j} \equiv \frac{\partial x_i^{(N)}}{\partial x_j}$. For each x_j, there are two cases to consider :

- $d(x_i', x_j) \geq N$

  By Lemma 4 we have $\frac{\partial x_i^{(N)}}{\partial x_j} = 0$ so the jth term of the sum vanishes.

- $d(x_i', x_j) < N$

  By Definition 4 we have that $d(x_i', f_j) < N+1$. If $d(x_i', f_j) < N$ then by Lemma 6 we have $g_j \equiv f_j$ and so the jth terms of the sums are equivalent. If $d(x_i', f_j) = N$ then either $g_j \equiv f_j$ (in which the terms are equal) or $g_j \neq f_j$. In this case, by Definition 2, Part 2 (if $\mathbf{y} \neq \mathbf{x}$) or Part 3 (if $\mathbf{y} = \mathbf{x}$) we have $g_j \ \mathbf{R} \ f_j$ and hence $\frac{\partial y_i^{(N)}}{\partial y_j} g_j \neq \frac{\partial x_i^{(N)}}{\partial x_j} f_j$.

The relation between these unequal terms is dependent on the signs of the partial derivatives. There are two cases to consider

- $bound(y_i) = bound(y_j)$
  By Lemma 5 we have that $\frac{\partial y_i^{(N)}}{\partial y_j} > 0$ and so $y_i^{(N+1)} \ \mathbf{R} \ x_i^{(N+1)}$.
- $bound(y_i) \neq bound(y_j)$
  By Lemma 1 we have $g_j \ \neg\mathbf{R}_{=,\mathbf{y_i}} \ f_j$ and by Lemma 5 we have that $\frac{\partial y_i^{(N)}}{\partial y_k} < 0$. Since $y \ \neg\mathbf{R}_{=} \ x$ implies $-y \ \mathbf{R} \ -x$ we have $\frac{\partial y_i^{(N)}}{\partial y_j} g_j \ \mathbf{R}_{\mathbf{y_i}} \ \frac{\partial x_i^{(N)}}{\partial x_j} f_j$ and so $y_i^{(N+1)} \ \mathbf{R}_{=} \ x_i^{(N+1)}$.

□

We are now in a position to prove Lemma 2.

Proof of Lemma 2 At any t we have either $y_i \ \mathbf{R} \ x_i$ or $y_i = x_i$. If $y_i \ \mathbf{R} \ x_i$ then clearly y_i does not cross x_i at t. Assume therefore, that $y_i = x_i$. There are two cases to consider.

- $\mathbf{y} = \mathbf{x}$ and $\mathbf{g} \equiv \mathbf{f}$
  In this case, $\alpha \equiv A$ and so $\mathbf{y} = \mathbf{x}$ for all time greater than t and so y_i does not cross x_i at t.
- $\exists_j \ y_j \neq x_j$ or $\exists_j \ g_j \neq f_j$
  Assume that the closest discrepancy occurs at a distance N from x_i'.

If $N = 0$ then either $\mathbf{y_i} = \mathbf{x_i}$ in which case by Definition 2, Part 3 we have that $y_i' \ \mathbf{R} \ x_i'$, or there exists some $x_j \in \mathbf{x_i}$ such that $y_j \neq x_j$ in which case by Definition 2, Part 2 we have that $y_i' \ \mathbf{R} \ x_i'$.

If $N \geq 1$ then by Lemmas 7 and 8 we have that $y_i^{(m)} = x_i^{(m)}, 1 \leq m \leq N$ and $y_i^{(N+1)} \ \mathbf{R} \ x_i^{(N+1)}$. Forming a Taylor expansion for x_i and y_i gives

$$\begin{aligned} x_i(t+\varepsilon) &= x_i + \varepsilon x_i' + \sum_{j=2}^{N} \varepsilon^j \frac{x_i^{(j)}}{j!} + \varepsilon^{N+1} \frac{x_i^{(N+1)}}{(N+1)!} + \mathcal{O}(\varepsilon^{N+2}) \\ y_i(t+\varepsilon) &= y_i + \varepsilon y_i' + \sum_{j=2}^{N} \varepsilon^j \frac{y_i^{(j)}}{j!} + \varepsilon^{N+1} \frac{y_i^{(N+1)}}{(N+1)!} + \mathcal{O}(\varepsilon^{N+2}) \end{aligned}$$

By assumption $y_i = x_i$. For the case $N = 0$ we have $\varepsilon y_i' \ \mathbf{R} \ \varepsilon x_i'$ and so for arbitrarily small ε $y_i(t+\varepsilon) \ \mathbf{R} \ x_i(t+\varepsilon)$ and the two variables do not cross. For the case $N \geq 1$, we have that all terms of the series but the last are equal and so once again $y_i(t+\varepsilon) \ \mathbf{R} \ x_i(t+\varepsilon)$ and so the two variables do not cross

Putting together these cases we see that y_i does not cross x_i at t.

□

We can now prove Theorem 1.

Proof of Theorem 1 Assume that at time t_1 $\underline{x}_i$ crosses x_i and assume further that this is the first time that any variable of α crosses a variable of A. By Lemma 2 we have that for some $j \neq i$ $y_j \ \neg\mathbf{R}_= \ x_j$ at t_1. But since $y_j \ \mathbf{R}_= \ x_j$ at $t = 0$, we must have that y_j crosses x_j at some time $t_2 < t_1$. This contradicts our assumption that t_1 is the earliest time for a crossing and therefore there can be no such time. Hence $\underline{x}_i$ never crosses x_i and therefore $\underline{x}_i \leq x_i$ for all t.

□

As an example of the method, consider the case of the two tank cascade in section 2.3. The equations for this system are :

$$\begin{aligned} A' &= inflow - outflowA(A) \\ B' &= outflowA(A) - outflowB(B) \end{aligned}$$

Let us assume that $inflow$ is precisely known and that $outflowA$ is bounded by

$$b(A) < outflowA(A) < a(A)$$

and $outflowB$ is bounded by

$$g(A) < outflowB(B) < f(B)$$

The envelope systems for this QDE are :

$$\begin{aligned} \underline{A}' &= inflow - a(\underline{A}) \\ \underline{B}' &= b(\underline{A}) - f(\underline{B}) \end{aligned}$$

and

$$\begin{aligned} \overline{A}' &= inflow - b(\overline{A}) \\ \overline{B}' &= a(\overline{A}) - g(\overline{B}) \end{aligned}$$

Note that these systems do not represent physically meaningful models of the cascade as the outflow of A does not equal the inflow of B. They do, however, bound the solutions of the QDE. Figure 3.4 shows the plots for these two systems together with the outline of the range predicted by Q2. Note that the envelope for B has an "optimistic" upper bound with respect to the

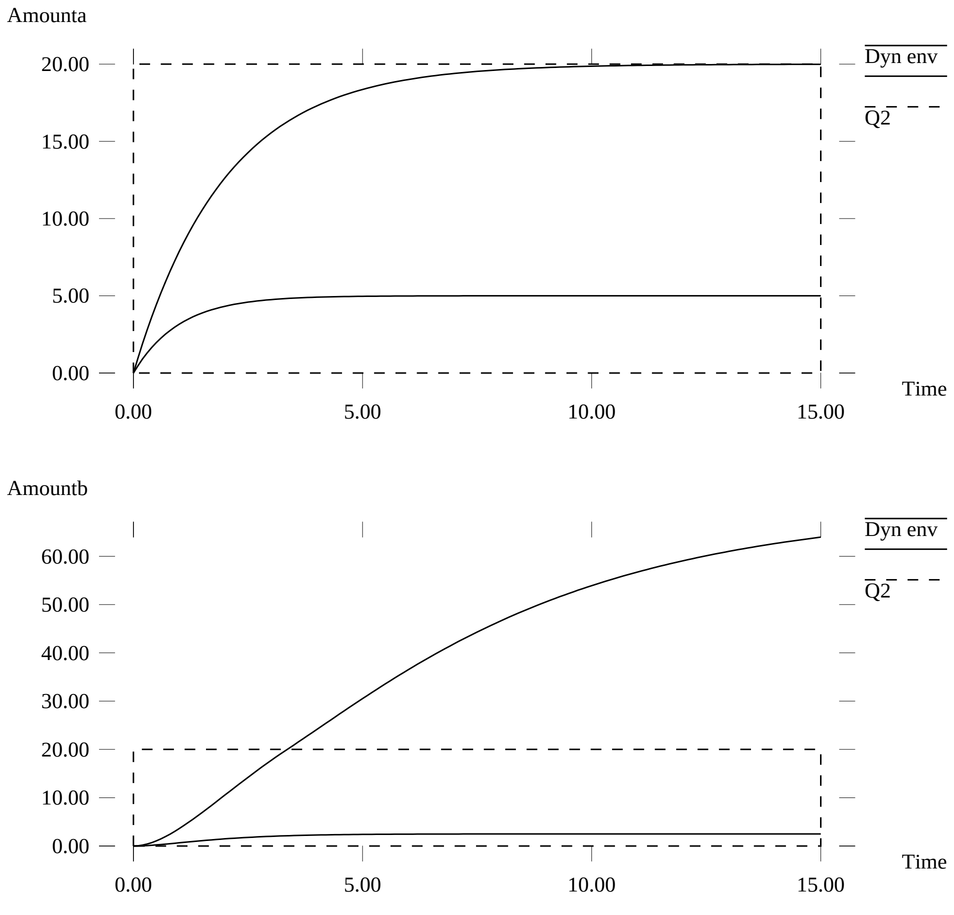


Figure 3.4: Dynamic envelopes for the two tank cascade with Q2 range prediction superimposed.

Q2 prediction after approximately $t = 4$. This disparity exists because the extremal systems are not "true" solutions of the system – they only guarantee to bound the solutions. This example points out that over a given time interval, dynamic envelopes do not necessarily provide tighter bounds than does Q2 at every point within the interval.

3.4 Algorithm

Theorem 1 guarantees that if extremal systems can be found for a system A then these extremal systems will bound A, but it does not explain how to generate these systems. This section describes how the dynamic envelope method is implemented, with specific details on the method by which extremal systems are constructed from the QDE.

The steps in constructing the dynamic envelopes for a QDE are

- Find the state variables.
- Construct a general form for the QDE.
- Using this general form, construct the extremal systems.

The state variables are easily identified as the second members of the D/DT constraints. For the purposes of numerical simulation, the derivative of each state variable needs to be defined as a function of state and independent variables only. As the QDE normally includes other "placeholder" variables, it must be "unwound" into a set of general equations. This process starts with the D/DT constraints and replaces each placeholder variable with an expression that contains no placeholders. To simplify the number of cases, QDEs are assumed to use only D/DT, M^+, M^-, MULT, ADD, and MINUS constraints.

In the translation process, MULT and ADD constraints are transformed into division, multiplication, subtraction, and addition equations.

Definition 5 *A* general equation *g is one that includes only state variables, constant variables, M^+ constraints, or M^- constraints, and whose operators are +, - , *, or /. The* general form *G of a QDE is the set of general equations for its state variables.*

Assumption 1 *A QDE can be defined so that it is reducible to a general form.*

3.4.1 Generation of extremal systems

In order to generate bounds for a system, a maximal and a minimal equation for each state variable in the system must be found. The algorithm used to find these systems is described below. It uses the transformation functions $L(e)$ and $U(e)$ which take an expression and return a minimal or maximal expression as defined in Table 3.3. They also return a bound label (designated by a lower or upper bar in the table). In the table, the function $lowval(c)$ returns the lower bound for the variable range of the constant c. This value will be less than or equal to c. The function $upval(c)$ returns the upper bound for the variable range of c. The function $lowenv(g)$ returns the lower envelope of the monotonic function g. We assume that if g is a function of the vector $\mathbf{y}$ then for all $\mathbf{y}$ either $lowenv(g) < g$ or $lowenv(g) = g$, but not $lowenv(g) \leq g$.

To compute the extremal systems, we view the QDE as a digraph whose nodes are the state variables and whose edges are defined by the equations $x'_i = f_i(\mathbf{x}_i)$ such that for each x_i there is an edge from x_i to each $x_j \in \mathbf{x}_i$.

Forming a compatible set of equations amounts to 1) assigning an extremal expression to each node, and 2) labeling it with either an L or a U, but not both. This can be done as follows :

1. Find a node x_i from which all other nodes are reachable[3]. Label x_i with L.

2. Using the rules in Table 3.3, form $L(f_i)$. This will assign an extremal expression to x_i and will label each node adjacent to x_i.

3. Repeat Step 2 for each adjacent node using its label to select the proper column from Table 3.3. Continue this process until all nodes are labeled and have extremal expressions assigned.

4. If at any point, a node receives both an L and a U label then the process fails. This might happen if there are cycles in the graph.

This process produces one extremal system. To produce a complementary system, rerun the algorithm labeling the node in Step 1 with a U rather than an L.

When successfully completed, this process will produce two extremal systems. The process may not complete, however, due to conflicting bounds on some of the variables. In such cases, this method cannot be used. As an example, consider the system

$$\begin{aligned} x' &= v \\ v' &= -x \end{aligned}$$

[3] We assume that the QDE system is not partitioned and so the graph is connected.

e	$L(e)$	$U(e)$
x_h	$\underline{x}_h$	$\overline{x}_h$
c	$lowval(c)$	$upval(c)$
$g(A)$, $g \in M^+$	$lowenv(g)(L(A))$	$upenv(g)(U(A))$
$g(A)$, $g \in M^-$	$lowenv(g)(U(A))$	$upenv(g)(L(A))$
$A + B$	$L(A) + L(B)$	$U(A) + U(B)$
$A \times B$	$L(A) \times L(B)$	$U(A) \times U(B)$
$A - B$	$L(A) - U(B)$	$U(A) - L(B)$
$A \div B$	$L(A) \div U(B)$	$U(A) \div L(B)$
$-A$	$-U(A)$	$-L(A)$

Table 3.3: Transformation rules for generating extremal equations.

Using the above procedure, we conclude

$$\begin{aligned} \underline{x}' &= \underline{v} \\ \underline{v}' &= -\overline{x} \end{aligned}$$

which requires both $\underline{x}$ and $\overline{x}$.[4]

We now show that each equation $y_i = g_i(\mathbf{y_i})$ is an extremal equation as defined in Definition 2.

Theorem 2 *Let A be an ODE system with a state variable x_i whose derivative equation is $x_i' = f_i(\mathbf{x_i})$. Let $y_i' = g_i(\mathbf{y_i})$ be the equation generated by using either the transformation $L(f_i)$ or $U(f_i)$. Then $y_i' = g_i(\mathbf{y_i})$ is an extremal equation for x_i.*

Proof To prove this, we must show that the transformations $L(f_i(\mathbf{x_i}))$ and $U(f_i(\mathbf{x_i}))$ satisfy the three conditions of Definition 2 :

[4]One possible solution to this problem would be to form a fourth order system in $\underline{x}$, $\underline{v}$, $\overline{x}$, and $\overline{v}$. Unfortunately, such a system is unstable and cannot be simulated numerically.

1. For all $x_h \in \mathbf{x_i}$, y_h is a bounded state variable of x_h such that

$$\frac{\partial g_i}{\partial y_h} \begin{cases} \geq 0 & \text{if } bound(y_i) = bound(y_h) \\ \leq 0 & \text{if } bound(y_i) \neq bound(y_h) \end{cases}$$

2. If

   - $\mathbf{y_i}\ \mathbf{R}_=\ \mathbf{x_i}$, and
   - $y_i = x_i$, and
   - $\exists_{x_k \in \mathbf{x_i}}\ y_k \neq x_k$

   then $g_i\ \mathbf{R}\ f_i$,

3. Either $\forall_{\mathbf{x}}\ g_i(\mathbf{x})\ \mathbf{R}\ f_i(\mathbf{x})$ or $\forall_{\mathbf{x}}\ g_i(\mathbf{x}) = f_i(\mathbf{x})$.

We prove the theorem for $L(f_i)$ only. The proof for $U(f_i)$ is similar. We begin by proving the theorem for the "ground" expressions x_h and c.

- If $f_i = x_h$ then $g_i \equiv L(f_i) = \underline{x}_h$. Clearly, $\underline{x}_h$ is a bounded state variable with $\frac{\partial g_i}{\partial \underline{x}_h} \geq 0$. For Condition 2, we consider two cases

  - $h = i$:

    We have either $y_i = x_i$, in which case the third antecedent is false or $y_i \neq x_i$ in which case the first antecedent is false. Therefore Condition 2 is true.

  - $h \neq i$:

    Assume that $y_i = x_i$. We have either $y_h = x_h$, in which case the third antecedent is false or $y_i \neq x_i$ in which case $g_i < f_i$. Therefore Condition 2 is true.

  Condition 3 is trivially met since $f_i(\mathbf{x}) = g_i(\mathbf{x})$.

- If $f_i = c$ then $g_i \equiv L(c) = lowval(c)$. Condition 1 is trivially true as is Condition 2. Since $lowval(c) < c$ or $lowval(c) = c$, Condition 3 is also met.

Now assume that $L(A)(\mathbf{y_i})$ is a minimal expression for $A(\mathbf{x_i})$, $L(B)(\mathbf{y_i})$ is a minimal expression for $B(\mathbf{x_i})$, and $U(B)(\mathbf{y_i})$ is a maximal expression for $B(\mathbf{x_i})$.

If $f_i = f^+(A)$, $f^+ \in M^+$ then $g_i \equiv L(f^+(A)) = lowenv(f^+)(L(A))$. Condition 1 is met since no new state variables are introduced. For any $\mathbf{x_i}$ and $\mathbf{y_i}$ such that $L(A) < A$ then $g_i = lowenv(f^+)(L(A)) < lowenv(f^+)(A) < f_i$ and so Condition 2 is met. Assume that for all $\mathbf{x_i}$ $L(A) < A$. Then $g_i < f_i$. Now assume that for all $\mathbf{x_i}$ $L(A) = A$. Then $g_i < f_i$ if $lowenv(f^+) < g$ and $g_i = f_i$ if $lowenv(f^+) = g$. Hence Condition 3 is satisfied as well.

If $f_i = f^-(A)$, $f^- \in M^-$ then $g_i \equiv L(f^-(A)) = lowenv(f^-)(U(A))$. Condition 1 is met since no new state variables are introduced. For any $\mathbf{x_i}$ and $\mathbf{y_i}$ such $U(A) > A$ then $g_i = lowenv(f^-)(U(A)) < lowenv(f^-)(A) < f_i$ and so Condition 2 is met. Assume that for all $\mathbf{x_i}$ $U(A) > A$. Then $g_i < f_i$. Now assume that for all $\mathbf{x_s}$ $U(A) = A$. Then $g_i < f_i$ if $lowenv(f^-) < g$ and $g_i = f_i$ if $lowenv(f^-) = g$. Hence Condition 3 is satisfied as well.

If $f_i = A+B$ then $g_i \equiv L(A+B) = L(A)+L(B)$. Clearly Condition 1 is met. For any $\mathbf{x_i}$ and $\mathbf{y_i}$ such that $L(A) < A$ and $L(B) < B$ then $L(A) + L(B) < A + B$ and so Condition 2 is met. For any $\mathbf{x_i}$ and $\mathbf{y_i}$ such that $A = L(A)$ and $B = L(B)$ then $L(A) + L(B) = A + B$. If either $L(A) < A$ or $L(B) < B$ then $L(A) + L(B) < A + B$ and therefore Condition 3 is met.

If $f_i = A - B$ then $g_i \equiv L(A - B) = L(A) - U(B)$. Clearly Condition 1 is met. For any $\mathbf{x_i}$ and $\mathbf{y_i}$ such that $L(A) < A$ and $U(B) > B$ then

$L(A) - U(B) < A - B$ and so Condition 2 is met. For any $\mathbf{x_i}$ and $\mathbf{y_i}$ such that $A = L(A)$ and $B = U(B)$ then $L(A) - U(B) = A - B$. If either $L(A) < A$ or $U(B) > B$ then $L(A) - U(B) < A - B$ and therefore Condition 3 is met.

If $f_i = -B$ then $g_i \equiv L(-B) = -U(B)$. Clearly Condition 1 is met. For any $\mathbf{x}_i$ and $\mathbf{y_i}$ such that $U(B) > B$ then $-U(B) < -B$ and so Condition 2 is met. If $B = U(B)$ then $-U(B) = -B$. For any $\mathbf{x_i}$ and $\mathbf{y_i}$ such that $U(B) > B$ then $-U(B) < -B$ and therefore Condition 3 is met.

If $f_i = A \times B$ then $g_i \equiv L(A \times B) = L(A) \times L(B)$. Clearly Condition 1 is met. There is a problem with Conditions 2 and 3. For any $\mathbf{x_i}$ and $\mathbf{y_i}$ such that$L(A) < A$ and $L(B) < B$ then it is possible that $L(A) \times L(B) > A \times B$. However, if we restrict $L(A)$ and $L(B)$ to the positive reals, we have $L(A) \times L(B) < A \times B$. Given this assumption, Condition 2 holds. If $A = L(A)$ and $B = L(B)$ then $L(A) \times L(B) = A \times B$. If either $L(A) < A$ or $L(B) < B$ then, with the above assumptions $L(A) \times L(B) < A \times B$ and therefore Condition 3 is met. The same assumptions will also serve in the proof for $g_i \equiv U(A \times B)$.

If $f_i = A \div B$ then $g_i \equiv L(A \div B) = L(A) \div U(B)$. Clearly Condition 1 is met. Using the assumption that $L(A)$ and $L(B)$ are positive reals it is clear that $L(A) \div U(B) < A \div B$. Given this assumption, Condition 2 holds. If $A = L(A)$ and $B = U(B)$ then $L(A) \div L(B) = A \div B$. If either $L(A) < A$ or $U(B) > B$ then, with the above assumptions $L(A) \div U(B) < A \div B$ and therefore Condition 3 is met. A similar argument holds for $g_i \equiv U(A \div B)$.

□

Because of our restrictions to multiplication and division, we must add the following assumption to our algorithm.

Assumption 2 *For the QDEs under consideration, if A and B are expressions such that $A \times B$ or $A \div B$ is a term of a general expression then $L(A)$ and $L(B)$ must be positive real numbers.*

Under this assumption, the proof of Theorem 2 is complete. In Section ? of Chapter 5 we describe a method for removing this restriction.

3.4.2 Simulation

Given that the assumptions in Theorem 2 are met, the information needed to construct the extremal systems is automatically generated by the dynamic envelope algorithm. What remains is to set the initial conditions for the state variables and then solve the systems with a numerical differential equation solver such as Runge-Kutta or Gear's method [Kahaner *et al.*, 1989]. For non-state variables, equations can be derived that express the variable in terms of state variables and constants and these can be solved at each simulation step. Since the numerical simulation does not rely on any properties of the Q2 solution, it can be run independently of the QSIM algorithm. This means that the numerical simulation can be run selectively while still performing a complete qualitative simulation. The complete simulation algorithm is thus

1. For each initial state of the QDE, generate its extremal systems.
2. Run QSIM and the Q2 algorithm to generate all possible behaviors.
3. Simulate the extremal systems numerically for all initial states.
4. Intersect the Q2 and dynamic envelope bounds for each variable at each simulation timepoint.

3.4.3 Syntax

Because the dynamic envelope method uses exactly the same QDE representation as does Q2, there is very little additional syntax needed to use the method. The examples in Appendix B demonstrate the use of the functions and variables defined below.

QDE slots Since the extremal equations can be automatically extracted, there are no additional variables needed from the QDE. In order to minimize the computation of unnecessary non-state variables, only those mentioned in the `NSIM-PARAMETERS` slot are computed.

Functions There are two procedures that are called to use dynamic envelopes.

`nsim-initialize` *state-or-states*

> Return one or more nsim-structs based on the QDEs used in *state-or-states*. The nsim-struct maintains the information needed to control the simulation. This function is normally called with the initial state list returned by the QSIM function `make-initial-state`.

`nsim` *struct &key :start :stop :sim-step :save-step :sys-select*

> Begin the numerical simulation using the nsim struct *struct* and the specified start, stop, and simulation steps. Save the results every *save-step* time units. *sys-select* is used to prevent the simulation of either the maximum or minimum system. The argument should be a two-element list, the first element corresponding to

the minimal system, and the second element corresponding to the maximal system. A nil value in an element prevents the corresponding system from being simulated.

Control Variables There is also one control variable :

`*use-dynamic-envelopes*` *boolean*

This is set to T when using dynamic envelopes.

3.5 Dynamic envelopes and vacuum systems

The dynamic envelope method was applied to the Q2-augmented vacuum models of Chapter 2. Figure 3.5 shows a composite plot of the envelopes for the A variable. The dynamic envelope method improved the predicted bound for this variable. The plot also shows that after $t = 4$, there is very little overlap between the real leak model and the other two models. Note that this separation of behavior values was not observable in the Q2 models. The nonoverlap of envelopes suggests that it should always be possible to detect a real leak after four minutes. This idea will be pursued in the next chapter.

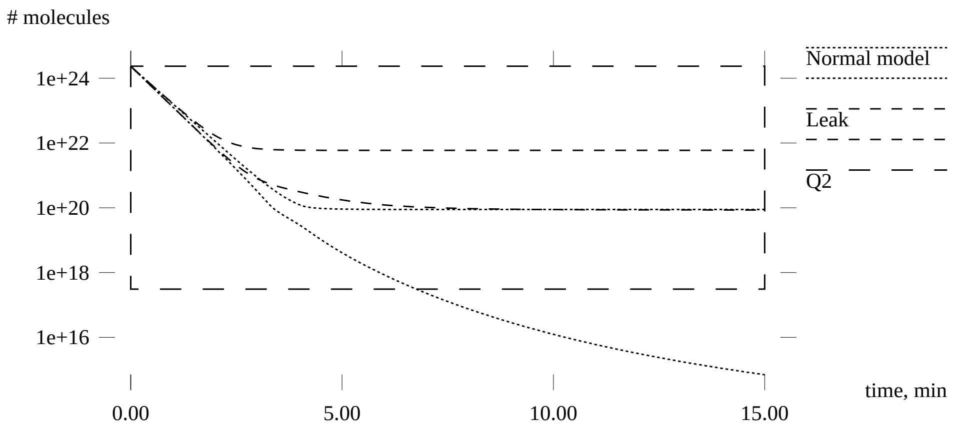


Figure 3.5: Results of dynamic envelope simulation of the A variable for the vacuum models of Chapter 2.

Chapter 4

Qualitative Process Monitoring

4.1 Introduction

Qualitative models can predict the behavior of systems over time. Their ability to simulate inexact models makes them a promising approach for simulating imprecisely characterized processes such as the vacuum system of Chapter 2. Augmenting these models with numeric envelopes not only improves their predictive accuracy but also permits their use in process monitoring applications, where predictions must be numerically compared to sensor data from the process. The MIMIC [Dvorak and Kuipers, 1989] system has been previously developed to monitor continuous processes using qualitative models. This chapter describes process monitoring in general and then describes the application of MIMIC to the vacuum pumpdown task. Finally, it describes a new version of the MIMIC system that uses the dynamic envelope method of Chapter 3 and compares it to the previous version of MIMIC as a means of evaluating the dynamic envelope method.

4.2 Qualitative Process Monitoring

Process monitoring and control systems are of vital importance to applications where a system must continuously operate in the face of changing conditions. Applications such as chemical processing, energy production, aeronautics, and automotive engine control use forms of process monitoring and control. The key elements of such systems are:

- They operate continuously and cannot be stopped.
- Only a subset of the system variables and parameters can be examined.
- The behavior of the system varies continuously.

A general process monitoring and control system must keep a dynamic system running at or near its optimal operating point. To do this, it must perform three tasks:

1. Detection of deviations in process variables from their nominal values. Since some deviations may be too small to matter, a threshold is often set so that deviations smaller than the threshold are ignored. These thresholds are often called *alarm thresholds* because they signal important events.
2. Diagnosis of faults. If a deviation is significant then it is important to determine what caused it. In some systems, the deviation-to-fault mapping is simple and so this task may be relatively easy. For more complex systems, determining the cause of a deviation may be extremely difficult.
3. Control of the system to return it to nominal behavior based on the fault diagnosis. For complex systems with multiple faults, this step may involve a substantial planning effort.

Figure 4.1 illustrates the interaction between these tasks and the underlying process function with which they are associated. All process mon-

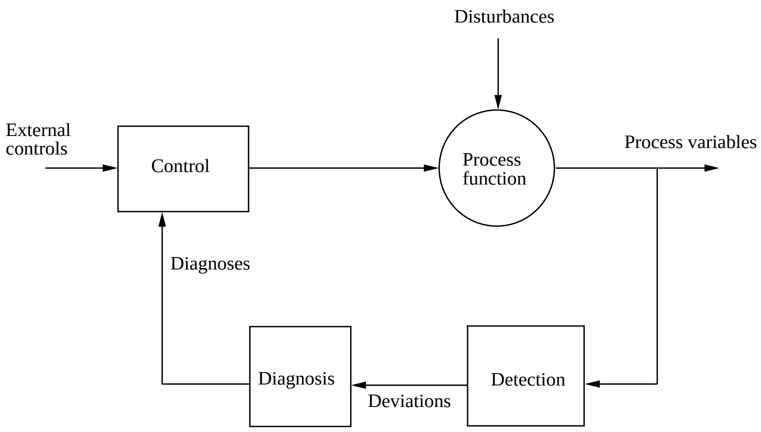


Figure 4.1: A generalized process monitoring and control system.

itoring and control systems perform these three tasks, although the control step is sometimes performed by an outside agent (for instance, human intervention may be required to open a valve or call for service). The process function will normally have one or more operating modes, each corresponding to some steady-state equilibrium point of the system. The task of the process monitoring and control system is to maintain the process function at the equilibrium point in the face of disturbances and to "gracefully" change the system state when the operating mode is changed.

The complexity of the three monitoring and control tasks varies with respect to the complexity of the underlying process function. For traditional chemical process control systems, the detection and diagnosis steps are quite simple. Deviations are determined by computing the difference between the process variables and a fixed set of equilibrium values for the variables called the *setpoint* of the system. Any deviation, no matter how small, is passed on to the control task. The control task, however, is fairly complex in that it must guarantee that the setpoint is maintained in the face of deviations of various magnitudes and that the system moves smoothly and quickly to a new equilibrium state when the setpoint is changed. These control functions are often based on mathematical formulae that take into account the nature of the process, the sensors, and the control elements. For all but the simplest systems, such a control function can be difficult to derive [Franklin, 1986].

For more complex systems, process monitoring and control is more difficult because there are more possible control actions and operating modes. This increased complexity leads to a more complex dynamic response which means that the system is even harder to control when changing operating modes. Furthermore, the number of possible disturbances to the system increases with its complexity (i.e., there are more possible components that can fail) and its coupling to the external environment.

There are several methods for dealing with this added complexity. For some applications, the easiest thing to do is to simply ignore the behavior of the system between equilibrium states. Instead, a predetermined control strategy is used, and once the system settles down to an equilibrium point, monitoring can begin again. This is exactly the strategy used to monitor vacuum pumpdown processes. The system is pumped down for some time (the fixed control strategy) and once it has reached a new equilibrium, the deviation from the desired pressure is measured and a new control action is taken (e.g., continue pumping to maintain the pressure or call the service man). The problem with this approach is, of course, that there are potentially large periods of time over which the system state is not being monitored. Failures during this time may even result in a catastrophic behavior that would go undetected until it was to late to correct.

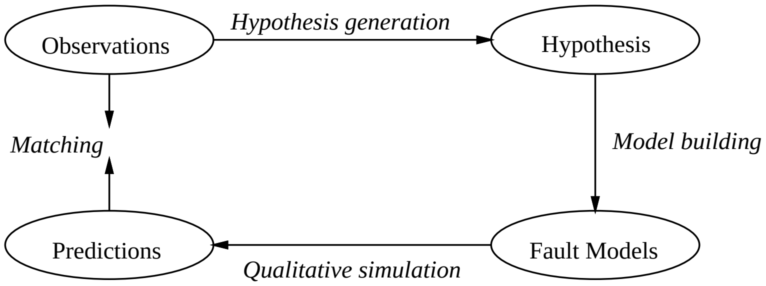


Figure 4.2: The basic cycle of MIMIC.

Another method for monitoring complex systems is to model the behavior of the system and use it to determine thresholds over periods of dynamic behavior so that continuous monitoring is possible. These models could be based on a structural and functional description of the system which is simulated to produce behaviors. The simulated behaviors would then provide nominal values for the observed variables over time. Fault descriptions of the system could also be simulated and used for diagnosis.

This method has been used in a number of systems [Doyle *et al.*, 1989]. One such system is MIMIC. MIMIC does both monitoring and diagnosis by using qualitative models of a process. The basic cycle of the system begins by reading process variables from sensors. On the basis of these values, it selects a set of fault hypotheses with which to construct and initialize a set of Q2-augmented QSIM models. These models are then simulated and compared to the sensor data. Those models whose predictions match the observations are retained and those that do not are discarded as they do not reflect a plausible model for the current state of the system. This process is shown graphically in Figure 4.2.

As new observations are read, the set of existing plausible models (also called the *candidate set*) is tracked by comparing the variable ranges of the current simulation state of each model with the observations. If some variable of the simulation state does not match the observations, then the model is incrementally simulated to the next qualitative timepoint and the variable values of this new state are compared. This process continues up to a fixed number of simulation steps at which point the model is removed from the candidate set if it still does not match the observations. As more observations are taken, the actual behavior of the process becomes better known, and so the diagnosis becomes more precise.

4.3 Monitoring of the Vacuum System

In this section, MIMIC is used to monitor the vacuum system described in Chapter 2. The qualitative models described in Chapter 2 consist of a working model and two fault models. However, as described in Chapter 2, since the virtual leak model is not distinguishable from the normal model in the medium to high vacuum range, it is not considered here. Since there were only two models, the model construction portion of MIMIC was removed and the initial candidate set was defined to include all the models.

One difficulty with simulating a model of a complex system is that the model will never be exact and therefore neither will its predictions. What is needed is a method for producing the tightest bounds possible for the model. By augmenting Q2 with dynamic envelope generation, we improve the model predictions and thus produce better bounds for alarm thresholds.

To test the utility of dynamic envelopes for this purpose, a modified version of MIMIC was constructed that could incrementally simulate the dynamic envelopes for a model as new sensor data was read. This version otherwise functioned identically to the first system. Figure 4.3 is a block

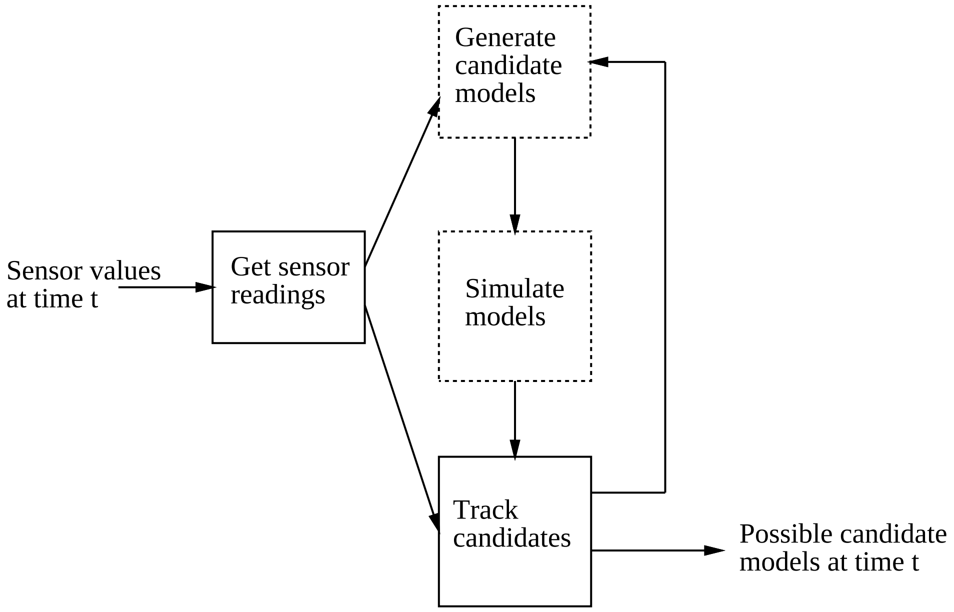


Figure 4.3: Block diagram of the MIMIC system. Boxes with broken lines indicate modules changed from the standard system.

diagram of the modified system used here.

The monitoring systems were tested with two datasets – one from a working system and one from a system with a real leak of conductance $5\pi \cdot 10^{-4}$ liters/sec[1]. Because there was no experimental data for the faulted system, the numerical models described in Chapter 2 were used to generate the data.

Table 4.1 summarizes the results of the tests. The final candidate sets of both systems included the "no faults" model when tested on the normal dataset, indicating that the data is consistent with a working system. However, whereas the standard MIMIC system was unable to eliminate ei-

[1]This corresponds to a gasket leak caused by a roughly ground seal in a pipe of diameter 5 cm.

Dataset	MIMIC	Dynamic envelope MIMIC
Normal	No models eliminated	REAL-LEAK eliminated at $t = 4$
Real leak	REAL-LEAK at $t = 9$	REAL-LEAK at $t = 4$

Table 4.1: Final candidate sets for each test run. The time that the final candidate set was determined is also included for cases where the candidate set shrunk.

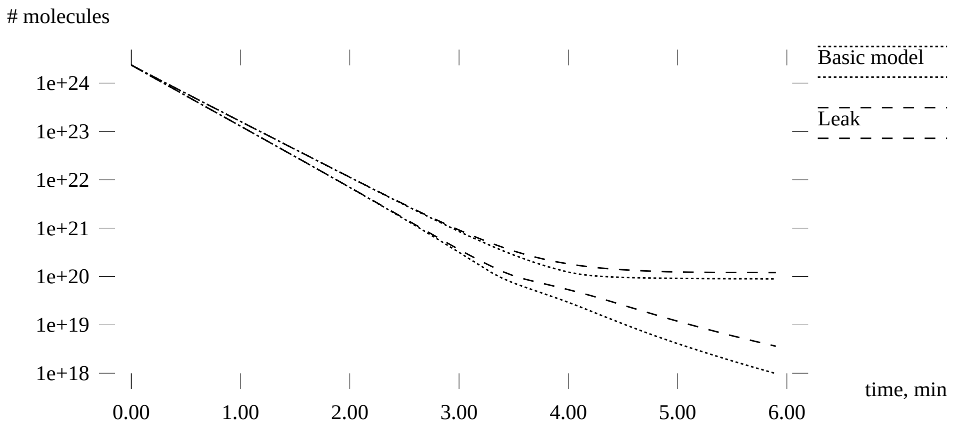
ther fault model, the dynamic envelope version was also able to eliminate the possibility of a real leak. The dynamic envelope method therefore improved the preciseness of the diagnosis for this dataset.

Both systems unambiguously identified the real leak, although the unimproved system took twice as long to detect it as the dynamic envelope system because the qualitative distinctions between behaviors occurred after a quantitative distinction. In this case, the dynamic envelope method shortened the time to failure detection.

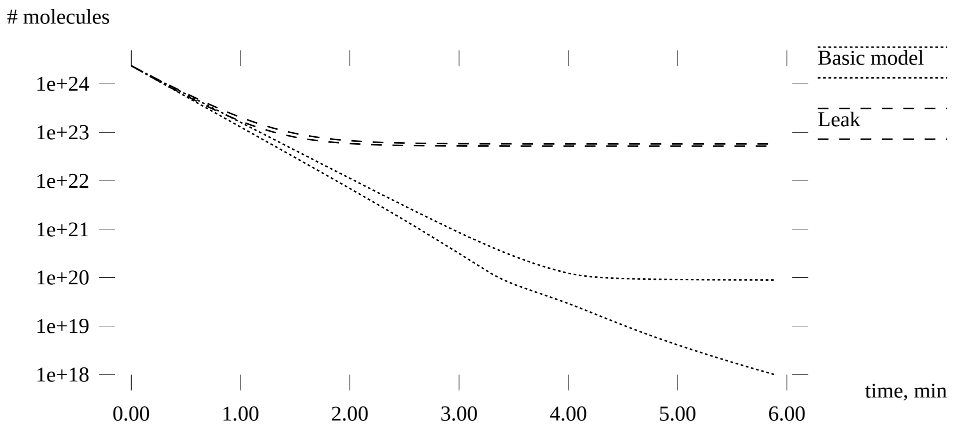
This improvement is in part due to the magnitude of the hypothesized leak in the leak fault model. Figure 4.4 shows that for a leak that is an order of magnitude smaller, the two models overlap more. In this case, it would be more difficult to rule out either model given a particular set of sensor data. Figure 4.4b shows that for a leak that is an order of magnitude greater, the envelopes become distinct even earlier.

4.4 Recomputation of Envelopes Based on Sensor Data

The envelopes can be further tightened by reevaluating the envelope computations after each measurement. The idea here is that sensor data for model variables can further pin down the state of the system at the time of the measurement. Since the QDE is autonomous (i.e., the derivative functions



(a) Leak conductance [0.001 0.001]



(b) Leak conductance [0.1 0.1]

Figure 4.4: The effect of leak conductance on envelope predictions (compare with Figure 3.5).

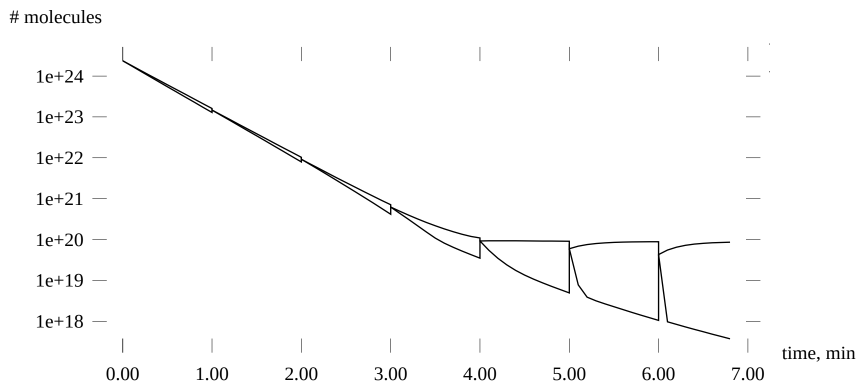


Figure 4.5: Envelopes for the A variable with recomputation of envelopes after every sensor reading.

are independent of time), the simulation can be restarted using any available sensor data as an initial condition for the corresponding model variable. If no sensor data is available for a variable then the range predicted by the simulation so far is used. Note that because the simulation assumes that all variables are ranges, sensor error can be modeled by "blurring" the measurements so that they become a range. Figure 4.5 demonstrates the use of this method on the working vacuum model. Note that although the predicted envelope tightens immediately after a new reading, it quickly returns to the range predicted by the original simulation. This is due to both the extreme stiffness of the vacuum system model equations and the fact that there is no data for the B variable (the gas dissolved in the wall). Nevertheless, these new envelopes do provide a tighter bound on the behavior and therefore represent an improved alarm threshold for the A variable.

Chapter 5

Conclusions and Further Work

5.1 Summary of Research Conducted

This thesis described the construction of qualitative models using QSIM for multi-tank systems and the application of these models in monitoring and diagnosis systems. First, models for some simple tank configurations (the bathtub, two cascaded tanks, and the u-tube) were presented. These models demonstrated that useful simulation results could be obtained for systems that are inexactly defined. Next, as an example of a real-world application, a vacuum system of the type that is used in semiconductor processing was qualitatively modeled. This type of system is difficult to model mathematically because the exact relations between model variables are seldom known. A model of the working system, a system with a real leak, and a system with a virtual leak were constructed.

While the qualitative models gave correct behaviors for these systems, their predictions were fairly weak, determining only the general shape of the variable curves. Furthermore, the QSIM modeling language did not provide a way to relate the rates of different monotonic constraints. To overcome these problems, the models were recast using the Q2 system which augments qualitative reasoning with partial numerical information about monotonic constraints and variable values. When simulated with Q2, the additional quantitative information reduced the number of possible behaviors for some of the models. In addition, Q2 included numerical ranges in its prediction of variable values.

The Q2 algorithm relies on range propagation and the mean value theorem to make predictions across time intervals. This method, while sufficient to drive the range propagation method, results in fairly weak predictions across time intervals. As a means of tightening these predictions, a new method for variable range prediction was described called the *dynamic envelope method*. With this method, ordinary differential equations are generated based on the QDE and are simulated numerically to get bounds on the state variables of the model. These values can then be used to determine the bounds on the non-state variables. The dynamic envelope method works with the Q2 algorithm and uses the same information as Q2 uses. The tank models were simulated using dynamic envelopes and in all cases, the predictions were tighter than with Q2 alone.

The Q2 system generates predictions with numeric ranges. The existence of numeric ranges permits the qualitative models to be used in monitoring tasks as the predictions can be compared against sensor data. MIMIC is an example of a system that uses qualitative models in this way. It constructs a set of possible models for a process, retaining only those that are consistent with observations. Consistency is determined by simulating the possible models using Q2 and comparing the simulated results to the observations. Another version of MIMIC was constructed that used the dynamic envelope method to tighten predictions. Both systems were run on two sets of test data for the vacuum system. In both cases, the dynamic envelope version outperformed the standard version either in accuracy of diagnosis or in speed of problem detection.

5.2 Limitations of the Dynamic Envelope Method

In Chapter 3, the following limitations of the dynamic envelope method were discussed:

- When the extremal system generation process successfully completes the system can be simulated. However, it is not always the case that this process will complete. For instance, for the system $x' = v$, $v' = -x$, the extremal system which contains $\underline{x}$ is $\underline{x}' = \underline{v}$, $\underline{v}' = -\overline{x}$ which contains both $\underline{x}$ and $\overline{x}$ terms. This type of equation set is common to systems that have oscillatory behavior.

- Because multiplication is nonmonotonic over the real numbers, multiplication of two lower bounds which are negative may result in an upper bound for the product.

- The bounds predicted by the dynamic envelope method are not always smaller than the Q2 predictions. This is because the extremal systems do not represent the same system as the QDE – they simply represent bounding solutions.

The third limitation is addressed by intersecting the Q2 and the dynamic envelope predictions. The resulting prediction is then at least as tight as the original Q2 prediction.

The second limitation is also easy to fix. Since multiplication is a monotonic function over a given quadrant of the real plane and the QSIM behavior description has the property that all signs are uniform over any time interval, the simulation method can detect when the behavior is crossing a boundary and can recompute a new set of equations based on the new quadrant.

The first limitation is a subject for active research.

5.3 Related Work

There has been considerable interest in the combination of qualitative and quantitative reasoning. This work includes the development of combined qualitative representations (see [deKleer, 1977, Simmons, 1983, Sacks, 1987, Cheng and Stephanopoulos, 1988, Kuipers and Berleant, 1988, Williams, 1988, Yip, 1988, Karp and Friedland, 1989, Skorstad and Forbus, 1989]) and the use of numerical and qualitative knowledge for process monitoring [Dvorak and Kuipers, 1989] and process planning [Fusillo and Powers, 1988, Lakshmanan and Stephanopoulos, 1988, LeClair and Abrams, 1988]. The methods and software described in [Kuipers and Berleant, 1988] and [Dvorak and Kuipers, 1989] (Q2 and MIMIC) are integral parts of this research. Recently Berleant and Kuipers have extended Q2 to provide a single representation for both qualitative and quantitative simulation [Berleant, 1989, Berleant and Kuipers, 1990]. In their method, called Q3, the range of a qualitative variable is narrowed through an adaptive discretization technique that subdivides qualitative intervals. As the intervals are further subdivided, the ranges shrink and the accuracy of the prediction increases. In contrast, dynamic envelopes use direct numerical simulation. In particular, for an accurate set of envelope functions, Q3 can provide an arbitrary level of accuracy by subdividing to a fixed level of granularity. To reach a very fine level may require many subdivisions. Dynamic envelopes will provide a fine level of accuracy with a fixed simulation cost, but only over fixed period of time.

This research also relates to the measurement interpretation theories ATMI [Forbus, 1986] and DATMI [DeCoste, 1990]. Both of these methods abstract a measurement stream into qualitative values and then select possible behaviors by comparing measurement segments to states in the total envisionment graph. By hypothesizing measurement errors, DATMI also manages to interpret noisy sensor data. By contrast, the dynamic envelope method augments the qualitative behavior with numerical envelopes that are guaranteed to bound any solution of the system and then compares the measurement data directly. This approach has the advantage that distinctions between models can be detected over intervals where their qualitative behaviors are identical. Furthermore, by recomputing the envelopes as new measurements are received, the bounding solutions can be further tightened. Measurement faults can also be modeled by assuming that the measurement data itself represents a range rather than a precise point.

The work on SIMGEN as described in [Forbus and Falkenhainer, 1990] is also related to the work described in this thesis. It, too, generates a standard numerical simulation by extracting the relevant information from a qualitative model. It differs in that it generates an exact numerical model based on a library of predefined functions rather than generating a bounded model expressing the inexactness of the qualitative model. As such, it is not particularly suited to tasks such as process monitoring in which an exact numerical model cannot be found.

5.4 Conclusions

This research has underscored the following points of previous work

:

- Qualitative models are useful for representing and simulating systems which are inexactly defined. With the addition of partial qualitative reasoning, numerical results can be gathered even though the model is not numerically precise.

- In process monitoring and control tasks, qualitative models can be used to provide a method for dynamically determining variable alarm thresholds and to classify anomalous behavior by simulating fault models.

This work has also contributed the following results :

- Qualitative simulation is a tractable method for simulating aspects of vacuum system dynamics.

- The dynamic envelope method combines qualitative and quantitative simulation so that both representations can be used in problem solving. It produces all behaviors associated with a particular model, and it provides detailed numerical ranges for each behavior. Because the qualitative simulation guides the use of numerical simulation so that only plausible sets of initial conditions are simulated, needless simulations are not run. Furthermore, the dynamic envelope simulation can be conditionally applied to a subset of the behaviors of a model. This means that its use can be limited as computational resource demands change. Because the envelope systems are automatically generated from Q2-augmented QDEs, the method can be used with any existing Q2 model.

- In monitoring tasks, the dynamic envelope method improves the predictive power of Q2-augmented qualitative models both in accuracy (meaning that fault hypotheses can be more easily eliminated) and failure detection time (meaning that there is more time to recover from failures).

In cases where measurement acquisition is expensive, the increased accuracy of the predictions may allow fewer measurements to be made and errors to be detected sooner.

Although the vacuum systems described in this thesis used qualitative simulation to predict behaviors, they do not demonstrate the full power of the qualitative method. One advantage of qualitative reasoning is that it can simulate all possible behaviors of a system that are consistent with the given constraints. Further, systems which exhibit discontinuities can also be simulated using operating region transitions as was shown in the two-tank cascade. Since the vacuum system models had only one behavior each and did not require operating region transitions, the advantages of qualitative simulation over numerical simulation are not as clear as they would be with a highly dynamic system. Another problem with the vacuum system is that it is extremely "stiff" and is therefore difficult to simulate with a simple differential equation solver such as Runge-Kutta. This limited the type of system that could be simulated.

5.5 Further Work

The integration of qualitative and quantitative reasoning methods is an area with many interesting problems. I focus here on extensions to the dynamic envelope method as described in this thesis as well as on the effects of this method on systems monitoring.

- The types of systems that the method applies to could be enlarged. As mentioned above, the unrestricted use of multiplication is simple to implement. Problems with oscillatory systems may be more difficult.

One approach is to use more of the qualitative information about the behaviors to guide the construction of extremal systems.

- Region transitions should be integrated into the method.

- A method for tightening the predictions of the method is needed. At present, the dynamic envelope predictions are sometimes wider than the Q2 predictions.

- Presently, the dynamic envelope method uses the results of the Q2 algorithm but the reverse is not true. By integrating the two, dynamic envelopes could be used to further reduce the ranges used by Q2 and perhaps eliminate more behaviors. The combination of the two methods could also be used to infer trend information about behaviors for times beyond the end of the dynamic envelope simulation interval.

Appendix A

Vacuum Terms and Equations

This appendix explains a number of terms and equations from the field of vacuum engineering and kinetic gas theory. More details on these terms can be found in [Duval, 1988], [Lewen, 1965], and [Roth, 1976].

Pressure The pressure of a gas is a measure of the impulse of the gas molecules hitting the walls. Pressure is measured in a variety of units. The most common ones used in vacuum engineering are torr, millibars, and microns. The relation between these units is

$$1\text{torr} = 1.33 \text{ millibar} = 1000 \text{ microns}$$

Atmospheric pressure (more correctly termed *standard atmospheric pressure*) is defined to be 760 torr. For vacuum applications, pressures are grouped into four ranges. These ranges are shown in Table A.1.

Range	Pressure (torr)
Low and medium vacuum	760 to 10^{-3}
High vacuum	10^{-3} to 10^{-5}
Very high vacuum	10^{-6} to 10^{-8}
Ultra-high vacuum	$< 10^{-9}$

Table A.1: Degrees of vacuum and pressure ranges

Sorbtion This term refers to the process by which gas or vapor is taken up by a solid. There are two types of sorption; *absorption*, in which the gas dissolves into the interior structure of the solid, and *adsorption* in which the gas adheres to the surface of the solid. These two processes have different mechanisms and are therefore distinct. In vacuum engineering, the removal of sorbed gases from chamber walls is critical to proper operation.

Conductance This measures the ability of a pipe or orifice to permit gas flow. Conductance in gas flow is analogous to conductance in electrical flow. It is measured in liters/sec.

Throughput The throughput of a pump measures the flow rate of gas through the pump. It is measured in micron-liters/sec. Its equation is

$$throughput = pumpspeed \times pressure$$

R This is known as the *gas constant*. It is used to convert between the number of moles of a gas and its pressure via the formula $PV = nRT$. For pressure in microns, volume in liters, and temperature in degrees Kelvin, $R = 62364$ micron-liters/degK-mole.

N_A This is Avagadro's number. It measures the number of molecules of a gas in one mole. $N_A = 6.0235 \times 10^{23}$.

Appendix B

Qualitative Models and Initializations

The QSIM system used in this work is written in Common LISP running on Texas Instruments Explorer and Symbolics 3600 machines. Specifically, the examples in this thesis were run using an experimental version of QSIM called NQ with release date March, 1990. The public release of QSIM can be obtained by request form Professor Benjamin Kuipers, Department of Computer Sciences, The University of Texas at Austin, Austin, TX, 78712 (kuipers@cs.utexas.edu).

B.1 Simple Tank Models

In Chapters 2 and 3, purely qualitative, Q2-augmented, and dynamic envelope qualitative models were developed for the bathtub and the two-tank cascade. As an aid to understanding the use of these methods, the complete Q2-augmented model of the two-tank cascade is presented together with initialization code. Statements beginning with a comment character (";") indicate statements that are included when performing dynamic envelope simulation.

```
(define-QDE Cascaded-tanks-w-ranges
  (text "Two cascaded tanks")
  (quantity-spaces
    (inflowa    (0 ifa* inf))
    (amounta    (0 inf))
```

```
    (outflowa   (0 inf))
    (netflowa   (minf 0 inf))
    (amountb    (0  inf))
    (outflowb   (0 inf))
    (netflowb   (minf 0 inf)))
  (constraints
    ((M+ amounta outflowa) (0 0) (inf inf)   )
    ((ADD outflowa netflowa inflowa))
    ((d/dt amounta netflowa))
    ((m+ amountb outflowb) (0 0) (inf inf)   )
    ((add outflowb netflowb outflowa))
    ((d/dt amountb netflowb)))
  (independent inflowa)
  (other
    (ignore-qdirs netflowb)
    (m-envelopes
      ((M+ amountA outflowA)
          (upper-envelope (lambda (x) x))
          (upper-inverse  (lambda (y) y))
          (lower-envelope (lambda (x) (if (< x 20) (/ x 2)
                                                   (+ (/ x 5) 6))))
          (lower-inverse  (lambda (y) (if (< y 10) (* y 2)
                                                   (* 5 (- y 6))))))
      ((M+ amountB outflowB)
          (upper-envelope (lambda (x) x))
          (upper-inverse  (lambda (y) y))
          (lower-envelope (lambda (x) (if (< x 20) (/ x 2)
                                                   (+ (/ x 5) 6))))
          (lower-inverse  (lambda (y) (if (< y 10) (* y 2)
                                                   (* 5 (- y 6)))))
  ))
    (initial-ranges ((inflowA ifa*) (5 10))
                    ((time t0)      (0 0)))))


(defun nsim-fill-cascade-from-empty ()
  (let ((*check-quantitative-ranges* t)
;       (*use-dynamic-envelopes* T)
        (init (make-initial-state cascaded-tanks-w-ranges
                                  '((inflowa (ifa* std))
                                    (amounta (0 nil))
                                    (amountb (0 nil)))
                                  "Fill from empty")))
;    (nsim-init init)
```

```
    (qsim init)
;    (nsim (state-nsim-struct init))
    (qsim-display init)))
```

B.2 Vacuum Models

In Sections 2.4 and 2.5.1, qualitative models for a vacuum system were developed. This section presents the complete Q2-augmented models for these systems. Comment lines indicate statements that are added or deleted to the basic model to form the fault models.

B.2.1 Constants

```
;;;-----------------------------------------------------------------
;;;
;;; Problem constants and functions
;;;
;;;-----------------------------------------------------------------

;;; Note that all pressures are in microns, volumes are in liters,
;;; times are in minutes, distances are in cm, temperatures are in
;;; Kelvin.

;;; Gas constant (in micron-liters/deg-gram_mole)
(defparameter R 62364.)

;;; Molecular weight of gas.
(defparameter mass 30.)

;;; Energy of desorption.
(defparameter Edsorp 20.)

;;; Temperature (deg Kelvin)
(defparameter temp 280.)

;;; Volume of the vacuum chamber (liters)
(defparameter volume 90.)

;;; Surface area of the vacuum chamber (sq cm)
```

```
(defparameter area 13800.)
(defparameter area_h 14500.)
(defparameter area_l 13100.)

;;; Avagadro's number (#molecules/gram_mole)
(defparameter Na 6.0235e23)

;;; A constant needed to convert from # molecules to pressure.
(defparameter M (/ (* R temp) (* Na volume)))

;;; A constant needed to convert from throughput in ul/sec to
;;; # molecules/sec
(defparameter Mq (/ Na (* R temp)))

;;; Soujourn time of a molecule
(defparameter ts (* 1.e-13 (exp (/ Edsorp (* .002 temp)))))

;;; Saturation pressure in microns.
(defparameter psat 76.)

;;; Max molecules adsorbed into the wall of the chamber.
(defparameter bmax (* 3 area Mq))
(defparameter bmax_h (* 3 area_h Mq)) ; 1
(defparameter bmax_l (* 3 area_l Mq)) ; 5

;;; Initial number of molecules in A
(defparameter Ainit (* (/ volume (* 0.082 temp)) Na))

(defun pressure (A)
  (* M A))

;;; Room pressure
(defparameter proom (pressure Ainit))

;;; C doesn't vary
(defparameter c_h (/ bmax_h
                     (/ (* ts 3.5e19 area_h proom)
                        (sqrt (* mass temp)))))
(defparameter c_l c_h)


;;; Leak conductances for a real leak.
(defparameter cleakupper 0.01)
(defparameter cleaklower 0.001)
```

```
;;; Leak conductances for a virtual leak.
(defparameter cvleakupper 1.e-5)
(defparameter cvleaklower 1.e-7)

;;; Volume of the virtual leak.
(defparameter vleakvolume (* 10.0 0.001))

;;; 'M' constant for the virtual leak.
(defparameter Mleak (/ (* R temp) (* Na vleakvolume)))
```

B.2.2 Models

```
(define-QDE q2-vac-sys
  (text "A vacuum system.")
  (quantity-spaces
              (p         (0 poff proom))
       (A         (0 Aoff Aroom))
       (dA        (minf 0 inf))
       (B         (0 bmax))
       (area      (0 area0))
       (theta     (0 one))
       (pumptp    (0 inf))
       (pumptp/mq (0 inf))
       (mq        (0 mq0))
       (speed     (0 speed0))
       (flowAB    (minf 0 inf))
;;            This variable is only for the normal model.
;        (mflowAB   (minf 0 inf))
       (alpha     (0 alpha0 alphamax)) ; 0 needed for mult constraint
       (f*alpha   (0 f0*alpha0 inf))
       (f         (0 f0))
       (desorbrt  (0 g0 inf))
       (adsorbrt  (0 f0*alpha0*a inf))
       (m         (0 m0))
       (bmax      (0 bmax))
;;            Add these variables for the leak model only.
;             (proom     (0 proom))
;             (dLeak     (minf 0 inf))
;             (flowAB+pumptp (minf 0 inf))
;             (proom-p   (0 inf))
;;            Add these variables for the virtual leak model only.
;             (Leak      (0 Leakoff Leakroom))
```

```
;               (Mleak     (0 mleak0))
;               (pleak     (0 poff proom))
;               (dLeak     (minf 0 inf))
;               (pLeak-p   (minf 0 inf))
;               (flowAB+pumptp (minf 0 inf))
;               (mdLeak    (minf 0 inf))
       )

  (constraints
       ((d/dt A dA))
       ((d/dt B flowAB))
       ((mult theta bmax B) (one bmax bmax))
       ((mult speed p pumptp/Mq))
       ((mult pumptp/Mq Mq pumptp))
       ((M+ p speed) (poff 0) (proom speed0))
       ((mult m A p) (m0 Aoff poff)
             (m0 Aroom Proom))
       ((M+ p f) (0 0) (proom f0))
       ((mult f alpha f*alpha) (f0 alpha0 f0*alpha0))
       ((mult f*alpha area adsorbrt) (f0*alpha0 area0 f0*alpha0*a))
       ((add desorbrt flowAB adsorbrt) (g0 0 f0*alpha0*a))
       ((M+ B desorbrt) (0 0) (Bmax g0))
       ((M- theta alpha) (0 alphamax) (one alpha0))
;;              Add these constraints for the normal model only.
;               ((add pumptp dA mflowAB))
;               ((minus mflowAB flowAB))
;;              Add these constraints for the real leak model only.
;               ((add flowAB+pumptp dA dleak))
;               ((add flowAB pumptp flowAB+pumptp))
;               ((add proom-p p proom) (0 proom proom))
;               ((M+ proom-p dleak) (0 0) (inf inf))
;;              Add these constraints for the virtual leak model.
;               ((mult mLeak Leak pLeak) (mleak0 Leakroom proom)
;                                         (mleak0 leakoff poff))
;               ((add p pLeak-p pLeak) (proom 0 proom)
;                                       (poff 0 poff))
;               ((minus dleak mdleak))
;               ((d/dt Leak mdLeak))
;               ((M+ pleak-p dLeak) (0 0) (inf inf))
;               ((add flowAB pumptp flowAB+pumptp))
;               ((add dA flowAB+pumptp dLeak))
       )
  (independent m mq bmax area)
```

```
  (other
;; This line is for the normal model.
;  (ignore-qdirs dA flowAB mflowAB f*alpha adsorbrt))
;; This line is for the real leak model.
;  (ignore-qdirs dA flowAB f*alpha adsorbrt flowAB+pumptp)
;; This line is for the virtual leak model.
;  (ignore-qdirs dA flowAB f*alpha adsorbrt flowAB+pumptp
;                dLeak mdLeak pLeak-p)
  (m-envelopes
   ;; The upper envelope starts at 265, bends at 40 and hits 0
   ;; at 0.1.  The lower envelope starts at 245, bends at 40 and
   ;; hits 0 at 10.
   ((M+ p speed)
    (upper-envelope (lambda (x)
      (if (> x 40.)
  (+ (/ (- (* 5. x) 200.) (- proom 40.)) 260.)
  (- (* (/ 260. 39.9) x) (/ 26. 39.9)))))
    (upper-inverse  (lambda (y)
      (if (< y 260.)
  (* (+ y (/ 26. 39.9)) (/ 39.9 260.))
  (* (+ y -260. (/ 200. (- proom 40.)))
     (/ (- proom 40.) 5.)))))
    (lower-envelope (lambda (x)
      (if (> x 40.)
  (+ (/ (- (* 5. x) 200.) (- proom 40.)) 240.)
  (- (* (/ 240. 30.) x) (/ 2400. 30.)))))
    (lower-inverse  (lambda (y)
      (if (< y 240.)
  (* (+ y (/ 2400. 30.)) (/ 30. 240.))
  (* (+ y -240. (/ 200. (- proom 40.)))
     (/ (- proom 40.) 5.))))))

   ;; This function is completely dictated by the laws of gas
   ;; kinematics.
   ((M+ p f)
    (upper-envelope (lambda (p)
      (/ (* 60. 3.5E19 p) (sqrt (* mass temp)))))
    (upper-inverse  (lambda (f)
      (/ (* f (sqrt (* mass temp)))
 (* 60. 3.5E19))))
    (lower-envelope (lambda (p)
      (/ (* 60. 3.5E19 p) (sqrt (* mass temp)))))
    (lower-inverse  (lambda (f)
      (/ (* f (sqrt (* mass temp)))
```

```
 (* 60. 3.5E19)))))

   ;; This function is completely dictated by the laws of gas
   ;; kinematics.
   ((M+ B desorbrt)
    (upper-envelope (lambda (B) (/ (* 60. B) ts)))
    (upper-inverse  (lambda (g) (/ (* G ts) 60.)))
    (lower-envelope (lambda (B) (/ (* 60. B)  ts)))
    (lower-inverse  (lambda (g) (/ (* G ts) 60.))))

   ;; This measures the sticking coefficient.
   ((M- theta alpha)
    (upper-envelope (lambda (th)
      (- (* 2. (exp (* (log c_h) th)))
 c_h)))
    (upper-inverse  (lambda (a)
      (/ (log (/ (+ a c_h) 2.)) (log c_h))))
    (lower-envelope (lambda (th)
      (/ (+ (exp (* (log c_l) th)) c_l) 2.)))
    (lower-inverse  (lambda (a)
      (/ (log (- (* 2. a) c_l)) (log c_l)))))
;;        This envelope is only for the real leak model.
;         ((M+ proom-p dleak)
;          (upper-envelope (lambda (proom-p)
;                            (* 60 cleakupper proom-p Mq)))
;          (upper-inverse  (lambda (dleak)
;                            (/ dleak (* 60 cleakupper Mq))))
;          (lower-envelope (lambda (proom-p)
;                            (* 60 cleaklower proom-p Mq)))
;          (lower-inverse  (lambda (dleak)
;                            (/ dleak (* 60 cleaklower Mq)))))
;;        This envelope is only for the virtual leak model.
;         ((M+ pleak-p dleak)
;          (upper-envelope (lambda (pleak-p)
;                            (* 60 cvleakupper pleak-p Mq)))
;          (upper-inverse  (lambda (dleak)
;                            (/ dleak (* 60 cvleakupper Mq))))
;          (lower-envelope (lambda (pleak-p)
;                            (* 60 cvleaklower pleak-p Mq)))
;          (lower-inverse  (lambda (dleak)
;                            (/ dleak (* 60 cvleaklower Mq)))))
   )

 (initial-ranges ((B bmax)    (bmax_l bmax_h))
```

```
 ((A Aroom)   (Ainit Ainit))
 ((theta one) (1. 1.))
 ((area area0) (area_l area_h))
 ((m m0)      (M M))
 ((mq mq0)    (Mq Mq))
 ((bmax bmax) (bmax_l bmax_h))
;;                       These ranges for the real leak only.
;                        ((proom proom) (proom proom))
;;                       These ranges for the virtual leak only.
;  ((mleak mleak0) (Mleak Mleak))
;  ((Leak Leakroom) ((* proom (/ 1.0 mleak))
;                                                 (* proom (/ 1.0 mleak))))
 ((time t0)   (0 0))
 )
  (layout (p theta flowAB pumptp mflowAB)
  (speed alpha dA f*alpha nil)
  (adsorbrt desorbrt nil nil nil)
  (Time A B f nil)
  )
  )


(defun q2-vac-sys ()
  (let ((*state-limit* 50)
        (*perform-acc-analysis* nil)
        (*check-quantitative-ranges* t)
;;      Include this if using dynamic envelopes
;       (*use-dynamic-envelopes* T)
        (initial-state
          (make-initial-state q2-vac-sys
                              '((A        (Aroom nil))
                                (B        (Bmax nil))
                                (m        (m0 std))
                                (mq       (mq0 std))
                                (bmax     (Bmax std))
;;      Add this value for leak model
;                               (proom    (proom std))
;;      Add these values for virtual leak model
;                               (leak     (Leakroom nil))
;                               (mleak    (mleak0 nil))
                                )
                              "Pumping out the volume")))
;;   Include this if using dynamic envelopes
;    (nsim-init initial-state)
```

```
;    (format t "~%Initialized")
    (qsim initial-state)
;;   Include this if using dynamic envelopes
;    (nsim (state-nsim-struct initial-state))
    (qsim-display initial-state)
    ))
```

Appendix C

Numerical Models for the Vacuum System

In Sections 2.4.3 and 2.4.4, numerical models of the vacuum system were constructed. This section presents the actual models used. The models were originally constructed using the STELLA system [Richmond *et al.*, 1987] on a Macintosh II. Later, they were recoded into FORTRAN-77 and run on an HP 9000 computer so that different simulation algorithms could be compared. The example program presented here uses the fourth-order Runge-Kutta algorithm RK45 described in [Forsythe *et al.*, 1977]. The models were also implemented using Gear's method [Kahaner *et al.*, 1989] for simulation. Gear's method proved to be more accurate and faster than the Runge-Kutta methods because of the stiffness of the vacuum model.

```
      call initcommon
      call rootcomputemotion(14.0, 0.1, 'q2-vac-sys.rk45')
      end

      subroutine initcommon
      real Ainit, Binit, Ed, proom, M, MQ, area, mass, Na, R, temp,
     A     volume, ts, area_l, area_h, bmax_l, bmax_h, c_l, c_h
      common Ainit, Binit, Ed, proom, M, MQ, area, mass, Na, R, temp,
     A       volume, ts, area_l, area_h, bmax_l, bmax_h, c_l, c_h

      area = 13800.
      area_l = 13100.
      area_h = 14500.
      Ed = 20.
      mass = 30.
      Na = 6.0235e23
```

```
      R = 62364.
      temp = 280.
      volume = 90.

      M = R*temp/(Na*volume)
      MQ = Na/(R*temp)
      Ainit = (volume/(0.082*temp))*Na
      Binit = 3.*area*MQ
      bmax_l = 3.*area_l*MQ
      bmax_h = 3.*area_h*MQ
      proom = Ainit*M
      ts = 1e-13*exp(Ed/(.002*temp))
      c_l = bmax_h/(ts*3.5e19*area_h*proom/sqrt(mass*temp))
      c_h = c_l
      return
      end


***-----------------------------------------------------
*** Compute the behavior of the system described in
*** the subroutine eqnsys.
*** Inputs: timeout - the stop time of the simulation.
***         dt      - simulation step.
***         file    - output file.
***-----------------------------------------------------
      subroutine rootcomputemotion(timeout, dt, file)
      real timeout, dt
      character file *15

C     Change n to 3 for the virtual leak model
      parameter (n=2, mint=2, nroot=0, LW=3+6*n)
      real tout, ewt, eps, y(n), WORK(LW)
      integer IFLAG, IWORK(5)
      external eqnsys

      real Ainit, Binit, Ed, proom, M, MQ, area, mass, Na, R, temp,
     A       volume, ts, area_l, area_h, bmax_l, bmax_h
      common Ainit, Binit, Ed, proom, M, MQ, area, mass, Na, R, temp,
     A       volume, ts, area_l, area_h, bmax_l, bmax_h, c_l, c_h

      real theta, p, alpha, flowab
      pressure(A) = M*A

      t = 0.
```

```
      tout = t

      ABSERR = 0.0
      RELERR = 1.e-6
      ewt = 0
      eps = 0.0001

      y(1) = Ainit
      y(2) = Binit
C     Add these lines for the virtual leak model
C      leakvol = .001*10.
C      y(3) = proom*MQ*leakvol

      IFLAG = 1

      print 5
 5    format (1X, "time", 5X, "A", 5X, "B", 5X, "p", 5X, "ms")
      open (unit=1, file=file//'a', iostat=iostat)
      if (iostat .NE. 0) then
         print *, "Trouble opening ", file//'a', ".  Error code",
     A             iostat
         stop
      endif
      open (unit=2, file=file//'b', iostat=iostat)
      if (iostat .NE. 0) then
         print *, "Trouble opening ", file//'b', ".  Error code",
     A            iostat
         stop
      endif
      open (unit=3, file=file//'flowab', iostat=iostat)
      if (iostat .NE. 0) then
         print *, "Trouble opening ", file//'flowab',
     A            ".  Error code", iostat
         stop
      endif

 10   call rk45(eqnsys, n, y, t, tout, RELERR, ABSERR, IFLAG, WORK,
     A          IWORK)
      tout = tout + dt
      print 15, t, y(1), y(2), pressure(y(1)), IFLAG
      write (1,*) t, y(1)
      write (2,*) t, y(2)

      theta = y(2)/Binit
```

```
      p = y(1)*M
      alpha = exp(log(c_h)*theta)

      f1  = 60.*area*(3.5e19*p)/sqrt(mass*temp)*alpha
      f2  = 60.*y(2)/ts
      if (abs(f1 - f2)/f1 .lt. 0.0001) f1 = f2
      flowAB = f1 - f2
      write (3,*) t, flowAB

 15   format (1X, F5.2, 3(E11.5, 3X), I4)
      if (IFLAG .GT. 2) goto 100
      if (t .GT. timeout) then
         print *, "t >", timeout
         goto 100
      endif
      goto 10
 100  close(1)
      return
      end


***---------------------------------------------------------
*** Compute the equations.
*** A is y(1), B is y(2).
*** This set of equations is for the basic model.
***---------------------------------------------------------
      subroutine eqnsys(t, y, ydot)
      real t, y(*), ydot(*)

      real proom, area, Ed, M, mass, MQ, Na, oilspeed
      real flowAB1, flowAB2, alpha1, alpha2, theta1, theta2
      real p, pumptp, R, temp, theta, ts, volume, Ainit, Binit

      common Ainit, Binit, Ed, proom, M, MQ, area, mass, Na, R, temp,
     A       volume, ts, area_l, area_h, bmax_l, bmax_h, c_l, c_h

      theta = y(2)/Binit
      p = y(1)*M
      alpha = exp(log(c_h)*theta)

      f1  = 60.*area*(3.5e19*p)/sqrt(mass*temp)*alpha
      f2  = 60.*y(2)/ts
      if (abs(f1 - f2)/f1 .lt. 0.0001) f1 = f2
      flowAB = f1 - f2
```

```
      if (p .GT. 40.) then
         oilspeed = 250.
      else
         oilspeed = p*(250./39.) - 250./39.
      end if
      pumptp = oilspeed*p*MQ

      ydot(1) = -flowAB - pumptp
      ydot(2) = flowAB
      return
      end


***----------------------------------------------------------
*** Compute the equations.
*** A is y(1), B is y(2).
*** This equation set is for the leak model.
***----------------------------------------------------------
      subroutine eqnsys(t, y, ydot)
      real t, y(*), ydot(*)

      real proom, area, Ed, M, mass, MQ, Na, oilspeed
      real flowAB1, flowAB2, alpha1, alpha2, theta1, theta2
      real p, pumptp, R, temp, theta, ts, volume, Ainit, Binit
      real dleak, ca

      common Ainit, Binit, Ed, proom, M, MQ, area, mass, Na, R, temp,
     A       volume, ts, area_l, area_h, bmax_l, bmax_h, c_l, c_h

      theta = y(2)/Binit
      p = y(1)*M
      alpha = exp(log(c_h)*theta)

      f1      = 60.*area*(3.5e19*p)/sqrt(mass*temp)*alpha
      f2      = 60.*y(2)/ts
      if (abs(f1-f2)/f1 .lt. 0.0001) f1 = f2
      flowAB = f1 - f2
      if (p .GT. 40.) then
         oilspeed = 250.
      else
         oilspeed = p*(250./39.) - 250./39.
      end if
      pumptp = oilspeed*p*MQ
      ca = 5*3.1415926*1.e-4
```

```
      dleak = 60.*ca*(proom-p)*MQ

      ydot(1) = -flowAB - pumptp + dleak
      ydot(2) = flowAB
      return
      end


***---------------------------------------------------------
*** Compute the equations.
*** A is y(1), B is y(2).
*** This equation set is for the virtual leak model.
***---------------------------------------------------------
      subroutine eqnsys(t, y, ydot)
      real t, y(*), ydot(*)

      real proom, area, Ed, M, mass, MQ, Na, oilspeed
      real flowAB1, flowAB2, alpha1, alpha2, theta1, theta2
      real p, pumptp, R, temp, theta, ts, volume, Ainit, Binit
      real dleak, ca, leakvol

      common Ainit, Binit, Ed, proom, M, MQ, area, mass, Na, R, temp,
     A       volume, ts, area_l, area_h, bmax_l, bmax_h, c_l, c_h

      theta = y(2)/Binit
      p = y(1)*M
      leakvol = .001*10.
      pleak = y(3)/(MQ*leakvol)
      alpha = exp(log(c_h)*theta)

      f1      = 60.*area*(3.5e19*p)/sqrt(mass*temp)*alpha
      f2      = 60.*y(2)/ts
      if (abs(f1-f2)/f1 .lt. 0.0001) f1 = f2
      flowAB = f1 - f2
      if (p .GT. 40.) then
         oilspeed = 250.
      else
         oilspeed = p*(250./39.) - 250./39.
      end if
      pumptp = oilspeed*p*MQ
      ca = 1.5*3.1415926*1.e-6
      dleak = 60.*ca*(pleak-p)*MQ
```

```
ydot(1) = -flowAB - pumptp + dleak
ydot(2) = flowAB
ydot(3) = -dleak
return
end
```

BIBLIOGRAPHY

[Berleant and Kuipers, 1990] Daniel Berleant and Benjamin Kuipers. Qualitative and numerical simulation : Bridging the gap. Submitted to AI Journal special volume on qualitative reasoning about continuous systems., 1990.

[Berleant, 1989] Daniel Berleant. A unification of numerical and qualitative model simulation. In *Proceedings of the Model-based Reasoning Workshop*, 1989.

[Cheung and Stephanopoulos, 1989] Jarvis Tat-Yin Cheung and G. Stephanopoulos. Representation of process trends – part i. a formal representation framework. Technical Report LISPE-89-052, Massachusetts Institute of Technology, 1989.

[Dalle Molle, 1989] David Dalle Molle. Qualitative simulation of dynamic chemical processes. Technical Report AI89–107, Artificial Intelligence Laboratory, University of Texas at Austin, Austin, Texas 78712, May 1989.

[de Kleer, 1977] Johan de Kleer. Multiple representations of knowledge in a mechanics problem solver. In *Proceedings of IJCAI-77*, pages 299–304, 1977.

[DeCoste, 1990] Dennis DeCoste. Dynamic across-time measurement interpretation. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pages 373–379, 1990.

[Doyle *et al.*, 1989] Richard J. Doyle, Suzanne M. Sellers, and David J. Atkinson. A focused, context-sensitive approach to monitoring. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1231–1237, 1989.

[Dushman and Lafferty, 1962] Saul Dushman and J. M. Lafferty, editors. *Scientific Foundations of Vacuum Technique*. John Wiley and Sons, New York, second edition, 1962.

[Duval, 1988] Pierre Duval. *High Vacuum Production in the Microelectronics Industry*. Elsevier Science Publishers, Amsterdam, 1988.

[Dvorak and Kuipers, 1989] Daniel Dvorak and Benjamin Kuipers. Model-based monitoring of dynamic systems. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1238–1243, 1989.

[Forbus and Falkenhainer, 1990] Kenneth D. Forbus and Brian Falkenhainer. Self-explanatory simulations: An integration of qualitative and quantitative knowledge. Submitted to AAAI-90, 1990.

[Forbus, 1986] Kenneth D. Forbus. Interpreting measurements of physical systems. In *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, pages 113–117, 1986.

[Forsythe *et al.*, 1977] G. E. Forsythe, M. Malcolm, and C. Moler. *Computer Methods for Mathematical Computations*, pages 129–147. Prentice-Hall, Englewood Cliffs, 1977.

[Franklin, 1986] Gene F. Franklin. *Feedback Control of Dynamic Systems*. Addison-Wesley, Reading, MA, 1986.

[Fusillo and Powers, 1988] R. H. Fusillo and G. J. Powers. Operating procedure synthesis using local models and distributed goals. *Computer and Chemical Engineering*, 12(9/10):1023–1034, 1988.

[Jacquez, 1985] John A. Jacquez. *Compartmental Analysis in Biology and Medicine*. University of Michigan Press, Ann Arbor, MI, 1985.

[Kahaner *et al.*, 1989] David Kahaner, Cleve Moler, and Stephen Nash. *Numerical Methods and Software*. Prentice-Hall, Englewood Cliffs, 1989.

[Karp and Friedland, 1989] Peter D. Karp and Peter Friedland. Coordinating the use of qualitative and quantitative knowledge in declarative device modeling. In Lawrance E. Widman, Kenneth A. Loparo, and Norman R. Nielson, editors, *Artificial Intelligence, Simulation and Modeling*, chapter 7. John Wiley and Sons, 1989.

[Kuipers and Berleant, 1988] Benjamin Kuipers and Daniel Berleant. Using incomplete quantitative knowledge in qualitative reasoning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 324–329, 1988.

[Kuipers and Chiu, 1987] Benjamin Kuipers and Charles Chiu. Taming intractible branching in qualitative simulation. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, page ??, 1987.

[Kuipers, 1986] Benjamin Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:289–338, September 1986.

[Lakshmanan and Stephanopoulos, 1988] R. Lakshmanan and G. Stephanopoulos. Synthesis of operating procedures for complete chemical plants – i. hierarchical, structured modelling for non-linear planning. *Computer and Chemical Engineering*, 12(9/10):985–1002, 1988.

[LaPelle, 1972] Roland Rutledge LaPelle. *Practical Vacuum Systems.* McGraw-Hill, New York, 1972.

[LeClair and Abrams, 1988] Steven R. LeClair and Frances L. Abrams. Qualitative process automation. In *Proceedings of the 27th National Conference on Decision and Control*, pages 558–563, 1988.

[Lewen, 1965] Gerhard Lewen. *Fundamentals of Vacuum Science and Technology.* McGraw-Hill, New York, 1965.

[Richmond *et al.*, 1987] Barry Richmond, Steve Peterson, and Peter Vescuso. *An Academic User's Guide to STELLA.* High Performance Systems, 13 Dartmouth College Highway, Lyme, NH, 03768, 1987.

[Roth, 1976] A. Roth. *Vacuum Technology.* North-Holland, Amsterdam, 1976.

[Sacks, 1987] Elisha Sacks. Hierarchical reasoning about inequalities. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 649–654, 1987.

[Simmons, 1983] Reid Simmons. The use of qualitative and quantitative simulation. In *Proceedings of AAAI-83*, pages 364–368, 1983.

[Skorstad and Forbus, 1989] Gordon Skorstad and Ken Forbus. Qualitative and quantitative reasoning about thermodynamics. In *Proceedings of the Third Internationsal Workshop on Qualitative Physics (QPW-89)*, 1989.

[Van Atta, 1965] Chester Murray Van Atta. *Vacuum Science and Engineering*. McGraw-Hill, New York, 1965.

[Williams, 1988] Brian C. Williams. Minima : A symbolic approach to qualitative algebraic reasoning. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 264–269, 1988.

[Yip, 1988] Kenneth M. Yip. Generating global behaviors using deep knowledge of local dynamics. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pages 649–654, 1988.

VITA

Herbert "Bert" Kay, the son of Jack and Olga Veronica Kay, was born on May 1, 1959, in Boston, Massachusetts. He graduated from Belmont High School in June, 1977. In September, 1977, he entered the California Institute of Technology. He received a Bachelor of Science in Engineering and Applied Science in June, 1981. After summering in Europe, he began work as a Development Engineer for the Hewlett-Packard Company in their logic test group in Santa Clara, California. In October, 1986, he took a position as a Software Engineer at Mettler Instrumente AG, in Greifensee, Switzerland. In September, 1988, he entered The Graduate School of The University of Texas. On August 4, 1989, he married Margaret Clare Fletcher.

Permanent address: 302 East 41st St
Austin, Texas 78751

This thesis was typeset[1] with LaTeX by the author.

[1] LaTeX document preparation system was developed by Leslie Lamport as a special version of Donald Knuth's TeX program for computer typesetting. TeX is a trademark of the American Mathematical Society. The LaTeX macro package for The University of Texas at Austin thesis format was written by Khe-Sing The.