A CONJOINT PATTERN RECOGNITION APPROACH TO NONINTRUSIVE LOAD MONITORING

by

Steven Bruce Leeb

S.B., Massachusetts Institute of Technology, 1987

S.M., Massachusetts Institute of Technology, 1989

E.E., Massachusetts Institute of Technology, 1990

Submitted to the Department of
Electrical Engineering and Computer Science
In Partial Fulfillment of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

at the

Massachusetts Institute of Technology

February 1993

Signature of Author _____
Department of Electrical Engineering and Computer Science
January 20, 1993

Certified by _____
Professor James L. Kirtley, Jr.
Thesis Supervisor

Accepted by _____
Professor Campbell L. Searle
Chairman, Departmental Committee on Graduate Students

A CONJOINT PATTERN RECOGNITION APPROACH TO NONINTRUSIVE LOAD MONITORING

by

STEVEN B. LEEB

Submitted to
the Department of Electrical Engineering and Computer Science
on January 20, 1993 in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy

ABSTRACT

The nonintrusive load monitor (NILM) is intended to determine the operating schedule of the major electrical loads in a building from measurements made at the utility service entry. Conventional load monitoring schemes require separate metering equipment or connections for every load of interest. The NILM would considerably simplify the installation effort required to collect load usage data. When the costs of aggregating the data from a number of conventional load monitors are weighed against the ease with which a single NILM can collect and preprocess data, the NILM may also turn out to provide substantial savings in cost and time.

The feasibility of the nonintrusive monitoring approach for small homes has been demonstrated by the residential nonintrusive load monitoring project undertaken in MIT's Laboratory for Electromagnetic and Electronic Systems. Fundamental limitations in the design of the residential NILM hinder its ability to operate in a commercial or industrial building, where appliances turn on and off frequently and substantial efforts, e.g., power factor correction, are made to homogenize the steady state behavior of different loads. This thesis conjoins vector space and syntactic pattern recognition systems to increase the applicability of the NILM. In particular, this thesis focuses on the development of multiscale event detectors to augment the minimum information estimation techniques used in the residential NILM. The result is a new syntactic grammar that extends minimum information techniques into many possible applications that can be viewed in a multiscale framework.

A prototype of the multiscale event detector is presented. It is implemented on an advanced digital signal processor that serves as a prototype for a commercial NILM. Results of experimentation with the prototype are reported.

Thesis Supervisor: Professor James L. Kirtley, Jr.

Acknowledgements

This research was funded by the Electric Power Research Institute under EPRI project number RP2568-15. Additional funding for this work was provided by AMP Incorporated. I am grateful to the EPRI project sponsor, Laurence Carmichael, and to Joseph Sweeney, Michael LeVan, and David Hitz of AMP Incorporated for their generosity, insights, and technical assistance.

I am indebted to my advisor and mentor Jim Kirtley. Through three theses, he has patiently trained and supported me, and provided his unerring sense of how and where to find the right answers. He has taught me by example what it means to be a great engineer.

I am also indebted to my thesis readers, George Verghese and Jacob White. Both are wellsprings of knowledge and creativity. They have helped me through many tough spots during the course of this research. The support of the LEES faculty and staff has been unwavering.

I thank my peers, especially my officemates Brett, Derrick, and Ray, for their friendship, assistance, and sense of humor.

Finally, I wish to express my sincere gratitude to my family for their patience, support, and understanding. I regret that my father, who inspired me to achieve, did not have the chance to read these words.

To my wife, Rebecca

Contents

List of Figures

Chapter 1

Monitoring Nonintrusively

Electric utilities want to develop detailed usage profiles of their customers to aid in leveling peak loads and in planning future capacity. Conventional metering of individual appliances is costly and inconvenient to the consumer. These costs increase swiftly as data requirements become increasingly complex. The situation is summarized succinctly in [1]:

> ... the high cost of equipment continues to limit the amount of [usage] data utilities can collect. Additional drawbacks of the equipment now available for collection of end-use load survey data range from their cost, reliability, and flexibility to intrusion into the customer's activities and premises.

To deal with these concerns, utilities have sought a way of determining the operating history of an electrical load from measurements made solely at the utility service entry of a building. This problem is actually quite tractable under certain conditions. The residential nonintrusive load monitoring project undertaken in MIT's Laboratory for Electromagnetic and Electronic Systems has demonstrated in [137] the feasibility of a low cost, microprocessor–based recorder that competes favorably with conventional load monitoring schemes. Recently, other researchers have rediscovered some of the driving principles that would support the development of such a load monitor; see for example [136]. The advantages of nonintrusive load monitoring over conventional load monitoring include:

- ease of installation at the monitoring site, since the nonintrusive load monitor (NILM) requires a single set of electrical ties
- simplification of data collection, since the NILM automatically determines the electrical nature of the simple, "two–state" appliances in a target building without the need for a load survey or inspection

- facilitation of data analysis since, by definition, the NILM collects and potentially analyzes all data at a central location.

The NILM is more than a convenient and economical means of acquiring energy usage data. It is, for example, a potentially important platform for power quality monitoring. Utilities, manufacturers, and consumers are becoming increasingly aware of "power pollution." Many loads, such as computers and other office equipment, lighting fixtures (which account for 20 to 25 percent of electric power sales ([149])), and adjustable speed drive systems found in modern air handling equipment, may contain solid state switch mode power converters that could draw distorted, nonsinusoidal input current waveforms ([154],[110]). These harmonic currents create harmonic voltages as they flow through impedances in the utility's transmission system, degrading the quality of the delivered voltage waveform. Power pollution has become more than an academic concern as consumers increasingly turn to power electronic solutions to increase efficiency.

Additional monitoring capability for harmonic current content would generally be cheaper and easier to install at a service entry location than on every load of interest. The NILM could track down power quality offenders by correlating the introduction of undesired harmonics with the operation of certain loads at a target site. This and other potential uses of a centralized, near real-time load monitor with a reasonable amount of computational power will be discussed and demonstrated in this document.

The next sections review the current state-of-the-art in nonintrusive monitoring as implemented in the "residential" NILM. The core of the residential NILM is a syntactic error correction and pattern recognition algorithm that works with remarkably simple input data, i.e., changes in the "steady state" real and reactive power levels observed at the utility service entry. This approach has certain fundamental restrictions that limit performance in more demanding commercial and industrial settings. The goal of this doctoral thesis is to develop novel numerical pattern recognition techniques to augment the existing syntactic methods in a systematic way. These methods can be used to create a new monitor more suitable for broader use than the current residential NILM. The techniques developed in this thesis are generally applicable to a variety of other signature recognition problems.

1.1 Background

Pattern recognition systems determine the class membership of an input data set; pattern recognition methods frequently are characterized as either *vector space* methods or *syntactic* methods ([46]).

Vector space methods, also called decision discriminant ([141]), determinant ([107]), or decision theoretic ([43]) methods, operate on a collection of numbers, possibly in vector or matrix form, which categorize the input data. A preprocessor or *feature extraction* ([107]) technique may be applied to the raw vector or matrix of input data to reduce dimensionality by eliminating redundancy, easing the computational burden of the pattern recognizer. Class membership determination may then be made either by a hard numerical discriminant, e.g., a threshold determination based on the norm of the difference between the input data vector and a template or prototype vector, or by a semi–adaptive method that might, for instance, make use of variable probability density functions to determine a classification based on certain input features. Vector space methods have the advantage of a vast array of powerful mathematical tools for manipulating numerical data. Such tools may be computationally intensive, however, restricting their applicability in real-time.

Syntactic or symbolic recognition methods attempt to classify input data by identifying ("parsing") patterns of symbols (a "grammar"). Vector methods typically attempt to compare numerical estimates or evaluations of both the input data and a known or hypothesized model structure or discriminator. Syntactic recognition methods are distinct in that they try directly to identify underlying structure in the input data.

Consider, for example, the classic example of chromosome identification discussed in [77], [46], and [141]. A biological chromosome may be thought of as a closed, simply connected region. At varying stages of development, a chromosome will assume different shapes that can be modeled as an assemblage of many curving line segments. In [46] five fundamental "building block" segments, labeled a – e, were sufficient to describe each of the different possible chromosome border segments under consideration. Many more than five total segments might be needed to completely circumscribe a particular chromosome; however, each of the necessary segments was one of the five building blocks. Any chromosome, then, could be described as a finite, syntactic sequence of segments, e.g., *abbacddde*....

Recognizing a chromosome as a member of a particular shape class involved identifying an observed syntactic sequence.

This type of structural pattern recognition is subtly distinct from vector methods. Pressing the chromosome example a bit further, we might expect, in a numerical pattern recognition setting, that a chromosome would be reduced by a preprocessor to a set of metrics such as moment (i.e., center of gravity), area, and luminescence under certain photochemical conditions. A numerical discriminator might be used to associate these observed metrics with known templates to classify the input data and, therefore, the chromosome.

Numeric and syntactic pattern recognition problems, and estimation problems in general, are made more difficult in many cases by uncertainty in an assumed underlying model. The minimization of error criteria is a common feature of both numeric and syntactic schemes, although the error criteria in syntactic schemes may be derived from indirect measures of information content in proposed classifications of input data rather than by direct numerical manipulation of the data. Class determination based on the minimization of error criteria is only likely to be useful if the classes are related in a meaningful way to the observations.

This uncertainty in modeling leads to a tendency to classify the observations poorly in terms of exposing common and differing features. The more classes or parameters in the assumed model, the better the data will fit. For example, in a clustering application where n vectors are to be assigned to m classes, perfect classification of the data is possible if $m = n$. This solution does little to expose common features in the observations. At the opposite extreme, all of the observations could be assigned to a single class, which does little to expose differences in the observations.

Many *ad hoc* approaches for dealing with modeling uncertainty have appeared in a vast range of pattern recognition and estimation settings. In the application of unsupervised data clustering alluded to above, for example, the *Akaike Information Criterion* (AIC) has been used to strike a balance between overfitting and underfitting input data ([20], [8]). The AIC computes the global error residual associated with assigning input data to an existing set of clusters or classes of data and the error residual for the same set of clusters plus one additional cluster or class. A preset threshold determines whether or not a reduction in global error gained by adding a cluster adequately motivates the addition.

The minimum information (MI) estimation method introduced in [56] is a remarkably versatile method for accounting for model uncertainty that encompasses many of the *ad hoc* methods found in practice. At the risk of oversimplifying, the MI method is a systematic way to trade off *goodness of fit* to the input data versus class or *model complexity.* A central feature of this approach is to develop an information measure for a particular problem that contains both a penalty for poorly predicting the observed events with a proposed model and also a penalty for model complexity, e.g., too many clusters. A steepest descent optimization algorithm is used to search the potential model space for a model that yields a minimum of this information measure. The method is applicable to both numeric and syntactic recognition problems, provided that the problem can be formulated in a way that exposes an information measure to be minimized. Several examples are presented in [56], including applications to clustering, waveform segmentation, and noise removal in binary images. A syntactic minimum information approach is the underpinning of the residential NILM.

Consider a simple MI estimation example adapted from [56], the identification of the structure of a finite state machine (FSM) from a sequence of arc transitions given as observed data. A sequence of observed transitions might be as follows:

... 1 2 1 2 1 2 1 2 1 2 1 2 1 2 ...

In general, there are an infinite number of finite state machines that could create this sequence of events. Four candidate machines, all consistent with the observed events, are shown in Fig. 1.1. The simplest candidate FSM labeled (A) underfits the data. It states that any event may occur at any time, with no recognition of the alternating pattern inherent in the observations. Candidates (C) and (D) overfit the data in the sense that the machines contain unnecessary states to generate a perfect fit to the observed data. If complexity is quantified as the number of arcs and states in the FSM, candidate (B) provides an optimum trade-off between quality of fit to the data and structural complexity of the assumed model. Note that the observations need not be numerical data. A minimum information estimator simply requires that the quality of fit and complexity of the model (FSM) be quantifiable.

The next section exposes more of the features of the MI approach by directly discussing a related example application, the residential NILM.

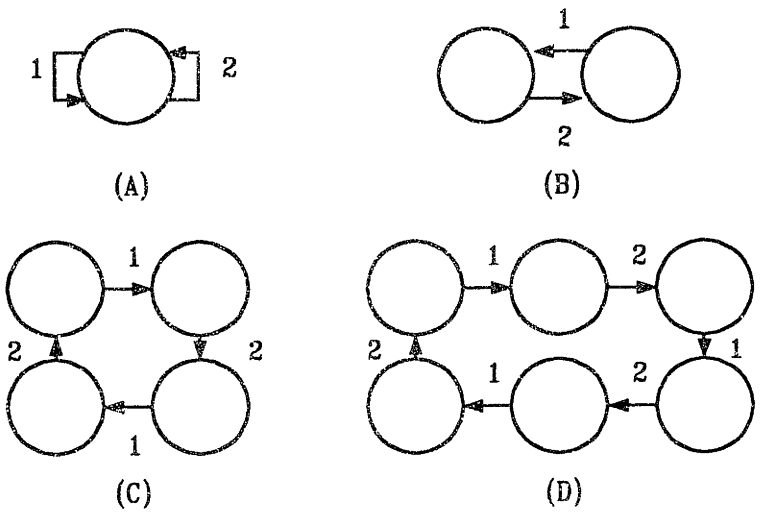


Figure 1.1: Candidate Finite State Machines

1.2 Syntactic Analysis and Nonintrusive Monitoring

In the parlance of nonintrusive monitoring, an electrical load is modeled as an FSM. The states of the FSM indicate the operating state of the load. The arcs or transitions of the FSM indicate *events* or observations made by the NILM at the service entry as the load changes states. In the implementation described in [137], the events are vectors of data that indicate changes in "steady state" levels of real and reactive power. In this section, the events are simplified to changes in apparent power.

Consider, for example, the waveform in Fig. 1.2 that shows the envelope of apparent power for a large resistive heating load. (All of the operating traces in this section are data collected from real, operating loads with a digital storage oscilloscope and an analog power monitoring circuit similar to the one described in Chapter 5.) This load would be modeled by the simple *two-state* FSM shown in Fig. 1.3. The labels 1 and 2 in Fig. 1.2 indicate turn–on and turn–off transitions in apparent power, respectively. These transitions are modeled by arcs in the FSM in Fig. 1.3. The details of the transients in Fig. 1.2 are ignored in the FSM. No attempt is made, for example, to account for the positive temperature-resistance coefficient characteristic of the heating element evidenced in Fig. 1.2. The arcs in Fig. 1.3 simply indicate the changes in "steady" power level. Clearly, the term "steady state" power

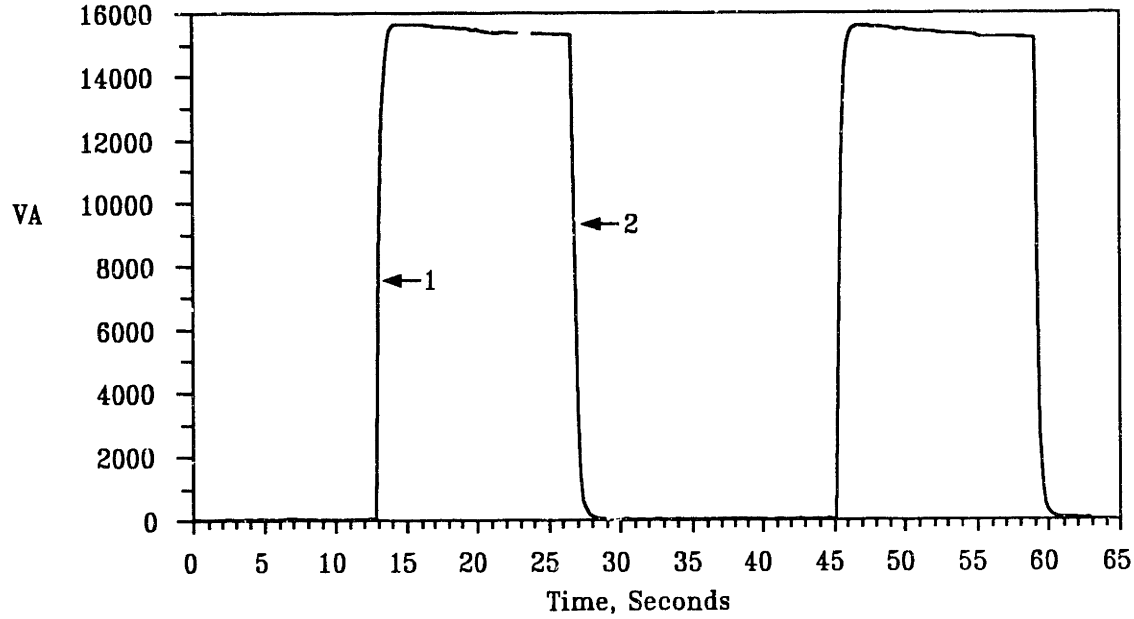


Figure 1.2: Apparent Power Consumed by a Resistive Heater

level needs to be defined carefully, but for the moment, it simply refers to an intuitive concept of reasonably long plateaus in power level. The load in Fig. 1.2 is a large industrial load, but it is similar except in scale to many residential loads, e.g. an incandescent lamp.

Other *multi-state* loads will draw a variety of power levels as they cycle through their operating modes, and will be described by FSM's with more than two states. A simple and familiar example of a multi-state load is a three-way lamp. Such a lamp may be off, or set to 50, 100, or 150 watts sequentially by a ratcheted switch. This lamp would be described by a four-state FSM with states for steady state power levels of 0, 50, 100, and 150 watts. The FSM would have a total of three 50 watt arcs and one negative 150 watt arc. Other residential examples include dishwashers, clothes dryers, and frost–free refrigerators ([137]). *Variable* loads that never settle to a steady state power level are essentially undetectable by the residential NILM algorithm.

At the service entry of a building, the NILM will observe a time pattern of changes in steady power level as the various loads in the building change operating state. These events correspond to arc transitions in the FSM models of the various loads in the building. For simplicity, ignore the details of the split phase 120V/240V utility service provided to residences (or the three phase service commonly provided to large commercial or industrial

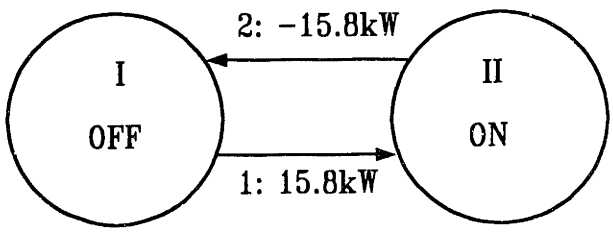


Figure 1.3: FSM Model of a Resistive Heater

consumers) and imagine changes in apparent power on a two wire service. The waveform in Fig. 1.4 shows the envelope of apparent power observed on a service with three large resistive heating loads operating in parallel. One of the loads is identical to the resistive heater shown in Fig. 1.2, and cycles on and off with a period of about 32 seconds. A second load cycles on and off with a period of about 90 seconds. A third load turns on and off only once during the interval shown in Fig. 1.4. In general the waveform observed at the service entry will be more complicated, especially with the introduction of multi-state loads.

The stated purpose of the residential NILM is to recover the operating history of the significant electrical loads in a target building. The process for performing this identification can be broken down into three steps, all of which could be formulated, in principle, as minimum information estimation problems:

- *Event Detection:* Changes in the steady state power levels observed at the service entry must be quantized and recorded as events. The process of identifying the events while ignoring transient behavior is essentially a problem of waveform segmentation. The problem is similar to "change in mean" fault or transient detection ([11]). A detailed review of the formulation of this problem in a minimum information estimation framework is left to [56]. The residential monitor described in [137] and [57] does not use an MI waveform segmentation scheme to identify events. Instead, a simple, effective, *ad hoc* scheme is used. The input power waveform is sampled at 1 Hertz. When several successive values over a fixed time period are equivalent within a cer-

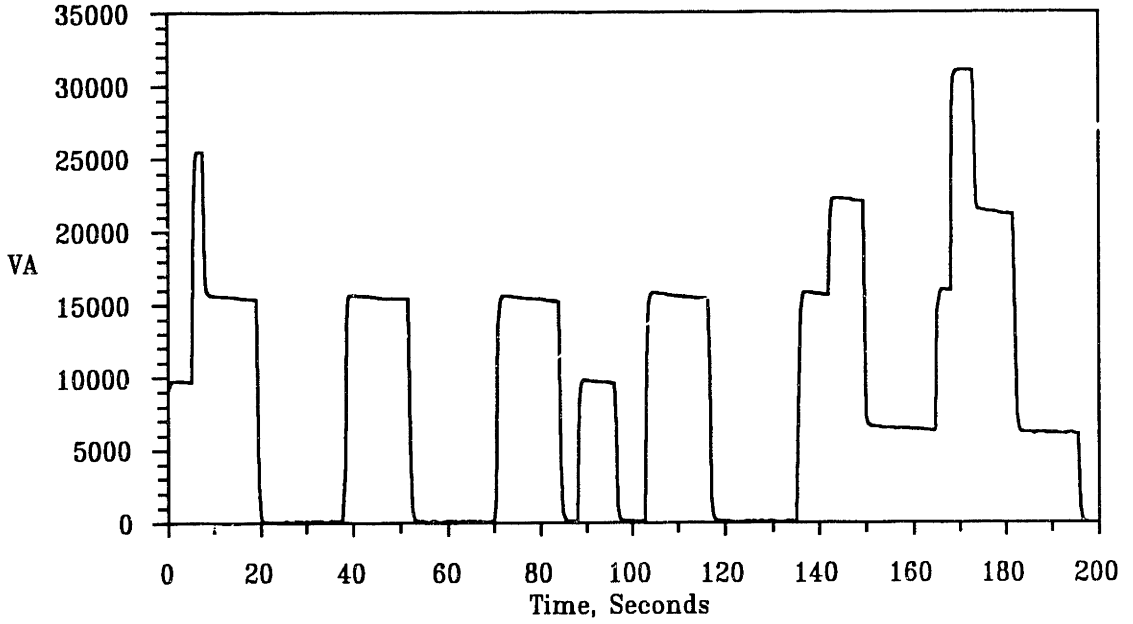


Figure 1.4: Apparent Power for Three Resistive, Two-State Loads in Parallel

tain tolerance, the waveform is presumed to be in a plateau or steady state period. Intervals between steady state periods are regions of change or transients. An *event* is the difference in power level between two steady state periods, one before and one after a transient.

- *Event Clustering:* The utility voltage may fluctuate over the course of observing a building's service entry. Unfortunately, most electrical loads commonly found in residences are not perfectly modeled as linear circuits. For example, the positive temperature-resistance coefficient response of the heating element shown in Fig. 1.2 is a common nonlinearity in many resistive heating and lighting loads. These nonlinearities make it difficult to normalize the observed events over voltage fluctuations ([57]). Events generated by a particular load will not tend to appear exactly the same at the service entry over time. A clustering algorithm is therefore used to group together observations which correspond to the same appliance events but which "scatter" slightly in the observation space. As hinted in the previous section, the clustering problem is amenable to the MI approach. An information measure that balances the area of the clusters and the total number of clusters can be minimized to yield a small number of tightly packed clusters. The current implementation of the residential NILM makes use of a clustering algorithm that is *ad hoc* but similar in many respects to a formal MI approach ([57]).

- *Operating History Determination:* Given a sequence of normalized, clustered events, the NILM must determine the operating history of individual appliances in the building. There are three approaches to this problem, each of which can be formulated in an MI framework. Each of these approaches is based on one of the following assumptions:

- Perfect event stream data/Unknown load models
- Flawed event stream data/Known load models
- Flawed event stream data/Unknown load models.

In short, different assumptions are made about the type and quality of the information available to the NILM, and the governing assumption set determines the extent to which the NILM is actually "nonintrusive." Each of the three sets of assumptions will be discussed in further detail below.

The first approach to decoding the observed event stream into individual load operation schedules is the primary focus of the examples in [54] and [56]. The event stream data is assumed to be uncorrupted by noise or simultaneous events. However, no assumptions are made about the number or type of loads (finite state machines) responsible for generating the event stream. An information criterion is developed in [54] that measures the complexity of a proposed set of FSM load models that could account for the observed event stream. The algorithms and software presented in the references minimize this criterion to develop a most plausible set of FSM load models.

In practice, however, the observed event stream is marred by noise and simultaneous events, limiting the practical applicability of this first approach. The second approach, therefore, assumes that the event stream is flawed but that some *a priori* statements can be made about the load models. This approach is the one used in the current residential NILM. In a strict MI framework, this approach involves modeling short windows of the event data by a "data FSM" that is to be examined and repaired if necessary by minimizing an information criterion developed from the known FSM load models.

In fact, the current residential NILM uses an even simpler approach based on a few observations about typical residential load characteristics. The majority of loads in a residence are well modeled by two-state FSMs. Nearly all of these models are also subject to a zero loop-sum constraint, i.e., that the sum of the arcs around any closed path in the FSM model will be zero. Essentially, most loads simply turn on and then turn off. If all of the loads are presumed to be accurately represented by two-state FSM models, the operating history reconstruction process devolves into merely collecting *matched* pairs of clusters of events whose centroids or means sum approximately to zero. In other words, every load will generate a cluster of turn–on events and a cluster of turn–off events of approximately

equal magnitude but opposite sign. Minor event stream errors due to noise, clustering errors, or simultaneous events will result in consecutive turn–on or turn–off events in the chronologically ordered event stream for a particular load. These errors are easily repaired by inserting the missing events. Precise timing for missing events may be established by ex *post-facto* examination of *anomalous* events that did not fall into matched clusters, or by stochastic means if necessary ([137]).

The situation becomes slightly more complicated in the presence of multi-state loads. One approach that works reasonably well in the residential NILM is to simply ignore the multi-state nature of the loads and identify the components as individual two-state appliances. For example, each resistive heating element in a stove could be identified and tracked as an individual two-state load. The compressor motor and the defrost heating element in a "frost-free" refrigerator could be tracked individually as two-state loads. If it is important to track multi-state loads, a multi-state event correction algorithm is required. The next section reviews an MI-type algorithm described in [55] that repairs slightly flawed event data generated by a multi-state load.

In the third approach, a "deluxe" NILM would ideally be able to use a flawed event stream to determine a previously unknown set of FSM load models. An algorithm based on the Viterbi algorithm for this ambitious technique is presented in [14] and [15]. This algorithm models the load as an FSM and also models windowed sections of the event stream as a data FSM. An MI optimization is conducted to modify both finite state machines to minimize a relevant information criterion. This algorithm is highly experimental in terms of use in the NILM. Particularly in the commercial and industrial monitoring setting, for example, it may turn out that a certain load has an FSM model with two or more closed loops with the same cyclic ordering. In this case, there is no guarantee that the FSM can be uniquely characterized by a finite run of observations. This is a general pitfall in the identification of multi-state loads, but it is especially unclear how this problem would affect the estimator working with flawed data and an uncertain load model. Furthermore, it seems possible that an unfortunate but plausible set of corruptions or simultaneous events in the observed data stream could create a "limit cycling" in the estimator, in which the NILM would be unable to converge on a likely set of load models, and would instead perenially cycle between different sets of models as more observations become available.

The third approach is an area of active research, and will require more experimentation with real load data before it can be used confidently. Fortunately, a variety of observations concerning load models have been made which render the second approach quite feasible for practical use. In the residential NILM, multi-state loads are modeled as several two-state loads. For a more sophisticated monitor, a one-time intrusion when the NILM is installed at a target site for the purpose of collecting load models is considered acceptable to many consumers, especially if loads can be studied without interrupting operation ([1]). This is particularly true for commercial and industrial users who are likely to be very interested in the usage data and other information that the NILM might provide.

1.3 Multi-State Event Correction

The multi-state load error correction algorithm is designed to recover the operating history of a multi-state load modeled with a known FSM structure. The algorithm assumes that a large number of the events in an input stream are attributable to a known FSM. Presumably, the input data does not perfectly match the FSM structure. That is, the input data sequence contains anomalies, so the data does not describe a perfect route through the FSM. The correction algorithm attempts to make a reasonable estimate of the operating history of the appliance based on the given input data. Appendix A presents a *C* software program, developed for this thesis, that implements the algorithm discussed in this section. The primary reference for the algorithm discussed here is [55]; please refer to that document for more background information.

In practice, the correction algorithm would be employed after the input event stream had been clustered. Each arc on the FSM describing a load would presumably correspond to a cluster of events. The chronologically ordered events in all clusters that could be associated with an FSM would form the input event stream for this algorithm. Since two-state loads are a degenerate case for the multi-state estimator, the two-state correction algorithm used in the residential NILM could be replaced by the multi-state algorithm.

A number of important assumptions are made in the development of this error correction algorithm. As already mentioned, the FSM structure of the load in question must be known exactly. This structure may be determined either by occupant survey or a one-time

intrusion during which the load in question is operated specifically to develop a load model. Also, the input event stream must generally be compatible with the FSM load model. Relatively infrequent anomalies that can be analyzed as insertions, deletions, or changes of a small number of events must be surrounded by correct partial routes through the FSM model. Finally, the complexity of the allowed FSM models is restricted in the sense that the state of a particular model must be determined by a finite string of events. This assumption permits the verification of event strings in the observed data and the generation of "repair strings" for corrupted regions. Results from the residential NILM and preliminary surveys of commercial and industrial loads indicate that these assumptions are not overly restrictive. As a fall back, most multi-state appliances can be analyzed as collections of two-state appliances, simplifying the FSM modeling issues considerably. Also, the modified Viterbi approach described in [14] for the "flawed events/unknown models" category discussed in the previous section may eventually yield a more robust approach when employed with known models.

The error correction algorithm described here and implemented in Appendix A proceeds in four distinct steps to estimate the state history of a load:

- *Initialization:* The program reads in an FSM description and a raw, presumably flawed, event stream largely attributable to the load in question. The program analyzes the FSM to determine u, the minimum number of events needed to uniquely identify the current state of the FSM.
- *Boundary Determination:* The program reads through the input event data, searching for one of two possible types of anomalies. The first possible anomaly is a sequence of events that could not be generated by the FSM. The length of events scanned is a variable parameter of the program. The second possible anomaly is that a state transition is made that is not found in the set of arcs described by the FSM. In case of either error, the program marks a "segment boundary" at the point of the anomaly and also marks as untrustworthy a certain number of events after the anomaly. This number is also a parameter of the program.
- *Repair Generation:* For each string of untrustworthy events around a boundary, the program generates a list of possible paths through the FSM model that could connect the last trusted event to the first trusted event following the questionable string.
- *Error Minimization:* The string most likely to be the correct repair is determined by minimizing an MI-style criterion of the differences in information content between the erroneous observed string and the proposed repair string.

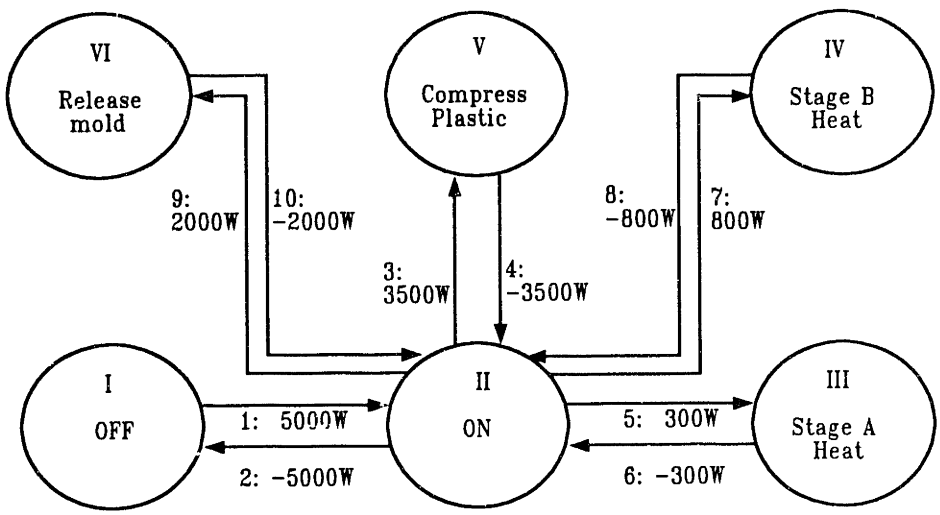


Figure 1.5: FSM Model for an Injection Molding Press

The third and fourth steps are repeated for each segment boundary. Once the algorithm evaluates every boundary, the corrected event stream will describe a complete path through the FSM load model.

To understand the algorithm better, consider the example of an industrial injection molding press. This example is a multi-state load containing a motor-driven hydraulic system, two resistive heaters, and a drying unit. The FSM shown in Fig. 1.5 describes the press examined here. The load begins completely off in state I. A 5000 watt change in apparent power occurs when the press turns on and enters state II. In this state, a nearly constant base load composed of the hydraulic system and the drying unit is on. From state II the press could repeat a sequence of operations as plastic parts are molded. First, the machine enters state V as the motor in the hydraulic system pressurizes a ram, forcing plastic into a mold. When the plastic is in the mold, the motor load falls as the hydraulic pressure on the ram relaxes, returning the machine to state II. The machine enters state III, returns to state II, enters state IV, and returns again to state II as the two heating units are cycled on and off individually to maintain a constant temperature in the mold. Finally, the machine enters and leaves state VI as the hydraulic system again pressurizes to move the ram back and release the molded plastic component. This entire cycle might repeat many

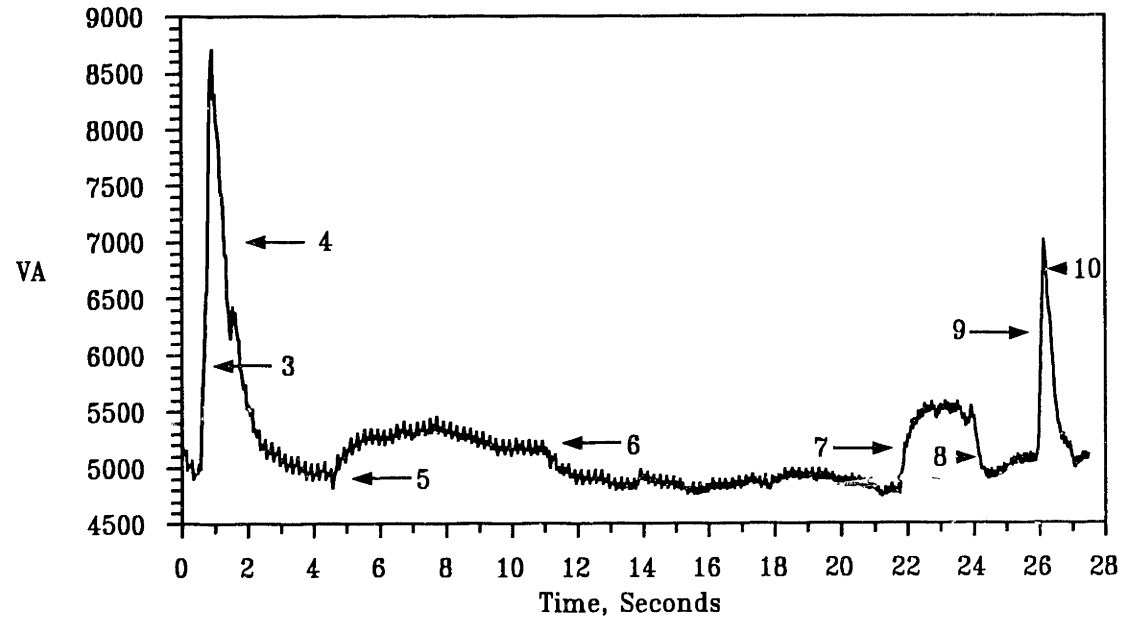


Figure 1.6: Apparent Power for an Injection Molding Press

times as parts are molded, until the machine finally is turned off and returns to state I. A recorded trace showing apparent power for an injection molding press is shown in Fig. 1.6. The transitions for the various events associated with molding a part are labeled in Fig. 1.6 to correspond to the arc labels in Fig. 1.5.

A section of an ideal event history for this load, with a corresponding state history, might be:

```
Event History:   1  3  4  5  6  7  8  9  10  3  4  5  ...
State History:   II V  II III II IV II VI II V  II III ...
```

The arc types 1, 3, 4, and so on presumably correspond to clusters in the clustered event data. The cluster corresponding to event type 1, for example, would include all events at or near 5000 watts. A raw, presumably inconsistent chronological listing of events in the clusters of observed data from the load monitor might appear as follows:

```
Observed Events: 1  3  4  5  6  7  8  9  10 | 10  3  4  5 ...
```

The two events 10 and 10 next to each other in the 9th and 10th positions in the input event stream are inconsistent with the FSM structure. A segment boundary (|) marks the anomaly. The multi-state error corrector cancels out a certain number of events in the

observed event data and attempts to fit a new, consistent and likely set of events to the observed data. In the implementation in Appendix A, the estimator cancels out $u+1$ events. For the FSM in Fig. 1.5, u, the minimum number of events needed to identify the current state of the FSM, is 1. So, the processed event history is:

```
Processed Events:  1   3   4   5   6   7   8   9   10 | -   -   4   5  ...
```

The "phrase" removed from the event history is 10 3. The next step is to generate hypothetical event phrases that could connect the two boundaries of the unknown (dashed) phrase. For example, four candidate event phrases for insertion are:

```
Candidate 1:  3
Candidate 2:  9  10  3
Candidate 3:  3  4   3
Candidate 4:  9  10  3  4  3
```

The error correction software in Appendix A will generate all possible event strings up to a certain length, preset by the user. Then, in the MI spirit, the best choice is selected by picking the candidate that is most like the removed phrase. Candidates 1 and 2 are similar in that each requires a single alteration in the removed phrase. Here, the somewhat arbitrary assumption is made that errors in the observed event stream are likely to be caused by insertions of events. So, removing a suspicious event is chosen to be a less costly act than inserting a new event. In practice, these cost functions, which determine the relative likelihood of insertions, deletions, and changes as corrections to the observed event data, would have to be estimated from empirical studies. Given this cost assumption, Candidate 1 will be chosen by the algorithm.

The corrected event stream can now be produced by inserting the selected candidate phrase back into the processed data. Also, since the corrected history is consistent with the FSM, the algorithm can now compute a correct state history, shown below:

```
Corrected Event History: 1   3   4   5   6   7   8   9   10   3   4   5    ...
Corrected State History:  II  V   II  III II  IV  II  VI   II   V   II  III ...
```

1.4 Contributions of This Work

Even augmented with the multi-state error correction algorithm, the approach used in the residential NILM contains several inherent limitations that restrict its applicability. These limitations are not fundamental to MI techniques in general. They are instead a function of the language or grammar of events to which the MI-type methods are applied. It is essential to realize that, while MI methods are remarkably versatile, their applicability depends in a fundamental way on the ability of the user to generate a useful formal language for the target application. No algorithmic or rote techniques are available in general for the creation of such languages.

In the current NILM, substantially different loads that coincidentally draw nearly identical steady state power levels are essentially indistinguishable. In the residential setting where load diversity is great, the simple monitoring scheme works well. Unlike residential consumers, however, commercial utility customers are often penalized not only for the real power that they use but also for the way they use the power, i.e., relative distribution of real and reactive power, quantity of harmonics generated, etc. Industrial and commercial buildings will, therefore, tend to contain loads that may be power factor corrected or otherwise modified to homogenize their steady state behavior. Schemes for load balancing, minimizing reactive power consumption, and limiting harmonic current generation in steady state are becoming increasingly economical and desirable ([154], [92], [80]). These schemes can severely limit the ability of the residential NILM to detect and distinguish different loads in a commercial or industrial environment.

To overcome this limitation, this thesis proposes to conjoin numerical and syntactic pattern recognition techniques to create a new, hybrid NILM. Vector space pattern recognition methods will be used to develop a transient event detector that can identify load transients or sections of transients. A match between a segment of the incoming data stream and a stored transient prototype will constitute an "event" for this new load monitor.

This thesis argues that the transient behavior of different electrical appliances is in large part determined by the physical design and purpose of the load. For example, a rotating machine accelerating a load would, during turn–on, draw power in quite a different manner than a gas discharge lamp just turning on. It is usually desirable and relatively

inexpensive to make the *steady state* electrical terminal waveforms of a motor and a bank of lights look similar to those of a resistive load. This tends to confound pattern recognition. It is a more expensive and less pressing matter entirely, however, to spend a great deal of effort masking the relatively brief *transient* behavior of most loads. Even loads with sophisticated power electronic front–ends, which might, for example, include very high quality power factor correction, tend to exhibit distinguishing behavior in transient operating modes. It should therefore be possible to distinguish different classes of devices performing different physical tasks by their transient electrical characteristics. Events composed of such characteristics should tend to be more robust differentiators of load type than events based on changes in steady state power levels, assuming that loads of interest do have some interesting transient dynamics (e.g., every load is not a resistor!).

The present NILM requires that the power drawn at the service entry must settle to a clean steady state power level after a transient in order that an event be recorded. This means that the residential NILM will monitor poorly or not at all in a building whose loads are constantly changing their electrical state. For proper event generation with a transient-based event detector, this thesis will show that only "key" features of a transient need be clearly identifiable or separable using the input waveforms. Provided these key features are identifiable, even constantly changing input waveforms with overlapping transients may be tractable. In other words, generating events from transients should permit the NILM to work in a much "busier" electrical environment in which the service entry rarely settles to a steady state. Important goals of this work will be to determine and expand the limits on a service entry's maximum allowable activity level under NILM operation.

There is an even more subtle limitation associated with the current approach to event detection in the residential NILM. One could argue that the performance of the residential system could be improved by increasing the sampling rate and narrowing the interval of "constancy" required before an event is generated. This would facilitate the detection of loads with pulse operating characteristics, like the hydraulic motor responsible for the events numbered 3, 4, 9, and 10 in Fig. 1.6. Commercial and especially industrial loads, however, may have transients that vary widely in time scale. This makes an intuitive or practically useful "steady state" interval much more difficult to define.

A major contribution of this thesis is the establishment of a multiresolution analysis

technique for pattern recognition applications. Multiresolution or multiscale analysis has become a familiar notion in signal processing and many other fields, but special considerations necessary for pattern recognition systems have often been ignored or insufficiently stressed. For example, this thesis argues that linear phase characteristics do not and should not have to be sacrificed in the development of filter banks for multiresolution search structures. Methods for controlling variations due to shift-variance are introduced. The Remez algorithm is extended in this thesis to permit the efficient design of optimal linear phase filters for tree-structured multiresolution search schemes. Nonlinear geometric filters that are compatible with the goals of multiresolution pattern recognition systems are reviewed. A number of important theoretical results, such as computationally efficient means for determining the condition numbers of decimated circulant matrices, are developed to permit the analysis and comparison of different approaches for multiresolution analysis.

The Radial Panel Installation Designer (RAPID) simulation software developed for this thesis provides a flexible and powerful means for designing and simulating electrical loads and building electrical services ([80]). Results from RAPID, in conjunction with data collected from actual loads, will be used to present and verify some of the heuristic arguments concerning load transients. Ultimately, many of the ideas developed are demonstrated in a real-time hardware implementation on an advanced digital signal processing (DSP) system.

As mentioned previously, the limitations of the current residential NILM are not inherent limitations of MI estimation. Hence, the techniques developed in this thesis constitute a major step toward the development of a general formal language for many multiresolution MI estimation problems. The versatility of the MI approach and the examples described in [56] will be enriched by the multiresolution tools developed here.

1.5 Thesis Outline

The next chapter presents a taxonomy of significant load classes likely to be found in commercial and industrial monitoring sites. Based on a review of the characteristics of these loads, including both theoretical considerations and examinations of RAPID simulations and collected waveforms, a logical foundation for the multiresolution search approach is developed.

Chapters 3 and 4 develop the mathematical tools used in the multiresolution search algorithm. Chapter 3 focuses on the construction of numerical pattern discriminators that are capable of identifying transients or components of transients at a particular time scale. Chapter 4 describes a tree-structured decomposition that employs these discriminators to search over multiple time scales. Filtering requirements are reviewed that permit the change of information content in an input signal as different scales are analyzed. Efficient algorithms for designing and employing such filters are presented.

Chapter 5 reviews the implementation of a tree-structured multiresolution event detector developed for this thesis on a TMS320C30 DSP system. A multiresolution event detection algorithm is described. The hardware and software described in this chapter expose many practical issues associated with the development of a real-time load monitor. Results from the implementation are presented and discussed. Finally, Chapter 6 concludes this thesis with a summary of results and a look at potential applications of the advanced load monitor and the multiresolution search algorithms.

Chapter 2

Load Survey

This chapter motivates the development of a multiscale load transient event detector for application in an advanced NILM. The turn–on transients for a variety of important commercial and industrial loads will be examined with the goal of exposing two key points. First, it will be seen that the transient behavior of many important load classes is sufficiently distinctive to serve as a reliable indicator of load type. Second, it will be seen that for many groups of loads, a single prototype transient shape can, with appropriate scaling in amplitude and duration, represent the transient behavior of all of the loads in a group.

While this load survey does not review every imaginable electrical load that could be present in commercial and industrial buildings, it does touch on the load categories that are responsible for the vast majority of electrical energy consumption in these settings. Therefore, the rationale for the development of the multiscale event detector is based on the groups of loads that are most likely to be of interest for nonintrusive monitoring and performance diagnostics. One conclusion suggested by the review that follows is that the transient behavior of most loads is intimately related to the physical task that the load performs. The transients associated with a fluorescent lamp and an induction motor, for example, are distinct because the physical tasks of igniting an illuminating arc and accelerating a rotor are fundamentally different. Load classes that perform physically different tasks are likely to be distinguishable based on their transient behavior. Hence, the analysis conducted here can be extended to many other load classes.

The waveforms presented in this chapter come from two sources. Some were collected directly from actual loads with a digital storage oscilloscope and current probe. The re-

mainder were produced by integrating differential equations describing the loads using the RAPID simulator described in Appendix B and [80]. The origins of each transient examined will be described in the text.

2.1 Simple Loads

The category of simple loads refers to those devices that are adequately modeled as time-invariant collections of passive, linear (LTI) circuit elements. The analytically simplest load of this type is one that can be represented as a resistor. The turn–on transient for such a device in, say, RMS AC current magnitude or in the magnitude of real power is a step. For comparison purposes, the only interesting feature of such a transient is its magnitude. Some heating loads, for example, might be adequately modeled as resistors.

Slightly more complicated load models might include reactive elements, i.e., inductors and capacitors. Loads modeled as impedances that include reactive components will generally exhibit more interesting turn–on transient behavior than a simple step. Variation in the transient behavior will be introduced by the homogeneous component of the solution of the state equations that describe the load. Unfortunately, with AC excitation, this variation will typically be highly time-variant and a particular transient shape will not reappear reliably. Efforts could be made to synchronize the pattern recognition process with the line excitation, minimizing the impact of the time variance. However, the only simple, consistently reliable aspects of these types of transients will be the changes in magnitude of features like real and reactive power from before to after the transient, when the load has entered approximate sinusoidal steady state operation.

If the loads at a target site are adequately modeled as LTI impedances, an event detector more sophisticated than the one employed in the residential NILM would not appear to be helpful. This is because individual loads could be identified by step changes in quantities like real and reactive power, provided that the changes for each load at the site were unique. Naturally, if two loads exhibited the same steady state changes, they could not be distinguished without additional information.

The load survey conducted for this thesis has revealed, however, that very few loads are well modeled as LTI impedances. Nonlinearities in the constitutive relationships of

the elements that comprise a load model, or in the state equations that describe the load, or both, tend to create interesting and repeatable transient profiles suitable for identifying specific load classes. Of course this behavior can always be ignored, and this is the approach adopted in the residential NILM. In the residential case, the loads are adequately modeled as LTI impedances sized appropriately to relate the changes in steady state levels before and after a load transient. In a residence, where practically no effort is made to correct power factor or homogenize the steady state power demand for each load, this approach has proven quite successful in differentiating the different loads at typical target sites.

But transient behavior does not have to be ignored. Useful transient profiles tend not to be eliminated even in loads which employ active waveshaping or power factor correction. Even relatively simple loads often exhibit helpful transient profiles.

2.2 Electrothermal Loads

Consider, for example, the turn–on transient of an incandescent lamp, shown in Fig. 2.1. The waveform shown in Fig. 2.1 is the instantaneous power transient of an actual lamp rated at 60 watts. The wire filament in the lamp exhibits a resistance that increases with temperature. Initially, when the lamp is off and the filament is cold, the resistance is relatively low. Immediately following turn–on, the resistance is still low and a large current surges through the filament. As ohmic losses increase the temperature of the filament, the lamp begins to glow, the filament resistance increases and the input current decreases. Within approximately three line cycles, the situation has stabilized. In a properly operating lamp, the filament resistance may be as high as ten times the resistance of the cold filament. The filament cools quickly when the lamp is turned off, and the transient profile shown in Fig. 2.1 is quite repeatable, even when the lamp is turned on and off with little time between repetitions.

This increase in resistance with temperature is common in most loads with heating elements. Loads whose transient profiles are shaped by the nonlinear, temperature dependent resistance of a heating element represent one class of a broad category of *electrothermal* loads. Electrothermal loads exhibit transient profiles that are functions of thermal energy transfer to or from some component of the load. In particular, loads with heating elements

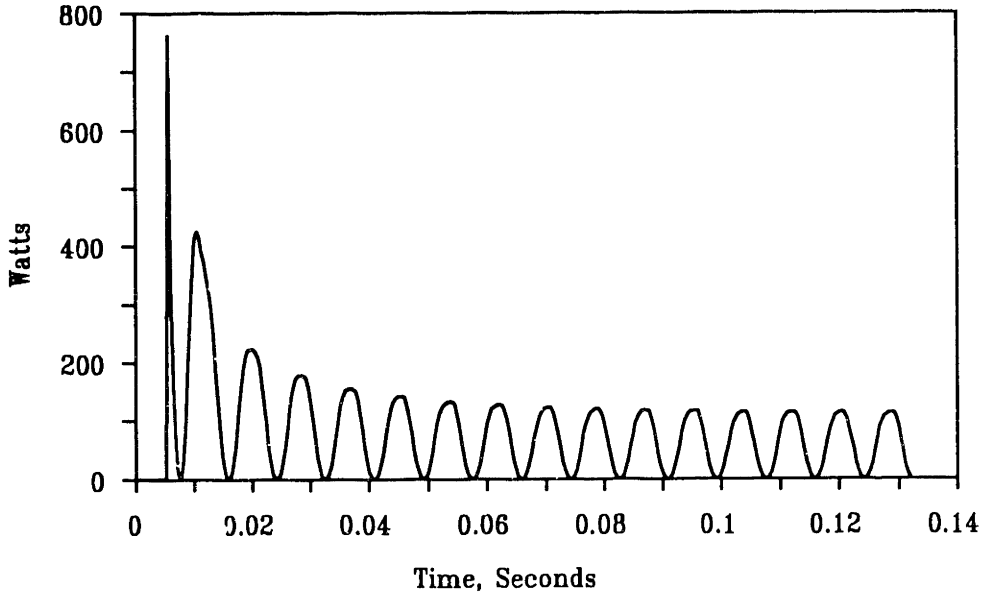


Figure 2.1: Incandescent Lamp Transient (Measured)

generally exhibit a transient profile that is similar in shape to the incandescent lamp transient, but possibly different in amplitude and scale. This shape is clearly apparent, for example, in the transients shown for the 15.8 kW heating element in Fig. 1.2, allowing for the fact that the plot in Fig. 1.2 is of apparent power magnitude instead of instantaneous power. Notice, however, that the absolute magnitudes of power are very different for the two loads. Also, while the duration of the incandescent lamp transient is approximately three line cycles, the duration of the heater transient in Fig. 1.2 is about ten seconds!

These similarities and differences are not surprising. The transient shapes are similar because essentially the same physical process is occurring during the turn–on transients of both the lamp and the heater. The two loads operate at very different power levels, however, creating differences in the amplitude and duration of the transients. The heating element operates in a liquid with a larger thermal mass than the gas surrounding the filament in the lamp. For the heating element, therefore, the ratio of peak in-rush current relative to its steady state current level is smaller than the same ratio for the lamp filament. This increases the time the heating element takes to reach a steady state operating point in comparison to the lamp. As the heat capacity of the thermal mass associated with a particular element varies, the duration of the transient will vary.

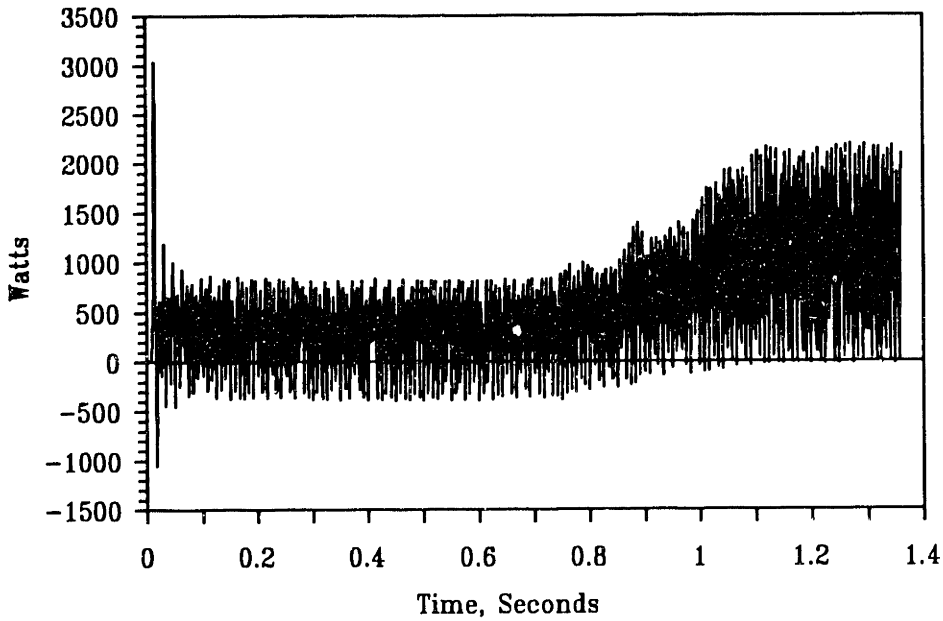


Figure 2.2: Fluorescent Lamp Bank Transient (Measured)

For a relatively small element, practically no transient at all might be observed. This is the case for some inexpensive coffee makers, for instance, where a relatively low power element is used to heat a large quantity of water ([137]). Particularly in industrial and commercial settings, however, the elements tend to be sized for a more powerful response, and the transient profile observed for the incandescent lamp is representative of a broad class of heating elements.

Another important class of electrothermal loads is typified by the fluorescent lamp. The turn–on transient in instantaneous power for an actual bank of 22 rapid start fluorescent lamps with power factor corrected iron core ballasts is shown in Fig. 2.2. [1]

The shape of the transient profile shown in Fig. 2.2 is strongly influenced by the electrical and thermal processes that lead to the ignition of an illuminating arc in each of the lamps in the bank. Immediately after turn–on, a large in-rush current is observed. This in-rush is caused by the in-rush current associated with the iron core ballasts, and also by the initial heating of the cathodes in each of the lamps to a steady preheat operating

[1]The profile in Fig. 2.2 is long and has therefore been sampled slightly to fit the enormous number of sample points into the available graph space. Some of the cycles have lost their shape and have a "squashed" or "spotty" look. The envelope of the profile is well preserved, however, and will serve here to expose key features of the load class.

temperature. For a single lamp fixture, the ballast in-rush current tends to be highly time-variant. For a bank of lights, differences in the nominal effective component values in each ballast and also differences in the remnant flux densities in the ballasts conspire to ensure that the bank as a whole exhibits a fairly consistent transient shape. This "consistency through numbers" seems to occur even in banks as small as four fixtures in the experiments conducted for this thesis.

A fairly long period of cathode heating – almost a full second – occurs after the initial in-rush current. During this preheat period the alkaline earth oxides that coat the cathode wires are heated to a point where they will emit electrons efficiently ([152]). At the end of the preheat period in each lamp, an arc fires across the tube and visible light is produced. When this occurs, the current drawn by the lamp fixture rises in magnitude to a steady state level. This process can be seen around the one second mark in Fig. 2.2.

There are several important observations that can be made about the profile in Fig. 2.2. Clearly, the shape of the transient is strongly influenced by the physical processes associated with starting the lamps. The transition to steady state around the one second mark is fairly wide, taking approximately two tenths of a second. This transition interval is considerably wider than the interval for a single lamp fixture for at least two reasons. Not all of the fixtures in the bank are exactly identical, and some random variation that results in a smearing or lengthening is to be expected. Also, in a large bank of lights wired in parallel, as is the case for the bank that created Fig. 2.2, wiring impedance causes the magnitude of the voltage applied to the individual fixtures to droop slightly with greater distance from the initial utility connection. This slight drop in voltage lengthens the time necessary for distant fixtures to ignite, and creates a multiscale aspect to at least some parts of the transient profile as a function of lamp bank size.

Notice that in steady state, the instantaneous power is essentially non-negative. Ignoring slight, negligible harmonic distortion not visible in the figure, the power factor of the bank in steady state is practically unity. The lamp ballasts contain carefully sized capacitors that compensate the inherent steady state reactive components of the lamp and ballast combinations, making each lamp look like a nearly resistive load in steady state operation. Before the lamp enters steady state, however, the impedances of the lamp and ballast combinations are different from their nominal steady state values. During this time,

the compensating capacitor is not correctly sized to correct the power factor of the lamp fixtures, and the power factor of the bank during the preheat period is definitely not unity. This is clearly indicated in Fig. 2.2 by the presence of negative instantaneous power during the preheat phase between time zero and time one. So, while a fairly inexpensive compensation scheme masks the inherent power factor of the load in steady state, the overall transient profile reveals a wealth of information that can be used to associate the observed profile with a known load class. This example pointedly suggests why a transient event detector is likely to expand the applicability of the NILM.

2.3 Electromechanical Loads

This section considers examples from the category of *electromechanical* loads, whose transient profiles are shaped by the process of transferring mechanical energy to or from some part of the load. There are many load classes within this category. Here, the focus is on the three most common types of rotating electric machines: DC machines, induction machines, and synchronous machines.

These load classes account for a large part of industrial and commercial energy consumption. The tenor of many of the observations made in this section holds for other potentially important electromechanical loads, such as the variable reluctance machine (VRM). Because the VRM is usually operated with a power electronic drive, many of the comments made in the next section are also relevant to this increasingly important load class.

2.3.1 DC Machines

Advances in the last two or three decades in the application of power electronic drives to the control of other machine types have not entirely supplanted the use of DC machines. The application of DC machines is still not uncommon, for example, in air handling components in ventilation systems. This section therefore begins with a look at DC motors. DC generators tend to be applied in very special purpose applications or as internal components of more complex loads. They will not be considered in detail here.

There are a variety of techniques for operating DC motors which differ in the way

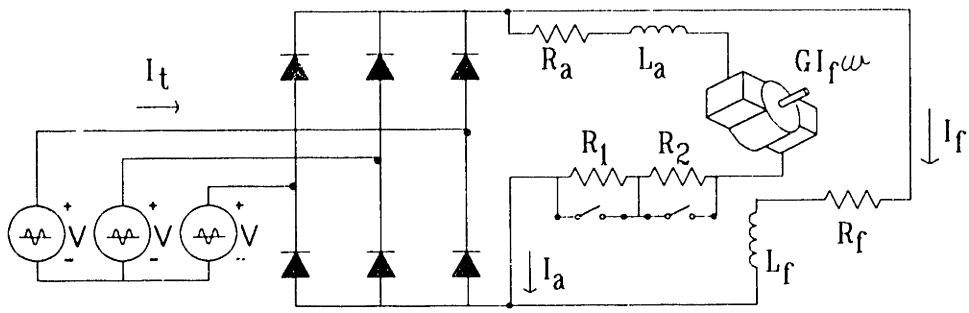


Figure 2.3: Shunt DC Motor Schematic

the field and armature circuits of the motor are interconnected, including shunt, series, and compound connections. Each of these connections will generally exhibit transient profiles that are different from each other and useful for identifying load class. Because the profiles are highly dependent on the operating protocol (i.e., the precise manner in which the motor is started and loaded), it is difficult to succinctly categorize all possible transient profiles. An example will serve here to illustrate many key points.

A shunt connected DC motor with a stepped resistance starter is shown in Fig. 2.3. The three voltage sources labeled V in the figure represent a balanced three phase voltage set supplied by the utility. The current I_t denotes the line current on one phase of the AC electrical service. This current represents what might be seen at the utility service entry when the DC motor is operating. A three phase rectifier bank provides DC voltage with a small amount of ripple to the field and armature circuits of the motor. The model for the field circuit, with current I_f, consists of a lumped field resistance R_f and a field coil inductance labeled L_f in the figure. The armature circuit model, with current I_a, consists of a lumped resistance R_a and an inductance L_a in series with a "motor shaped" voltage source which represents the back-EMF of the motor. The internal voltage or back-EMF of the motor is dependent on rotor speed and field current and is indicated in the figure by

the product $GI_f\omega$, where ω is the angular rotor speed and G is a constant that depends on non-varying physical details of the motor's construction.

As with all DC motors, the transient profile is critically dependent on the protocol used to start the motor. For a shunt connected machine, the internal voltage is initially zero at turn–on. The field circuit time constant is usually short and the field current reaches a steady state level quickly in comparison to the time needed to accelerate the rotor. The armature time constant is also fairly short and the in-rush current would be huge for all but small motors if the motor were started by simply connecting it across the rectifier bank. To avoid this, a number of control strategies may be used to start the motor. One simple and useful scheme is shown in Fig. 2.3. The control resistors R_1 and R_2 have been added in series with the armature circuit to increase the net armature resistance and limit the in-rush current. As the rotor accelerates and the internal voltage rises, the control resistors are shorted by the parallel connected switches shown in the figure. The switching instants may occur at predetermined times, or based on armature current level or internal voltage level. The effect of the control resistors is to bound the maximum excursions of armature current at the cost of lengthening the time necessary to reach rated speed. The number of control resistors and shorting switches used is a parameter of the control scheme which may be varied to ensure different maximum current levels and acceleration times. Details of this and other control schemes may be found in [37], [73], and [75].

The results of a RAPID simulation of a five horsepower, shunt connected DC motor are shown in Figs. 2.4 and 2.5. The electrical parameters of the motor are presented in [73]. The simulated system is essentially identical to the one shown in Fig. 2.3, with the exception that four control resistors and switches have been employed in the simulation instead of two. In the simulation, the motor is driving a load whose torque demand rises linearly with rotor speed to rated torque at rated speed.

Figure 2.4 shows $I_a + I_f$, the sum of the armature and field circuit currents, during the turn–on transient. This curve is essentially the positive envelope of the current I_t that would be observed at the service entry. The shape of the transient is very distinctive, and a strong function of the control strategy used to start the motor. A pulse in armature current occurs each time one of the four control resistors is shorted. For reference, Fig. 2.5 shows the rotor speed throughout the turn–on transient. The transient profiles for subclasses of DC

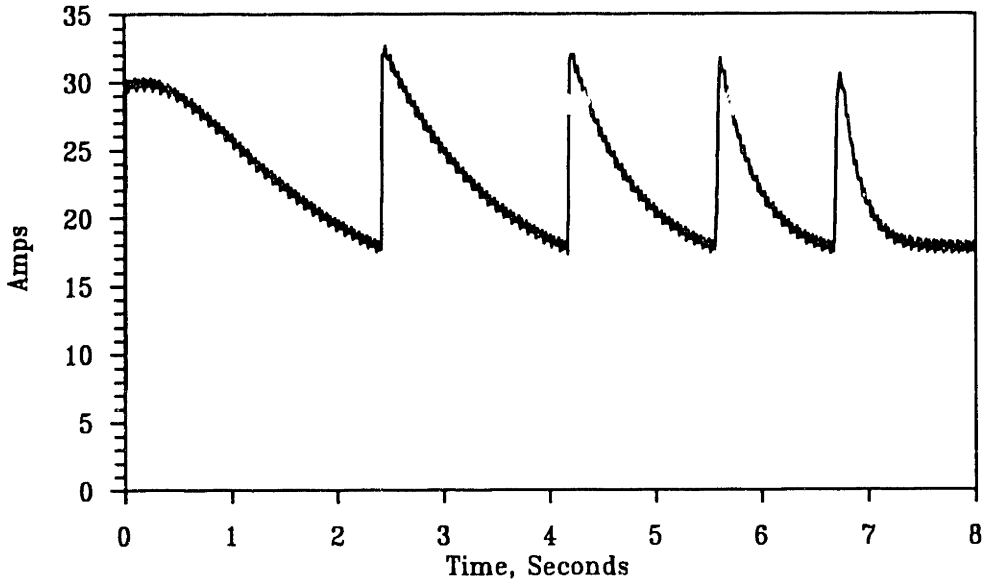


Figure 2.4: DC Motor Input Current Transient (Simulated)

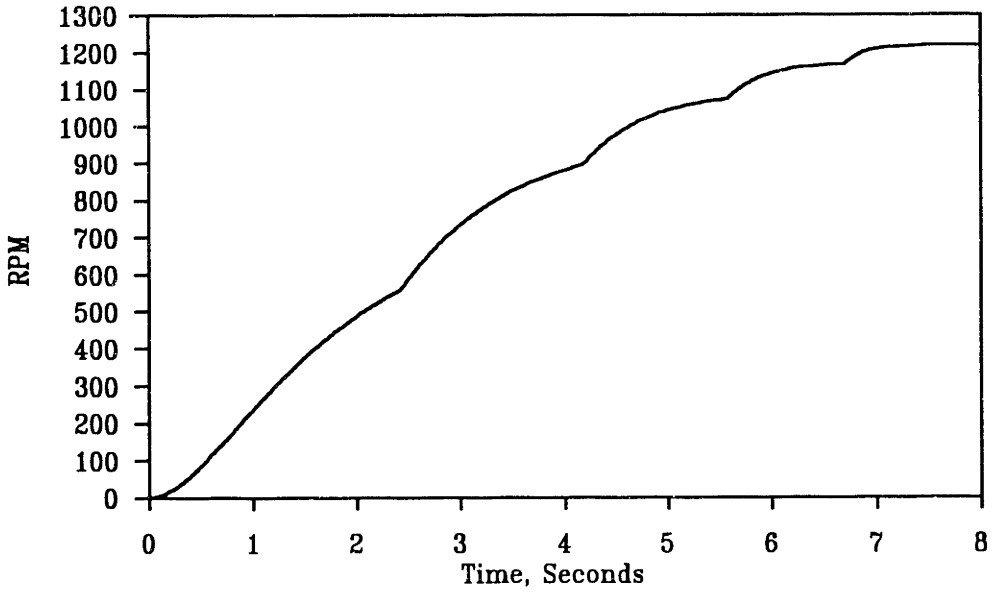


Figure 2.5: Rotor Speed Transient (Simulated)

motors like shunt connected motors may exhibit multiscale behavior, but this will be highly dependent on the control techniques and mechanical loads involved. Pattern templates for DC machines for use in the recognition schemes t be introduced later in this document may best be determined by a one-time training session at the site to be monitored.

2.3.2 Induction Machines

Induction machines are relatively inexpensive, reliable, available over a wide power range, and are by far the most prevalent rotating electric machine in industrial use. Three phase induction motors are self-starting and relatively benign in terms of the peak current magnitude drawn during the turn–on transient. For this reason, they are often, although not always, activated by a direct connection to a balanced three phase electrical service.

This section will examine three phase induction motors operated directly from a three phase service. However, other types or configurations of induction machines may be found in practice. Single phase induction motors are popular for low power applications. There are numerous starting schemes for these motors, many of which involve briefly running the motor as a two phase motor from a temporary quadrature voltage source created by a starting capacitor. This starting circuit may be dropped out once the rotor has achieved sufficient velocity to permit single phase operation. Many of the observations made here hold true for most single phase induction motors. Variable speed drives based on induction motors are becoming increasingly popular, especially as components in modern ventilation systems. The transient characteristics of these drives are usually distinctive but dependent on the control strategy employed. Induction generators are uncommon.

Figure 2.6 shows the no-load turn–on transients computed by RAPID for three different, three phase induction motors: a 5, a 7.5, and a 10 horsepower machine. The plots show instantaneous power on one phase of a three phase service for each of the motors. These particular motors are designed to operate with relatively low steady state slip. The electrical parameters for the machine models used in the simulations may be found in [73]. Details of the simulation model may also be found in [73] and [74].

The transients in Fig. 2.6 exhibit several striking features. Once again, the patterns are distinctive for the load class. In all cases, the machines draw a large magnitude of instantaneous power while the rotor is accelerating to rated speed. Because the machines

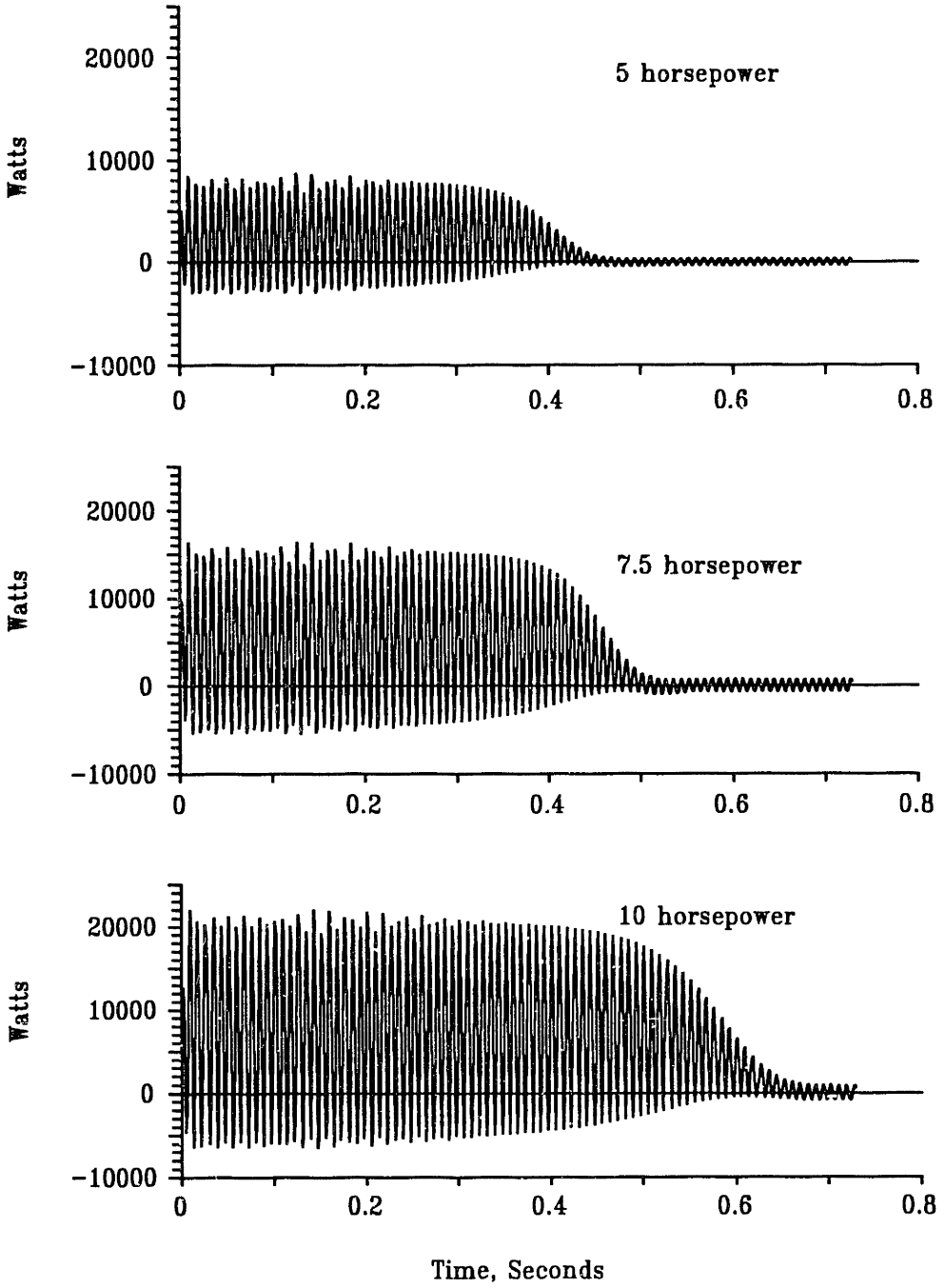


Figure 2.6: Three Induction Motor Transients (Simulated)

are unloaded, the power demands drop precipitously as steady state operation is approached. Each motor draws a substantial amout of real power during the acceleration period. Because the instantaneous power is periodically negative during this time, it is also clear that the motors do not operate at unity power factor. In steady state, the average value of the instantaneous power is approximately zero, and the power factor of each motor is very low. Notice the similarity of the shapes of the transient profiles. The slow envelope of the transients for all three motors could reasonably be represented by a single transient shape scaled in amplitude and time.

This starting characteristic is typical of many applications of large induction machines. To avoid excessive starting currents, the motor is often permitted to reach near steady state speed before any significant load torque is engaged. For example, in pumping applications, a centrifugal valve may be employed to ensure that no pressure is applied until the motor is at rated speed. In large machine tools, an induction motor drive is often protected with a flywheel and transmission which does not engage any significant load torque until a sizable, moderating mechanical energy has been stored in the flywheel.

Naturally, the transient shape presented in Fig. 2.6 might not represent every induction motor application. Nevertheless, experiments with RAPID and actual induction motors in practical applications indicate that a handful of prototype transient profiles could represent most common subclasses of induction machines and many loading conditions. Figure 2.6 illustrates the case for low-slip three phase machines accelerating under no-load conditions. Other templates could represent high-slip machines, variable speed drives, single phase machines, and so on. Multiscale behavior can potentially be present in all of these subclasses.

2.3.3 Synchronous Machines

Synchronous motors tend to be applied either in fairly low or very high power applications. At medium power levels, they are usually not competitive with induction machines. When paired with power electronic drives, small synchronous machines, especially those with fixed field excitation as in the case of a permanent magnet machine, are increasingly popular for servomechanical applications. The true synchronous motor generates torque only at speeds synchronous with the excitation frequency. A common starting scheme for small

synchronous motors, therefore, involves a variable frequency drive that sweeps the armature excitation from near zero frequency at turn–on to rated speed in steady state. This tends to yield a dramatic transient profile that can be used to identify the load class. Again, the profile is likely to be subclass specific.

For very high power applications, the synchronous motor occasionally proves to be more economical than the induction motor. Rotor excitation in the induction machine is often provided solely by induction from the stator (there are exceptions, i.e., doubly excited induction machines). Relatively narrow air gaps must be maintained for efficiency, and this complicates the construction of very large, singly excited induction machines. Large synchronous machines, on the other hand, are always doubly excited and larger air gaps may be tolerated without intolerable efficiency penalties. However, these larger machines are still not inherently self-starting.

In these cases, the rotor of the synchronous machine is typically constructed with *damper* or *amortisseur* bars that serve to enhance the transient stability of the machine for perturbations around steady state operating conditions. When away from synchronous frequency, the amortisseur bars act in a manner similar to a squirrel cage induction motor rotor. When the rotor nears synchronous speed as the "induction-like" transient is nearing its end, the rotor will jump to synchronous speed if the loading conditions are not too severe. Large synchronous machines, therefore, may exhibit transients similar to those of the induction motors examined in the previous section. Depending on the machine and the loading conditions, there may be sufficient differences to distinguish synchronous motors from other motor types. In general, large synchronous motors are not widely prevalent, and, when found, tend to be isolated on independent branches of a building's electrical harness, greatly facilitating nonintrusive identification. Further field studies may be necessary for a thorough categorization of any important applications of synchronous motors with regard to nonintrusive monitoring.

Synchronous generators, on the other hand, may be found over a somewhat wider power range. Very large generators, like very large motors, tend to be found on isolated legs of a wiring harness, and should not be a problem for nonintrusive monitoring. Moderate to large synchronous generators, on the other hand, are often operated unloaded with overexcited field windings. In this case, the generator acts as a source of reactive voltamperes

and may be used to correct the power factor of inductive loads on the local wiring harness. These generators operate with near zero power factor and should exhibit distinctive transient patterns. Again, however, the precise patterns will depend on the operating protocol, and should be a matter of further field study.

2.4 Power Electronic Loads

The growing use of power electronics in practically all classes of loads suggests that a closer look at the impact of power electronics on transient profiles will be valuable. As has been discussed in previous sections, power electronic circuits are often employed as drives for electrothermal or electromechanical loads. Actively controlled heating elements, power electronic lamp ballasts and variable speed motor drives are all examples of loads in which power electronic circuits work in conjunction with devices that bring their own unique dynamics to the total system. The transient profiles of these "hybrid" loads will generally be influenced not only by the dynamics of the thermal or mechanical system but also by the control or operating protocol employed by the power electronic drive. Other loads might be considered "pure" power electronic loads in the sense that the power consuming component of the load is very simple, e.g., adequately modeled as an LTI resistor, or masked by the power supply. In such a case the transient behavior of the load is almost completely determined by the operation of the power supply or power electronic drive. In either case, the inclusion of a power supply or drive typically has profound effects on the transient profile of a load.

One very important power electronic load is the personal computer. The transient behavior of the personal computer is broadly representative of may common types of home and office electronic equipment. The current during the turn–on transient of a typical personal computer is shown in Fig. 2.7.

As for many other load classes, the slow envelope of the current waveform exhibits a distinct shape that would help permit load classification based on observation of the current waveform. This slow envelope is created by an in-rush current that charges an internal bus capacitor in the power supply, and by the sequencing of internal loads in the computer. An additional, striking feature of the transient shown in Fig. 2.7 is the decidedly non-sinusoidal

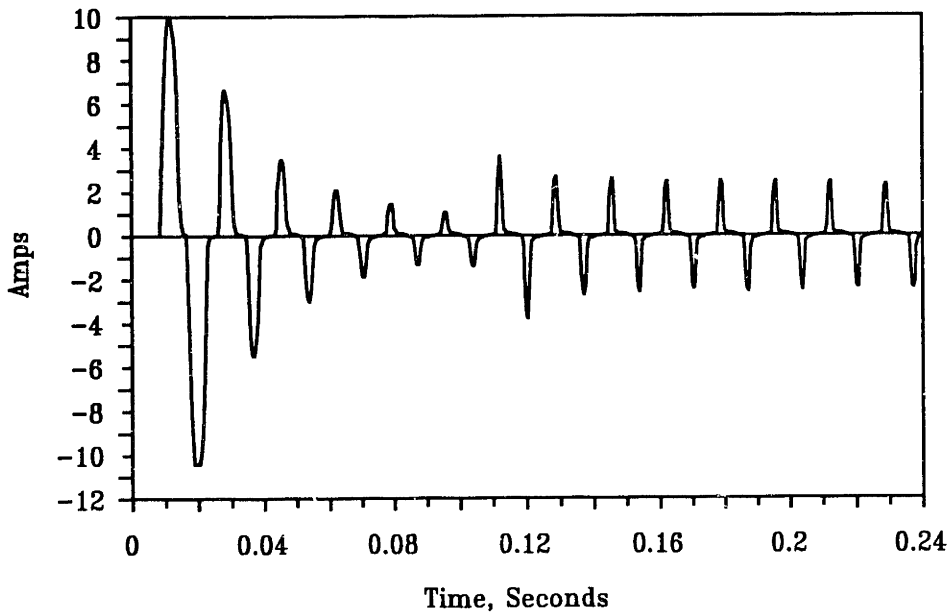


Figure 2.7: Personal Computer Transient (Measured)

current shape. (To permit close examination, Fig. 2.7 plots current directly, instead of the instantaneous power shown in previous plots.) The current waveform clearly contains substantial harmonic content at frequencies in excess of the fundamental 60 Hertz frequency. These harmonics are introduced because the first stage of the switching power supply in the computer consists of a full-bridge rectifier bank charging a large bus capacitor In particular, a large third harmonic component is present. Higher harmonic currents, therefore, as well as the slow envelope of real and reactive power, may be strong indicators of load class.

The introduction of higher harmonics is not unique to personal computers and similar forms of office equipment. When coupled with power electronic drives, electrothermal or electromechanical loads may exhibit load transients and steady state currents with higher harmonic content. Consider, for example, the steady state current waveform for a compact rapid start fluorescent lamp shown in Fig. 2.8. Once again, the harmonic content of the current waveform would appear to be a very important clue in determining load classification.

Steady state currents with higher harmonic components can be detrimental to the quality of the utility service. Harmonic currents flowing through the impedances associated with the utility wiring and distribution network have the effect, at least locally, of generating

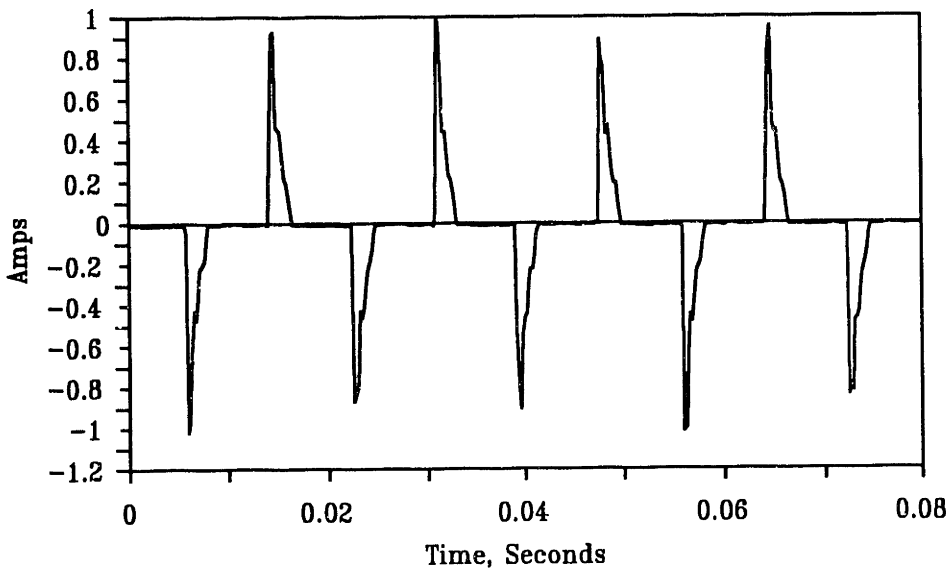


Figure 2.8: Compact Fluorescent Lamp Steady State Cycles (Measured)

harmonic voltages that distort the shape of the voltage waveform. For this reason, there is a rising interest in the application of power supplies which employ active waveshaping to minimize higher harmonic content in the demanded current waveform. Fortunately for the effort of constructing a transient event detector for the NILM, these waveshaping or power factor correction techniques alter but generally do not eliminate telltale signs during the turn–on transient. Consider, for example, Fig. 2.9, which shows the partial turn–on transient of a unity power factor power supply driving a resistive load.

The power supply that created the transient shown in Fig. 2.9 is a boost switch mode power supply controlled by a commercial power factor correction integrated circuit. Following an in-rush current not shown in the figure, the power supply must draw several "uncorrected" or spikey cycles of line current while a bootstrapping power supply and the boost power supply bus capacitor are charging. Three line cycles of uncorrected current are shown at the beginning of the transient waveform in Fig. 2.9. When the controller begins actively shaping the current waveform, the current shape becomes nearly sinusoidal. A slow, rising envelope of corrected current is created by the "soft-start" circuitry in the power supply controller, which works to minimize stresses on the components of the power supply. When the output voltage of the power supply reaches a nominal level, the current

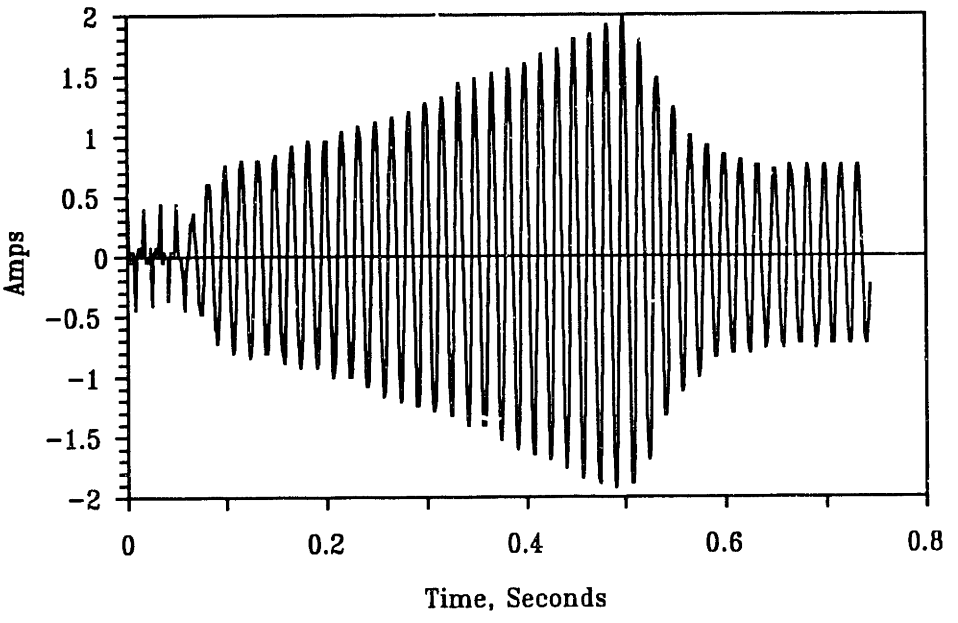


Figure 2.9: Power Factor Correcting Power Supply Transient (Measured)

drops to a steady state level appropriate for the loading condition. In steady state, the current waveform is practically indistinguishable from the waveform that would be drawn by a comparably sized resistive load placed directly across the line.

Clearly, it is possible at some cost to develop a power electronic solution to avoid distorted line current waveforms like those shown in Figs. 2.7 and 2.8. However, it is generally neither cost effective nor necessary to attempt to mask the telltale transient behavior of the combined power supply and load. Notice also that the transient in Fig. 2.9 is a simple one, created mainly by the dynamics of the power supply controller. For a more complicated load like a motor or a fluorescent lamp bank, the slow envelope of corrected current will be even more distinctive and revealing. It is essential to examine the transient profile. The transient profile contains critical information that would be missed by simply looking at steady levels after transients had occurred.

2.5 Summary

This load survey indicates that a large number of important load classes may be expected to exhibit revealing transient profiles. Both the slow envelope of these profiles as well as

higher harmonic content will be useful for matching patterns with load classes. Many of the load classes examined in this chapter exhibit multiscale transient behavior. That is, loads in a particular class which spans a wide power range often exhibit transient profiles that are identical in shape but scaled in amplitude and duration. The multiscale nature of the transient profiles of many load classes will be exploited later in this document to enhance the efficiency and minimize the storage requirements of a prototype transient event detector.

Chapter 3

Identifying Observed Patterns

For many common classes of devices, the characteristic turn–on transients of different load classes are apparently sufficiently distinct and repeatable to constitute sound bases for associating observed patterns with loads. This chapter considers the practical problem of generating and identifying patterns from the information available at the utility service entry of a building. In particular, an important goal will be to develop techniques that permit the maximum possible rate of event generation consistent with reliable identification. The chapter focuses predominantly on techniques for matching a pattern at a particular, known time scale. However, when the detection scheme developed here is operated in a multiscale environment, where similar patterns may occur over a variety of time scales, some special considerations are necessary and will be highlighted where appropriate. The next chapter will introduce specific techniques for formally addressing the multiscale pattern matching problem.

3.1 Data Acquisition and Preprocessing

In a typical, moderate to large size commercial or industrial building (10,000 to 100,000 square feet), the electrical service will typically be three phase. At the service entry of a building, the commercial NILM will sample the utility voltage waveform and also the aggregate current waveform drawn by the loads on each of the three phases in all or part of the building. The case of split single phase service, common in residences, could be addressed as a special case of the techniques to be developed here. As loads turn on and off, the current waveform observed by the NILM will presumably contain transient shapes like

those examined in the previous chapter. A variety of preprocessing steps may be necessary or desirable before performing pattern matching in order to ensure that the observations at the service entry fit reliably and consistently into an anticipated pattern space.

In between the service entry and some or all of the loads in a building there may be transformers, which serve to modify voltage levels or to electrically isolate parts of the building's distribution network. In particular, the current waveforms observed on the primary side of a three phase transformer may have different phase and amplitude characteristics from the current waveforms at the load terminals on the secondary side of the transformer ([71], [129], [130]). If the NILM is trained to search for patterns observed during a one-time training period on-site, the details of the building's internal wiring harness are largely irrelevant. These details may be more significant if the NILM is to monitor a building with a predetermined, off-site collection of transient template shapes. In this case, the NILM may require some way of "inverting" the effects of internal distribution fixtures like transformers. The RAPID program described in Appendix B contains software routines for computing the currents on the secondary side of a transformer given the primary side currents, and vice versa. The NILM could use these routines to modify predetermined transient templates to account for details of the building's distribution network during the installation phase.

The remainder of this chapter will be directed toward defining and solving the pattern recognition problem on a single phase, with the understanding that a practical NILM will generally monitor for pattern sets on more than one phase, and with the implicit understanding that real, observed data may need to be modified to account for the details of the building's wiring harness.

The review in the previous chapter indicates that the shape of the raw current waveform drawn by a load is indicative of load type or class. However, direct examination of the current waveform or a closely related waveform like that of instantaneous power (i.e., the product of the voltage and current) fails to accentuate important features for pattern recognition. It is very important to isolate key features from near constant frequency, "carrier wave" type signals like 60 Hertz input current or 120 Hertz instantaneous power. Otherwise, recognition systems looking for a modulating envelope may be overwhelmed by the presence of the carrier frequency. That is, very slight, irrelevant errors in matching the carrier frequency variation with a template will dominate the results of a recognition system that

should be searching for a modulating envelope.

The approach adopted here for eliminating carrier frequency artifacts from input data is to work with short time estimates that average over at least one carrier frequency period. For example, an estimate of the envelope of real power can be computed by first computing the instantaneous power by multiplying or mixing the input current with the input voltage, and then filtering the product with a low-pass filter whose cutoff frequency occurs significantly below the carrier frequency – 120 Hertz in this case. Only the slow envelope or estimate of the time average of instantaneous power, i.e., real power, will remain. Other relevant quantities may be computed in a similar manner. An estimate of the envelope of reactive power may be found by mixing the current with a sinusoid that lags 90 degrees out of phase with respect to the input voltage, and then low-pass filtering the product. Approximate envelopes of higher harmonic content in the current waveform may also be found, by mixing the current with a sinusoid at a frequency above the fundamental frequency of 60 Hertz and then low-pass filtering.

One method for generating quadrature sinusoids in practice is to integrate the voltage waveform. Higher harmonic sinusoids may be generated by using a phase–locked loop to track the input voltage and generate a higher frequency output sine wave. Precise details of an analog circuit implementation for generating envelope estimates of real and reactive power, and of in-phase and quadrature third harmonic content in the current waveform, will be presented in Chapter 5 and Appendix D. Additional interesting considerations for a variety of estimates of real and reactive power for arbitrary waveform shapes over arbitrary time intervals are given in [155].

Another important preprocessing concern arises from the fact that the peak line voltage will vary slightly. When the voltage changes, many loads will draw a different current in both transient and steady state operation. It may be important, therefore, to normalize estimates of the envelopes of observed data to reject those variations that are due to voltage fluctuation. For example, for an estimate of the envelope of real power, dividing the estimate by the square of the RMS utility voltage yields an estimate of load admittance which is in principle immune to voltage fluctuation, assuming that the loads are well modeled as linear circuit elements. In practice, the admittance of many loads is a function of applied terminal voltage, and other normalization schemes or approximations

may yield better results; see [57] and [137] for more details.

Finally, when the transients to be identified are step-like, a multiscale edge detector could be incorporated into the data preprocessor of the NILM to expose the location of transient events. The edge detector can expose sparsity in the input data, potentially freeing computational resources when the rate of event generation is low. Since more complicated pattern recognizers would only have to be engaged when an edge was detected at some time scale, extra processing time available to the NILM could be used for report generation or other bookkeeping operations. However, if the monitoring site generates events at a constantly high rate, the edge detector might be a waste of time. The possible value of a multiscale edge detector remains an area of research, as more data concerning the pattern of event generation rates in a variety of different buildings will need to be collected and analyzed. Multiscale edge detectors based on the Canny edge detection algorithm ([19]) are discussed in [121]. Preliminary experiments with a multiscale edge detector for use in the NILM have been conducted and are reported in [70].

3.2 Approach to Transient Recognition

One approach for associating observed transients in the preprocessed data streams with known load classes might be to insist that the NILM will only operate in environments where the service entry settles to clean steady state levels before and after each transient. No two transients could overlap to the extent that critically important features of either transient would be distorted. This condition would delineate each transient and allow for relatively easy extraction and identification. This scheme is undesirable because it limits the tolerable rate of event generation in the target environment, especially in a multiscale setting, where some transients may be quite long.

Rather than searching for a particular transient in its entirety, the commercial NILM will search for a time pattern of *segments* that correspond to a particular transient. The approach adopted here is in some sense a dual of techniques used to solve speech recognition problems. In many speech recognition applications, the speech waveform is segmented into short "stationary" regions with constant mean and variance that can be well represented, for example, by an autoregressive (AR) model. The parameters of an AR model, sometimes

called the linear predictor coefficients (LPCs), characterize a representation for the speech waveform in each short segment. A typical approach to word or utterance recognition is to segment the utterance into stationary segments, describe each by a set of LPCs, and then match the observed time pattern of LPCs to a library of known LPC pattern templates (see [64] and [100] for examples). Instead of searching for a time pattern of segments that are stationary or "fixed" in attributes, the NILM will search for regions of significant change embedded in quasi-static regions of a transient shape. (Of course, unlike speech recognition applications, the NILM will also require a systematic scheme for searching over a wide range of time scales; a multiscale search technique will be presented in the next chapter.)

A new approach to identifying transients, then, would be to search for sections of a transient with significant variation, or *v-sections*. During a training phase, either before installation or on-site, the NILM will employ a change-of-mean detector ([11]) or other waveform segmentation technique to segment a finest scale transient that represents a class of loads. This segmentation process delineates a set of v-sections that will represent a particular transient shape in each of the input data streams. Examples are shown in Figs. 3.1 and 3.2. The trace in Fig. 3.1 shows the envelope of real power during the turn–on transient of a rapid start fluorescent lamp bank. The trace in Fig. 3.2 shows the envelope of reactive power during the turn–on transient. Both waveforms are real data collected from an actual lamp bank with the analog preprocessor and *NILMscope* software described in Chapter 5. The locations of the v-sections in the two waveforms, as computed by a change-of-mean detector implemented in MATLAB, are approximately indicated by the the five ellipses labeled A–E in the figures.

A complete transient identification will be made by searching for a precise time pattern of v-sections. The v-sections are relatively narrow, and there is reason to expect that such sections will be less likely to be fatally corrupted than would an entire and lengthy transient. To see this more clearly, consider the following example. Figures 3.3 and 3.4 show the real power transients during turn–on for an instant start fluorescent lamp bank and an induction motor, respectively. Again, the traces are real data from actual loads. The single v-section in the instant start pattern is marked with an ellipse in Fig. 3.3, and the two v-sections in the motor pattern are marked with rectangles in Fig. 3.4. The v-sections in other data streams like reactive power will be ignored for this simple example. As long as each of the

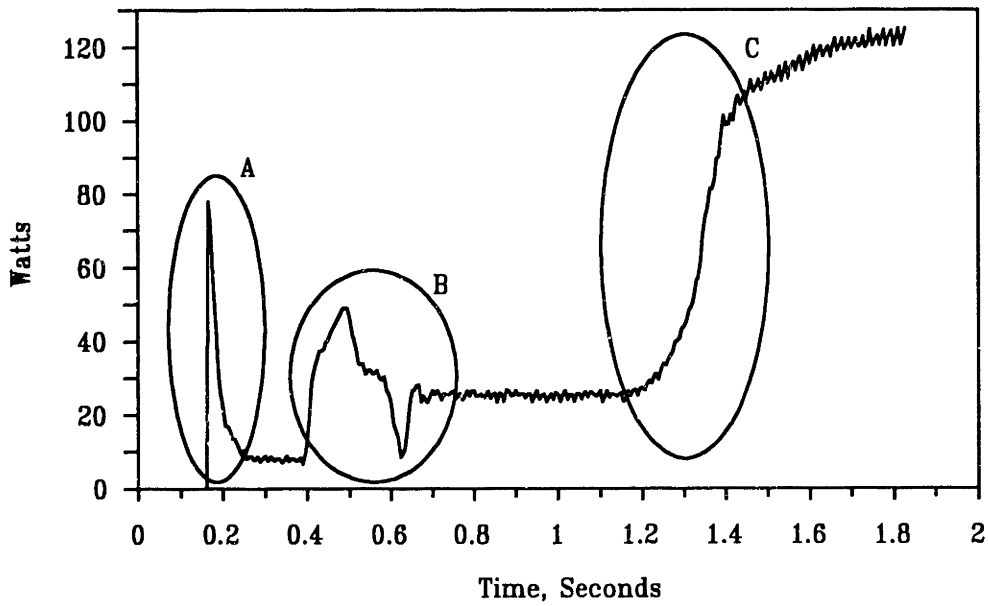


Figure 3.1: Rapid Start Lamp Bank Real Power Transient

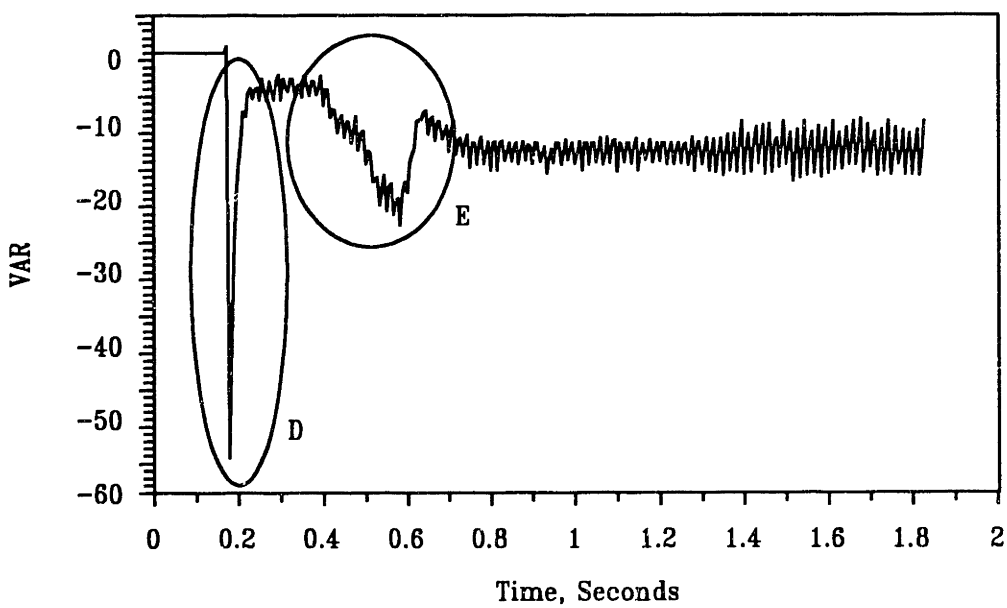


Figure 3.2: Rapid Start Lamp Bank Reactive Power Transient

v-section shapes is clearly distinguishable, the NILM will be able to identify the patterns of v-sections and therefore the transients. For example, the overlap of the two transients shown in Fig. 3.5 is tractable because all of the v-sections for both transients are distinguishable. That is, each v-section overlaps additively with a region that is relatively featureless and quasi-static. The overlap condition in Fig. 3.6 would not generally be tractable, since the instant start v-section and the first induction motor v-section overlap severely.

Since some degree of overlap is now tolerable, the v-section set recognition technique will generally operate successfully in a busier environment with a higher rate of event generation than would a NILM searching for whole, undisturbed transient shapes. For transient identification to work reliably, the transient patterns need to be relatively unique and reasonably repeatable. This has two implications for the v-section search technique. First, it will be advantageous to search as many data streams as will provide useful information, i.e., interesting v-sections. Envelopes of real and reactive power, as well as higher harmonic content, should all be considered as possible sources of v-sections when developing prototype v-section templates for use in the NILM. Second, if the v-sections are not repeatable or informative, the pattern search will not be robust and efficient. For instance, v-sections whose shapes are extremely dependent on the turn–on time for the load should probably not be included in the prototype template for that load. Imagine examining a collection of turn–on transients for a rapid start lamp bank like those in Figs. 3.1 and 3.2. Suppose that this examination revealed that the v-section labeled D in Fig. 3.2 changed shape considerably based on turn–on time, but that the other v-sections were functions of actively controlled processes in the lamp and lamp ballast and appeared quite consistently. Also, if the v-sections labeled B and E always appeared together in lock step, one of these v-sections might be discarded from the search set to minimize computational effort *if* it could be removed without creating a potential confusion with another load's v-section set or a combination of sets. Under these conditions, the search set might be limited to v-sections A, C, and E, which would constitute a reasonably time-invariant template satisfactory for positively identifying the turn–on of a rapid start bank.

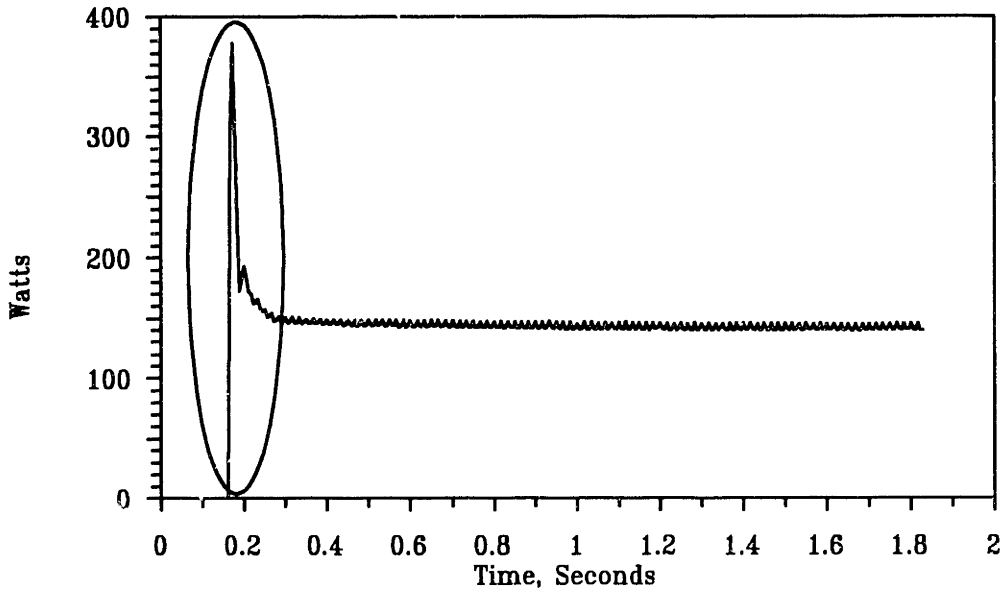


Figure 3.3: Instant Start Lamp Bank Real Power Transient

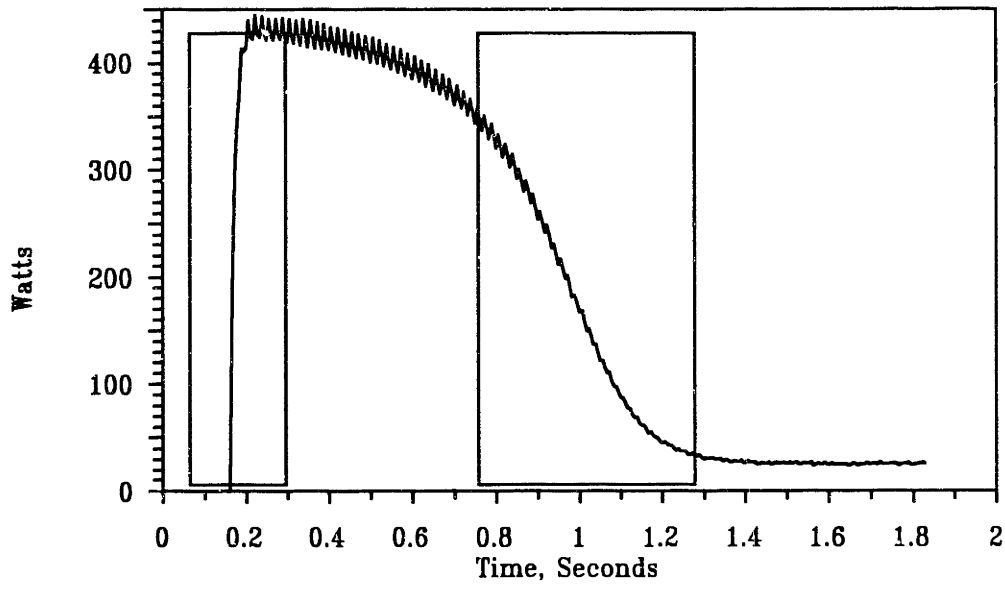


Figure 3.4: Induction Motor Real Power Transient

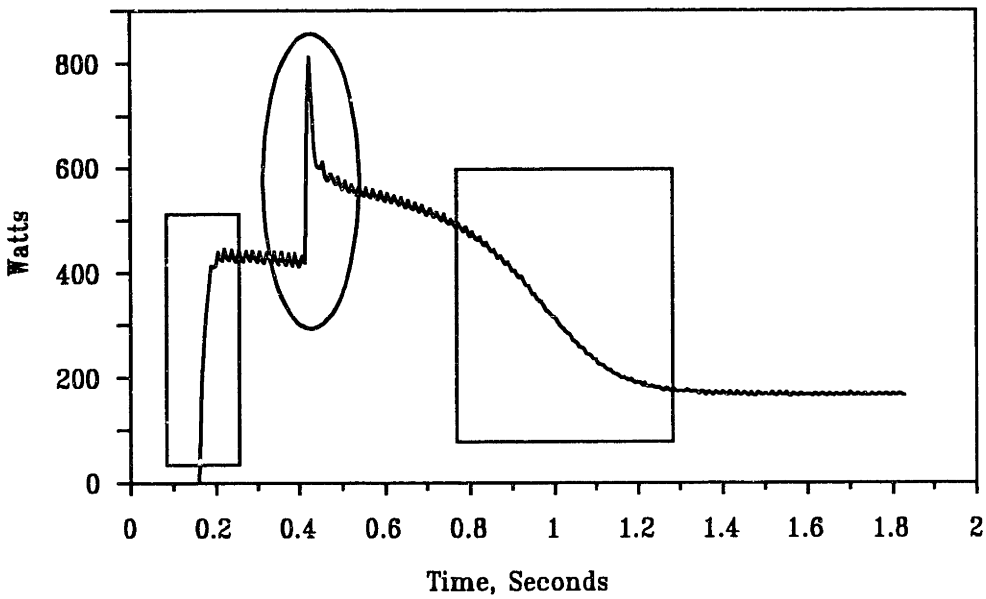


Figure 3.5: Acceptable Overlap

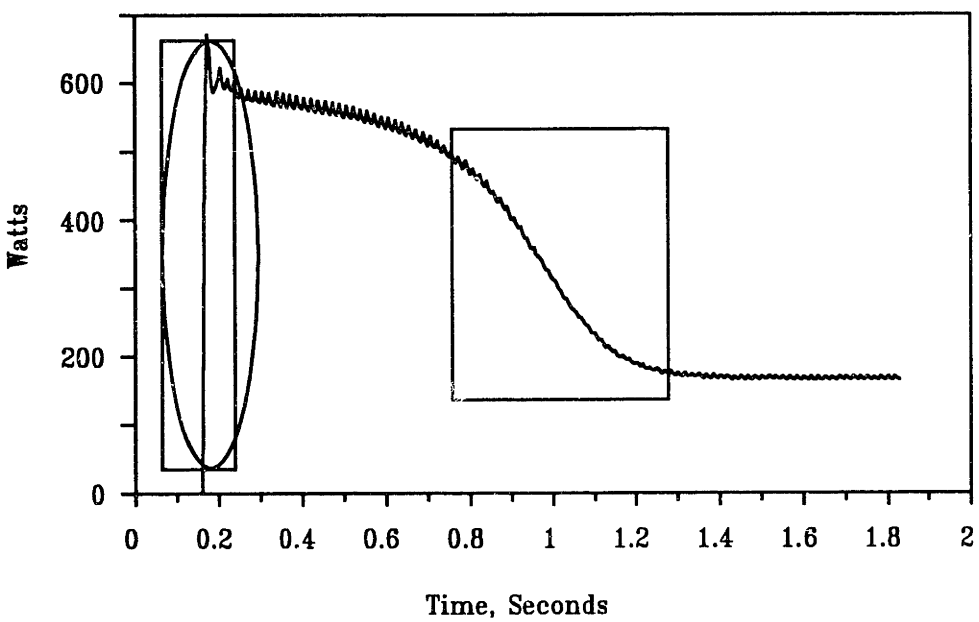


Figure 3.6: Intractable Overlap

3.3 Pattern Discrimination

Classification of individual v-sections in the input data streams will be determined by a set of discriminator functions. Discriminators partition the state space of all possible input data into regions or classes. The discriminator functions are used to compute a distance metric that locates a particular input vector in a region of the state space. Since a v-section may appear on top of a variably large static or quasi-static level created by the operation of other loads, the discrimination process will need to focus on only the "AC" or varying component of the v-section.

Some of the simplest pattern discriminators compute a distance metric between a vector of input data and a template vector. For example, the Euclidean metric computes the norm of the difference between two vectors. Another simple discriminator that has been used extensively for scene analysis ([35]) and image recognition ([33], [63], [83], [127], [69]) is the transversal filter. The impulse response of a transversal or matched filter is proportional to the time-reversed signal for which the filter is designed to search. In operation, the transversal filter computes the inner product of a vector of input data with a template vector. Because there are many highly efficient hardware solutions available for implementing a transversal filter, and because of its conceptual simplicity, the transversal filter is an attractive signal processing construct for performing pattern discrimination. For the NILM, each v-section will be positively identified by checking the outputs of two transversal filters.

The first transversal filter scans an input data stream for a particular shape. Let t denote a vector of N samples of a v-section of interest, such as one of the v-sections marked in Figs. 3.1 and 3.2. The vector t consists of elements

$$t[i], \quad i = 0 \ldots N-1.$$

An "AC coupled" and amplitude normalized version of the v-section, designated t_{ac}, can be computed as

$$t_{ac} = \frac{\left[t - \frac{1}{N}\sum_{i=0}^{N-1} t[i]\right]}{\| \left[t - \frac{1}{N}\sum_{i=0}^{N-1} t[i]\right] \|} \tag{3.1}$$

where the denominator in Eq. 3.1 is the 2-norm of the numerator. Thus, t_{ac} is a unit length vector with zero mean. This shape transversal filter operates by sliding an N-point

window across the input data stream. At any time, the window contains an AC-coupled and amplitude normalized vector of points x_{ac} of an N-point section or vector x of the input data. The vector x_{ac} is computed from the vector x using an equation structurally identical to Eq. 3.1. The output of the shape transversal filter is the inner product of the template vector t_{ac} and the data vector x_{ac}.

This output corresponds to the cosine of the angle between the two unit length vectors t_{ac} and x_{ac}. An output of unity indicates a perfect match between the template vector and the input data. In this case, the shape of the input data precisely matches the shape of the template of interest. Naturally, noise and slight variation in the repeatability of the v-sections will make a perfect match unlikely. In a practical system, then, some degree of imperfection will be tolerated and any inner product within a certain match tolerance of unity – i.e., between unity and some positive number less than unity – will constitute a match. In geometric terms, the match tolerance has the effect of defining a hypercone around the template vector t_{ac} in the pattern space, within which an input vector x_{ac} may fall and still be considered a match.

The match tolerance should be large enough to ensure that a reasonable degree of variation in the presentation of v-sections in the input data can be handled. On the other hand, the match tolerance should be small enough so that template vectors for other v-sections in the pattern space will not have inner products with the input data that could lead to numerous false matches and erroneous identification. Depending on the underlying physical processes occurring in the loads, different v-sections may exhibit different ranges of variation, and different match tolerances may be appropriate for different v-sections. More than one template vector could be employed to narrow the acceptable match region in the pattern space. A match with all such templates within a certain match tolerance would be required before an input vector would be identified as a particular v-section.

When a segment in the input data stream is found that matches the shape of a template v-section, a second transversal filter would be employed to check the amplitude of the segment. This amplitude transversal filter computes the inner product of the template vector t_{ac} and the input data vector x, rather than the inner product of t_{ac} and x_{ac}. By itself, the amplitude transversal filter would not be a satisfactory pattern discriminator. Since its amplitude is not normalized, a vector x could be practically orthogonal to the

template vector, but still yield a large inner product with the template. However, checking the amplitude is essential in conjuction with checking shape to ensure that a small wiggle or noise pattern that is fortuitously close in shape to a v-section template is not mistaken for an actual v-section. The inner product of t_{ac} and x is not amplitude bounded, so the value of the inner product that corresponds to a match could be any positive number. The correct match value range would, in practice, be determined *a priori* using the inner product of t_{ac} and t as the ideal amplitude match value. Since the template vector t_{ac} has zero mean, the inner product of t_{ac} and x will still indicate only the AC content of the input data and will not, in principle, respond to constant levels.

A v-section would be considered identified or matched in an input data stream only if the input segment yielded transversal filter outputs within acceptable match ranges for both the shape and amplitude transversal filters. The shape and amplitude filtering operations could actually be performed in either order, as long as both operations were checked before considering an input segment as matched. However, one of the filters might yield more "false positives" than the other, and computational effort could be reduced by first checking with the more discriminating of the two filters. The choice of which filtering operation to compute first will be an active area of study as more field data is collected with the NILM.

The transversal filter is flexible and relatively computationally undemanding. It will prove in Chapter 5 to be remarkably effective at distinguishing v-sections in practice. There are, however, an enormous number of more sophisticated schemes for the numerical discrimination of patterns described by a vector of input samples, offering a variety of potential benefits ([91]). Discriminators can be constructed that will check the shape and amplitude of input data simultaneously, eliminating the need for two separate transversal filtering operations. Consider, for example, the quadratic-linear filter, which computes a discriminant d as

$$d(x_{zm}) = x_{zm}^T W x_{zm} + x_{zm}^T u + c$$

where W, u, and c are, respectively, a matrix, a vector, and a scalar constant that define a pattern match space. The superscript T denotes transposition. The variable x_{zm} represents an N-point, zero mean but *not* amplitude normalized vector computed from an N-point vector x of raw samples $x[i]$ extracted from an input data stream. The vector x_{zm} is

calculated according to the formula

$$x_{zm} = x - \frac{1}{N}\sum_{i=0}^{N-1} x[i].$$

By carefully selecting W, u, and c, it is possible to define a closed region in the pattern space such that, for example, the discriminant value d is positive when the tip of the vector x_{zm} falls within the closed region, and negative when it falls outside the closed region. Because the match space may be closed and small, this filter will not be susceptible to a false positive from an anemic, low amplitude vector with a fortuitously correct shape as was the case for the shape transversal filter, or from a large amplitude vector with an incorrect shape, as was the case for the amplitude transversal filter. The quadratic-linear filter has been applied to pattern recognition problems in [35], [141], and more recently in [138].

The quadratic-linear filter can be interpreted as a special case of a class of generalized polynomial discriminators. Typical polynomial discriminators operate essentially as transversal filters with the possible addition of a scalar constant to the output. However, the input vector for the polynomial discriminator is augmented with higher powers and cross-products of the individual samples of a vector x_{zm}. So, a generalized polynomial discriminator which worked with all of the N entries of a vector x_{zm} and the squares of all of the entries would operate on an input vector with $2N$ samples. As higher powers and cross-products of the initial input vector are added, progressively more complicated geometric shapes can be developed as match regions in the pattern space. A generalized nonlinear discriminator might also work with arbitrary nonlinear functions of the initial input points. Training and implementation costs may become steep as nonlinearities are added into the discriminator function. Some useful training and "pruning" algorithms that help determine which nonlinearities selected from a restricted collection of functions will best discriminate a set of vector patterns may be found in [8], [9], [10], [108], and [151].

3.4 Summary

The practical success of most pattern recognition systems depends strongly on the assumptions made in formulating the pattern discrimination techniques. The NILM is no exception. Before tackling multiscale search techniques in the next chapter, this section briefly reviews

the assumptions inherent in the transient identification scheme described thus far.

To differentiate loads based on transient behavior, the NILM will require that the different loads of interest exhibit relatively unique and repeatable turn–on transient profiles. The load survey in the previous chapter suggests that this is a reasonable requirement. To monitor accurately even in the face of high rates of event generation, the NILM will search for only the v-sections, i.e., the most informative, varying segments, of a transient profile. For the NILM to declare a match with a known transient, all of the v-sections for that transient must appear in the input data stream in the right time pattern and with the correct amplitudes.

Some v-sections may not appear with precisely the exact same shape and amplitude time after time. If the transients associated with the loads of interest yield time patterns of v-sections that are complicated, i.e., many different v-sections must appear before the entire transient will be presumed present in the input streams, it may be possible to relax the match tolerances considerably and permit an occasional false match in one or two v-sections. This is true because the entire, complex time pattern of v-sections must be found before a transient event will be assumed to have occurred. To minimize the necessary match tolerance, a template for a difficult v-section could be generated by first clustering together many examples of the v-section, and then picking the cluster centroid or mean as the representative template. Also, multiple pattern discriminators, or a single nonlinear discriminator, could be employed to narrow the acceptable match space in some directions and expand it in others to help avoid conflicting identifications.

The prototype event detector implemented for this thesis will require that no v-sections overlap if all transients are to be identified without error. A hierarchical search scheme will therefore be employed to enhance the robustness of the NILM. The NILM will search for transients with the largest number of v-sections first. If a match is found for the complete v-section set for a very complicated transient pattern at some time scale, the NILM will assume that this transient pattern really is present in the input data. Subsequently, any less complicated patterns will not be permitted to "match" based on a v-section already associated with a more complicated pattern.

If all of the patterns are reasonably complicated, and especially if the v-sections appear with consistent and significantly different shapes, an explicit lock out step in the

algorithm will probably not be necessary. This reiterates a fundamental trade-off. When the individual v-sections are different in shape, and appear identically time after time, the match region tolerances can be made very small. In this case, if a v-section is detected, the identification is likely to be a reliable one. On the other hand, suppose that the match region tolerances must be relatively large because, for example, a v-section does not exhibit precisely the same shape time after time. In this case, individual v-section identifications are less reliable, and more emphasis must be placed on finding precisely the appropriate set or time pattern of v-sections associated with a complete transient pattern. If the match region tolerances must be large and at the same time there are few v-sections in one or more of the transient patterns of interest, the robustness of the event detection may be compromised. In this case, the v-section lock out attempts to enhance reliability by eliminating details that are known to be associated with a complicated event that has been detected with a high degree of surety.

The requirement that no v-sections overlap if transients are to be reliably recognized determines, in large part, the extent to which the NILM can really be nonintrusive. If the rate of event generation at the service entry of a building is so high that v-sections almost always overlap significantly, the event detection algorithm outlined in this chapter will perform poorly. In this case, the NILM will have to be installed deeper inside the building's internal wiring distribution network for robust operation. Several internal points might have to be monitored to determine the complete electrical usage profile for the entire building. Intuitively, it is likely that small to moderate size buildings might be monitored completely from the service entry, while large buildings may require monitoring at a few points on the internal wiring harness. Precise notions of small, medium, and large, and other factors that affect the observed rate of event generation within a building, will become apparent through further field studies with the NILM.

It is possible that the v-section lock out could be relaxed in an attempt to further increase the allowable rate of event generation. For example, if it were possible to guarantee that all of the possible transient patterns observable at the service entry were repeatable and known to the NILM, then any unknown v-section spotted, for instance, by an edge detector, would have to be a combination of known v-sections. The NILM could then attempt to analyze the unknown v-section as different combinations of known v-sections in order to

find matches. One risk with such a scheme would be that a new, previously unknown load might be installed in a building. It could present a v-section that fortuitously matched a combination of known v-sections, potentially leading to false identifications. The possible utility of such an approach will have to be evaluated in the light of further field studies.

Chapter 4

Searching for Patterns Over Many Time Scales

We have seen in Chapter 2 that many common load classes exhibit transient behaviors that are both relatively unique and also potentially present over a range of time scales. This diversity of load transient shapes supports the notion that transient shapes observed by a NILM could be accurately attributed to particular load classes. Multiscale behavior appeared not only within a class of devices over a range of load sizes, but also when comparing loads from different classes. Chapter 3 considered numerical pattern recognition techniques suitable for identifying transient shapes over a narrowly defined time scale. This chapter develops an efficient and systematic approach for searching over many time scales with these pattern recognition tools.

4.1 Background

Classical methods for harmonic or spectral analysis do not generally approach the localization bound implied by the uncertainty principle ([53], [128], [117]). The traditional Fourier transform, for example, localizes well in frequency but poorly in time. A Fourier coefficient indicates the quantity or presence of a single frequency in input data. This indicates the presence of a sine wave or complex exponential in time that is global in duration, or totally unlocalized. In other words, a Fourier transform represents an input signal as a weighted sum of basis functions – sine waves – that endure for the entire extent of the input data space.

Many applications are better analyzed by a transformation that is more localized in time. One familiar localized spectral analysis is a sonogram ([100]). A sonogram is a two dimensional plot of "frequency" versus time for an input such as a speech signal. In speech recognition or modification problems, a global frequency content estimate is not particularly useful because speech is naturally divided into subunits that typically must be recognized individually before word or statement recognition or analysis is performed. A sonogram attempts to provide some information about the local frequency content in a particular region of the input signal. A windowed or *short time Fourier transform* (STFT) is used to generate the sonogram ([100], [12], [38]). The STFT projects the input signal onto a space spanned by basis functions that are windowed or gradually tapered sine waves. These essentially local basis functions correspond to essentially local packets of frequency content. Time *scale* may be adapted versus frequency content or *resolution* by altering the extent or *support* of the analysis window. A presentation similar in some respects to the sonogram may be found on another familiar local spectral representation, a musical score, which presents notes or frequencies to be played in time ([117]).

The STFT is one example of an intermediate transformation that provides a time varying representation of frequency content. Such a transformation is sometimes called a time-frequency representation (TFR). Surveys of TFR techniques may be found in [61] and [117].

The need for a systematic method for *multiresolution* analysis has surfaced in many scientific and engineering disciplines. A common thread found in most applications is a desire to adaptively trade scale or localization in time versus resolution or localization in frequency. Sophisticated TFR methods introduced in the last 5 to 10 years for the analysis of time-varying or non-stationary signals attempt to provide an algorithmic framework for adapting time and frequency localization in the process of analyzing input data. In particular, the study of multiresolution transforms has been reinvigorated recently by the introduction of two related analysis techniques: the discrete wavelet transform (DWT), introduced in [89], [84], [85], and [32]; and quadrature mirror filter sub-band coding ([143], [144]).

The DWT analyzes a window of input samples by describing the data as a sum of basis functions related by a dilation equation. Each function in the basis for the input

space is a shifted and/or scaled (upsampled) version of a single prototype function derived from a *wavelet*. The analysis performed by the DWT may be viewed intuitively as passing the input data through a bank of band-pass filters, where the pass-band of each filter is proportional to its center frequency. At higher analysis frequencies, the larger filter bandwidths permit greater localization in time. For example, for signals that consist of relatively short, high frequency transients embedded in long, low frequency components, the utility of such a transform is immediate: transients may be distinguished in the transformed space by examining coefficients corresponding to high analysis frequencies that localize well in time. Some applications to transient detection are presented in [39] for the DWT and in [52], [51] and [40] for the continuous-time wavelet transform. Related applications of TFR methods to transient detection may also be found in [13], [60] and [115].

Considerable attention has been focused recently on applying the DWT algorithm using compactly supported, orthogonal bases. In [32], a general class of such bases is introduced, which complement the analysis and synthesis equations presented in [84]. A fast, computationally efficient algorithm for computing the DWT based on orthogonal basis functions, sometimes called a fast wavelet transform (FWT), was introduced in [84]. Almost simultaneous with the advent of the FWT was the introduction in the signal processing community of perfect reconstruction (PR), or quadrature mirror filter, sub-band coding schemes ([131], [142], [144], [145], [45]). Quadrature mirror filter (QMF) banks may be interpreted as atomic components of the logarithmic filter bank implemented in effect by the DWT. The precise mechanics of the FWT will be further illuminated where appropriate in later sections. Good surveys may be found in [135], [117], [121], and [45].

4.1.1 Current Approach

The approach adopted here for searching over many time scales for load transients is inspired by the FWT. For a continuous-time signal, the time scale of a signal may be changed without losing the resolution of the signal. This is apparent from the scaling property of the continuous-time Fourier transform ([128], [101]). Expanding or compressing a signal in continuous-time has the effect of compressing or expanding the transform of the signal, respectively. The scale change can always be reversed to recover the original signal with no loss in resolution. The discrete-time case, of primary concern for any practical implemen-

tation of a NILM on digital hardware, is somewhat more complicated. Decreasing the scale of a discrete-time signal is performed by upsampling, which has the impact of introducing imaging artifacts into the frequency transform of the upsampled signal ([144]). Increasing the scale, on the other hand, is performed by downsampling, which automatically changes resolution by lowering the maximum resolvable frequency, potentially corrupting the signal with aliasing distortion ([145], [117]). The process of changing the scale of a discrete-time signal is apparently complicated by the need to present only the information or resolution that should and can be displayed at the chosen scale.

The resolution of a discrete-time signal is typically adjusted by low-pass filtering before or after downsampling or upsampling. Convolution with a low-pass filter may be interpreted as projecting the input data into a "low-pass space" spanned by a basis that consists of shifted versions of the filter's impulse response. Repeated steps of filtering and downsampling, for example, are directly analogous to successive analysis steps in the wavelet transform, or to cascades of sub-band coders.

The development in this chapter begins with the heuristic assumption that transients or v-sections of interest to the NILM at a particular time scale are well represented by a fraction of the frequency content of the input data at a lower, more finely sampled scale. More precisely, we assume that, given two v-sections that are identical except in scale, the v-section at the higher scale can be made identical to the v-section at the lower scale by filtering and downsampling the v-section at the higher scale.

In general, this assumption is not strictly correct. However, experience with the v-sections commonly found in load transients of interest indicates that their frequency content tapers rapidly. The error introduced by truncating the frequency content through filtering and downsampling therefore does not appear intractable in the pattern recognition setting. Later sections in this chapter will consider possible approaches for situations where this assumption does not appear to be a good one.

Following a brief review of notation and basic mathematical tools, a tree-structured decomposition appropriate for pattern recognition systems searching over many time scales is introduced. Compactly supported basis functions, or FIR filters, are shown to be essential for preserving the shape of v-sections and for developing a scheme for circumventing problems associated with the shift variance introduced by downsampling. Since reconstruction

is not a primary concern, orthogonality will be sacrificed to preserve linear phase characteristics in the FIR filters. In this respect, the search schemes developed in this chapter are reminiscent of precursors to the FWT and QMF coding, such as the Laplacian pyramid image coding technique presented in [17] and near-perfect reconstruction filter banks like those presented in [29], [143].

4.1.2 Notation

This section establishes notation and reviews several signal processing constructs fundamental to the concepts developed here. The coverage is not complete, as there are many references cited in this section that provide more details. The initiated reader may wish to skip to the next section.

The z-transform $H(z)$ of a discrete-time sequence $h[k]$ is defined to be:

$$H(z) = \sum_{k=-\infty}^{\infty} h[k]z^{-k} \tag{4.1}$$

The variable z is complex. If the region of convergence of the z-transform of a sequence includes the unit circle in the complex z plane, the discrete-time Fourier transform (DTFT) of the sequence exists and is obtained by setting $z = e^{j\omega}$ in the z-transform. Throughout this chapter, all z-transforms will be presumed to be taken on finite length sequences unless stated otherwise, so the DTFT of a sequence will always exist. When discussing the frequency content of a signal, the convention adopted here will be to refer to the value of the DTFT of the signal on the fundamental region $-\pi < \omega \leq \pi$, with the understanding that the DTFT of a sequence is periodic in ω. Useful properties of the z-transform and DTFT as well as transforms for some important functions may be found in [65], [102], and [128].

For any sequence $h[k]$, it is possible to identify an even sequence $h_e[n] = h[2n]$ that consists of all of the even indexed samples of h. The odd sequence $h_o[n] = h[2n+1]$ consists of all of the odd indexed samples. The z-transform of $h[k]$ may thus be expressed as

$$H(z) = H_e(z^2) + z^{-1}H_o(z^2)$$

where $H_e(z)$ and $H_o(z)$ are the z transforms of the even and odd component sequences. This even and odd splitting generalizes to an M-band *polyphase decomposition*, which for

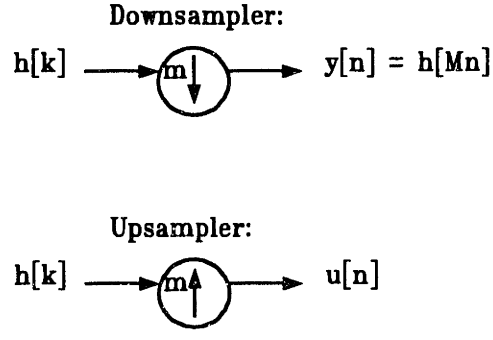


Figure 4.1: Discrete-Time Scale Alteration

any z-transform may be written as:

$$H(z) = \sum_{i=0}^{M-1} z^{-i} H_i(z^M)$$

where $H_i(z)$ corresponds to the i-th polyphase component of $H(z)$, or the transform of the time domain sequence $h_i[n] = h[Mn + i]$. The polyphase decomposition, a basic tool in analyzing many multirate signal processing applications [29], will be essential in the developments presented in later sections.

The scale of a sequence $h[k]$ may be increased by downsampling or decreased by upsampling. The M-fold downsampler, shown in Fig. 4.1, is a linear, time-varying system that extracts the sequence $y[n] = h[Mn]$ from an input sequence $h[k]$, where M is an integer. The DTFT of the output of the M-fold downsampler is

$$Y(e^{j\omega}) = \frac{1}{M} \sum_{k=0}^{M-1} H(e^{j(\frac{\omega}{M} - \frac{2\pi k}{M})}). \tag{4.2}$$

Compressing a sequence in time has the effect of expanding or stretching the transform of the sequence in frequency. If the transform of the input sequence has frequency content outside of the region $-\frac{\pi}{M} < \omega < \frac{\pi}{M}$, *aliasing* will distort the output signal ([128], [144]). A low-pass filtering operation typically precedes downsampling to suitably bandlimit the input signal.

The M-fold upsampler, also shown in Fig 4.1, is also a linear time-varying system. It produces as output a sequence with $M-1$ zero valued samples in between each input sample point. The transform of the output of the upsampler is

$$U(e^{j\omega}) = H(e^{j\omega M}). \tag{4.3}$$

The upsampler expands the input signal in time, compressing the transform of the sequence in frequency. Unlike the downsampler, the upsampler compresses without destroying the shape or content of the original spectrum. This is consistent with the fact that no input samples are discarded. *Imaging* artifacts will be present in the output spectrum, however ([144]). Upsampling is typically followed by a low-pass filtering operation that interpolates between the input sample points, "filling in" the zero samples and minimizing the effects of imaging.

Finally, all norms employed in this chapter are 2-norms unless stated otherwise.

4.2 Tree-Structured Decomposition

Downsampling and upsampling will play key roles in the development of a signal processing construct for multiresolution pattern recognition. This section presents an appropriate structure for efficiently organizing the pattern search. Requirements specific to pattern recognition problems will be outlined here and addressed in the remaining sections of this chapter. Commonalities and differences with respect to other methods extant in the literature, including the FWT and QMF coding, will be highlighted.

Consider the search structure shown in Fig. 4.2. An atomic section of the tree shown would consist of the two upsamplers with upsampling rates of l and b, operations $\mathbf{R}$ and $\mathbf{D}$, and the two trailing downsamplers with downsampling rates of m and c. The upsampling and downsampling operations in one section of the tree can alter the scale of the input signal – for example X_{s+1} at the resolution step $s+1$ – by fractional amounts $\frac{m}{l}$ and $\frac{c}{b}$. The scale alteration produces the output signals X_s and U_s, respectively. For the moment, $\mathbf{R}$ and $\mathbf{D}$ are unspecified operations. If $\mathbf{R}$ and $\mathbf{D}$ were implemented as linear filters and b,c,l, and m were adjusted to yield at output resolution step s a net increase in scale (a decrease in the number of points, i.e. $\frac{m}{l} > 1$ and $\frac{c}{b} > 1$), the atomic section would correspond to a single sub-band coder, generalized by the inclusion of arbitrary rational sample rate

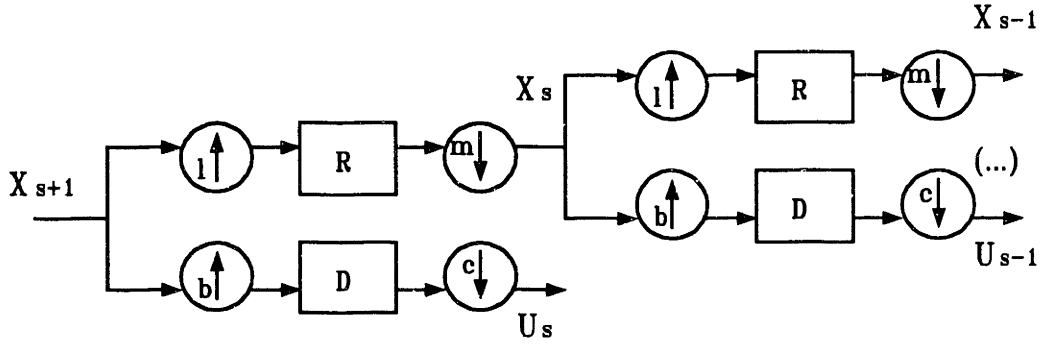


Figure 4.2: Tree-Structured Signal Decomposition

alteration ([29]). With linear filters and $b = l = 1$ and $c = m = 2$, the tree-structured organization of sub-band coders shown in Fig. 4.2 is referred to in the wavelet literature as Mallat's pyramidal or tree-structured wavelet decomposition, or the analysis component of the DWT ([32], [84], [134], [135]).

The data at resolution step $s+1$ at the first stage of the tree is presumably the original, "raw" sampled data collected for a particular application. That is, the data stream X_{s+1} is collected at a high enough sampling rate to adequately capture all of the frequency content present in the input data without aliasing. This lowest scale data contains information or signal detail at many different scales. Each upper path in any particular sub-band coder in the tree alters the resolution of the input signal with operation **R** and also alters the scale of the input signal with the up/downsampling operations at rates l and m. This path will be referred to as the *resolving* path. The lower, *discriminating* path typically implements a differencing or discriminating operation **D** that extracts scale specific information from the input data. In the case of a wavelet transform of an initial window of data X_{s+1} containing 2^M input points and with $b = l = 1$ and $c = m = 2$, the tree-structured decomposition will produce output coefficients U_i at M different scales or tree sections. In the wavelet transform, the operation **R** is typically a low-pass filter that extracts a low resolution signal X_s from the higher resolution input signal X_{s+1}. The operation **D** is a high-pass filter that

creates a "detail" signal that encodes the information or resolution that must be combined with X_s to recover the input data.

This tree-structured decomposition is an intriguing and potentially highly efficient means for organizing the examination of scale-dependent information. In the case of the wavelet transform, the number of input points is halved for each successive coder in the tree. Generally, the number of mathematical operations required to implement **R** and **D** is small and, for each output point, independent of the total number of points $N = 2^M$ in a window of input data X_{s+1}. Suppose that the total number of mathematical operations needed by a single coder to generate each output point is C. In this case, the first sub-band coder requires CN operations to generate X_s from X_{s+1}. The next coder requires $\frac{CN}{2}$ operations to create X_{s-1} from X_s. The total number of operations required to generate a complete transform over M scales is

$$CN + \frac{CN}{2} + \frac{CN}{4} + \frac{CN}{8} \ldots < 2CN$$

so, regardless of the size of the input data window, the operations count for the wavelet transform is essentially $O(N)$.

In a pattern recognition application, operation **D** will be a pattern discriminator that detects the presence of a particular waveform shape, or v-section in the case of the NILM. As in the DWT, **D** may be a linear operation such as a transversal matched filter, or it might be a nonlinear detector, as discussed in Chapter 3. In general, the tree-structured search for pattern recognition applications should probably be conducted with $b = c = 1$. Sample rate changes in the discrimination branch of each sub-band coder will affect only the constant term and not the order in the overall operations count for the tree. The decrease in operations which would result from downsampling in the discrimination branch would probably not outweigh the disadvantages of introducing time-variance in the discrimination, i.e., introducing shift-variant uncertainty in the shape of the v-sections in the input data stream.

The low operations count is a powerful inducement, however, to work with net scale increases of 2 to 1 in each top branch coder stage in the tree-structured decomposition. The next three sections will concentrate on this case, on the assumption that the range of observed variation in v-section shapes on any particular scale is negligible or can be dealt

with by a sufficiently flexible pattern discriminator. If this assumption is not applicable, smaller increases in scale from coder to successive coder will be required. Fractional scale increase will be addressed in Section 4.6. If the previously stated assumption that v-sections at one scale are reasonably well represented by a fraction of the frequency content at the previous scale is valid, one fairly obvious choice for pattern recognition systems like the NILM is to implement operation **R** with a low-pass filter. Then, **R** can be crafted to preserve the resolution or frequency content appropriate for the next scale level in the tree while rejecting any frequency content that would be aliased.

It also seems best to implement **R** with an operation that has a phase characteristic that is zero or, more practically, linear. A nonlinear phase characteristic will distort the envelope or shape of the input waveform ([65], [102], [103]). Compounded with the time-variance associated with sample rate alteration, a nonlinear phase characteristic drastically complicates the process of developing templates or prototypes for designing robust pattern discriminators. A finite impulse response (FIR) filter will therefore be required if **R** is to be a linear operator; possible nonlinear operators will be discussed later in this chapter.

The wavelet transform is generally implemented with a linear FIR filter for operation **R**. Note, however, that Daubechies has shown that all, except the lowest order (Box/Haar), shift orthogonal FIR filters will not be symmetric or antisymmetric and will not, therefore, have a linear phase characteristic ([32], [134]). This has greatly complicated the application of the FWT in pattern recognition applications ([135]).

In fact, even with linear phase, practical FIR filters will not implement an ideal low-pass magnitude spectrum that would ensure no aliasing or amplitude distortion. The time-variance introduced by sample rate alteration has nearly crippled the application of the DWT to pattern recognition problems. Many researchers have practically given up on performing direct pattern discrimination with wavelet coefficients and instead have dealt with the time-variance problem by using coarse characteristics of the transform coefficients to determine useful information about the input waveform. In [121], for example, the well known Canny edge detection algorithm ([19]) is shown to fit in a wavelet multiresolution framework. Local extrema in a wavelet transform based on filters with linear phase characteristics are used to find steps or step-like features in an input waveform. For a sophisticated pattern recognition application like the NILM, techniques like this edge detection scheme

would at best serve as preprocessors that identified regions meriting further attention in an input waveform.

Key results developed in the next three sections will demonstrate direct means for controlling time-variance effects. These methods will permit more sophisticated pattern discrimination techniques to operate effectively and directly as the **D** operation at each coder stage in the tree-structured decomposition. Given the requirements reviewed in this section, for the initial analysis of the tree-structured decomposition applied with scale increases of 2 to 1, **R** should be a half-band, low-pass FIR filter with a linear phase characteristic.

4.3 Ideal Resolution Alteration

In order to develop and evaluate the performance of appropriate and practical filters for operation **R**, it is necessary to first understand the operation of ideal filters in a sample rate alteration application. An ideal low-pass filter with positive cutoff frequency ω_c has a DTFT $\overline{L}(e^{j\omega})$, where

$$|\overline{L}(e^{j\omega})| = \begin{cases} 1 & \text{if } |\omega| < \omega_c \\ 0 & \text{otherwise.} \end{cases} \tag{4.4}$$

Possible phase characteristics for the ideal filter will be considered shortly. For an ideal half-band, low-pass filter with impulse response $\overline{h}[n]$ and DTFT $\overline{H}(e^{j\omega})$, the cutoff frequency $\omega_c = \frac{\pi}{2}$. An ideal half-band, high-pass filter with impulse response $\overline{g}[n]$ and DTFT $\overline{G}(e^{j\omega})$ may be constructed by shifting the transform of the ideal half-band, low-pass filter by π:

$$\overline{G}(e^{j\omega}) = \overline{H}(e^{j(\omega-\pi)}) \tag{4.5}$$

The two phase polyphase decomposition will prove invaluable for analyzing 2 to 1 decimation applications. The transform of the ideal half-band, low-pass filter may be expressed as

$$\overline{H}(z) = \overline{H}_e(z^2) + z^{-1}\overline{H}_o(z^2) \tag{4.6}$$

where $\overline{H}_e(z)$ is the transform of a sequence $\overline{h}_e[n] = \overline{h}[2n]$ and $\overline{H}_o(z)$ is the transform of $\overline{h}_o[n] = \overline{h}[2n+1]$. With the help of Eq. 4.2 it is possible to find explicit expressions for the even and odd polyphase components. Since $\overline{H}_e$ is the transform of the impulse response

$\overline{h}[n]$ *after* downsampling every other point,

$$\overline{H}_e(e^{j\omega}) = \frac{1}{2}\sum_{k=0}^{1} \overline{H}(e^{j(\frac{\omega}{2}-\pi k)}). \tag{4.7}$$

If $\overline{H}(e^{j\omega})$ additionally corresponds to the transform of a filter with zero phase, i.e., if the imaginary part of the transform is zero,

$$\overline{H}(e^{j\omega}) = \begin{cases} 1 & \text{if } |\omega| < \frac{\pi}{2} \\ 0 & \text{otherwise} \end{cases} \tag{4.8}$$

Substituting Eq. 4.8 into Eq. 4.7 reveals that $\overline{H}_e(e^{j\omega}) = \frac{1}{2}$.

Similarly, since $\overline{H}_o$ is the transform of the impulse response $\overline{h}[n]$ after *first* shifting the sequence by one point and *then* downsampling every other point, an expression for $\overline{H}_o$ may be found by applying Eq. 4.2 to the transform of the sequence $\overline{h}[n]$ shifted one point backward in time:

$$\overline{H}_o(e^{j\omega}) = \frac{1}{2}\sum_{k=0}^{1} e^{\frac{j\omega}{2}} e^{-j\pi k} \overline{H}(e^{j(\frac{\omega}{2}-\pi k)}) \tag{4.9}$$

Still assuming that the ideal half-band, low-pass filter has zero phase, substituting Eq. 4.8 into Eq. 4.9 yields $\overline{H}_o(e^{j\omega}) = \frac{1}{2}e^{\frac{j\omega}{2}}$.

The polyphase components of the complementary high-pass filter follow immediately. The two phase description of $\overline{G}(z)$ is

$$\overline{G}(z) = \overline{G}_e(z^2) + z^{-1}\overline{G}_o(z^2) \tag{4.10}$$

Substituting Eq. 4.5 into the polyphase description of $\overline{H}(z)$, Eq. 4.6, will expose specific expressions for the polyphase components $\overline{G}_e(z)$ and $\overline{G}_o(z)$:

$$\overline{G}(e^{j\omega}) = \overline{H}(e^{j(\omega-\pi)}) = \overline{H}_e(e^{j2\omega}) + e^{-j(\omega-\pi)}\overline{H}_o(e^{j2\omega}) = \overline{H}_e(e^{j2\omega}) - e^{-j\omega}\overline{H}_o(e^{j2\omega}) \tag{4.11}$$

In z-transform terms,

$$\overline{G}(z) = \overline{H}_e(z^2) - z^{-1}\overline{H}_o(z^2). \tag{4.12}$$

Comparing this expression with Eq. 4.10 reveals that $\overline{G}_e(e^{j\omega}) = \overline{H}_e(e^{j\omega}) = \frac{1}{2}$ and $\overline{G}_o(e^{j\omega}) = -\overline{H}_o(e^{j\omega}) = -\frac{1}{2}e^{\frac{j\omega}{2}}$.

Because the frequency response of the ideal half-band, low-pass filter is obviously bandlimited, the impulse response of the filter is infinite in duration. Practical half-band,

low-pass FIR filters are constructed by a variety of methods that attempt to approximate the ideal impulse response with a finite response as closely as possible according to some criterion. FIR filters designed to match the frequency response of Eq. 4.8 will necessarily have an odd number of taps or impulse response samples, and are sometimes called half-band Type I FIR filters. One way to see the necessity for an odd number of taps is to examine $\overline{H}_e(z)$ and $\overline{H}_o(z)$. The transform $\overline{H}_e(z)$ is constant for all frequencies and corresponds to a discrete-time impulse at the origin in time. Since $\overline{H}(z)$ has zero phase, the odd indexed samples of $\overline{h}[n]$ must be symmetric, i.e., $\overline{h}[-1] = \overline{h}[1], \overline{h}[-3] = \overline{h}[3]$, etc. So, to maintain a zero or at least linear phase characteristic, FIR filter taps approximating $\overline{h}[n]$ must contain matched pairs of sample points in the odd indexed locations, plus one non-zero sample at a single even indexed location. So, the Type I FIR filter will always have an odd number of symmetric taps. Equation 4.12 indicates that the complementary high-pass filters will also be symmetric and therefore will have a linear phase characteristic.

An ideal half-band, low-pass filter with a linear, but always non-zero, phase characteristic may also be constructed by multiplying $\overline{H}_e(e^{j\omega})$ and $\overline{H}_o(e^{j\omega})$ by $e^{\frac{-j\omega}{4}}$. This new filter with transform $\overline{Hs}(z)$ will have the same magnitude response as the filter in Eq. 4.8, but with $\overline{Hs}_e(e^{j\omega}) = \frac{1}{2}e^{\frac{-j\omega}{4}}$ and $\overline{Hs}_o(e^{j\omega}) = \frac{1}{2}e^{\frac{j\omega}{4}}$. These transforms are conjugates and correspond, therefore, to sequences that are mirror images of each other. Linear phase is achieved by matching pairs. Each point in the even indexed sequence pairs with a single, mirror image point in the odd indexed sequence of the filter impulse response. FIR filters developed from this ideal half-band filter will necessarily have an even number of taps and are called half-band Type II filters. Complementary high-pass filters may still be constructed for both the ideal and FIR cases by shifting the transform by π as in Eq. 4.5, resulting in an antisymmetric, high-pass impulse response.

The phase difference between $\overline{H}(z)$ and $\overline{Hs}(z)$ is not a great concern in the analysis of sample rate alteration with *ideal* filters; however, the performance of Type I FIR filters may be quite different from that of Type II filters, as will be seen in the next section.

As an aid to understanding the differences between Type I and Type II filters, consider the result of filtering a *half-band limited* signal $x[n]$ with the high-pass filter $\overline{g}[n]$. The z-

transform of $x[n]$, $X(z)$, may again be expressed in polyphase form:

$$X(z) = X_e(z^2) + z^{-1}X_o(z^2)$$

Filtering the signal with the high-pass filter is equivalent to multiplying the transforms of the signal and the filter,

$$X(z)\overline{G}(z) = [X_e(z^2) + z^{-1}X_o(z^2)][\overline{G}_e(z^2) + z^{-1}\overline{G}_o(z^2)] \tag{4.13}$$

Substituting Eq. 4.12 for the polyphase description of the transform $\overline{G}$,

$$X(z)\overline{G}(z) = [X_e(z^2) + z^{-1}X_o(z^2)][\overline{H}_e(z^2) - z^{-1}\overline{H}_o(z^2)].$$

Expanding the product of the polyphase expressions yields

$$X(z)\overline{G}(z) = X_e(z^2)\overline{H}_e(z^2)+z^{-1}[X_o(z^2)\overline{H}_e(z^2)-X_e(z^2)\overline{H}_o(z^2)]-z^{-2}X_o(z^2)\overline{H}_o(z^2). \tag{4.14}$$

By definition, the signal x has no frequency content in the upper half-band, so

$$X(z)\overline{G}(z) = 0. \tag{4.15}$$

This indicates that the entire right-hand side of Eq. 4.14 is zero. The trivial product of $X(z)$ and $\overline{G}(z)$, or equivalently the convolution of the signal $x[n]$ and the impulse response $\overline{g}[n]$, could be downsampled and the result would still be zero. This is equivalent to applying Eq. 4.2 to the product of the transforms, $X(z)\overline{G}(z)$, or identically, to the right-hand side of of Eq. 4.14. Considering Eq. 4.15 after substituting Eq. 4.14 into Eq. 4.2 reveals that

$$X_e(z)\overline{H}_e(z) - z^{-1}X_o(z)\overline{H}_o(z) = 0.$$

Recalling the explicit expressions derived above for the polyphase filter components,

$$X_e(e^{j\omega})\frac{1}{2} - e^{-j\omega}X_o(e^{j\omega})\frac{1}{2}e^{\frac{j\omega}{2}} = 0$$

or simply

$$X_e(e^{j\omega}) = e^{-\frac{j\omega}{2}}X_o(e^{j\omega}). \tag{4.16}$$

Equation 4.16 reveals two interesting facts about the polyphase components of any signal $x[n]$ that is half-band limited. The first concerns the magnitudes of the polyphase components:

$$|X_e(e^{j\omega})| = |X_o(e^{j\omega})| \tag{4.17}$$

The second concerns the relative phase of the polyphase components:

$$\arg(X_e(e^{j\omega})) = \arg(X_o(e^{j\omega})) - \frac{\omega}{2} \tag{4.18}$$

These two characteristics are independent of the phase of the *ideal* half-band linear phase filter used for the derivation. That is, high-pass filters derived from either $\overline{H}$ or from $\overline{Hs}$ would yield the same results. A signal that is *not* half-band limited may contain high frequency content by violating the first condition, or the second condition, or both. These different ways in which a signal may have high frequency content will affect which type of FIR filter, Type I or Type II, is the appropriate choice for the tree-structured decomposition in a pattern recognition setting.

4.4 Practical Resolution Alteration

A practical FIR filter, with its compactly supported impulse response, can not implement a perfect "brick wall" low-pass frequency response. After convolving an input signal with an FIR filter and then downsampling, three types of error or distortion may be present in the output signal in comparison to the output following ideal filtering ([143]). *Phase distortion* will be eliminated by insisting on Type I or Type II FIR filters with linear phase characteristics. *Aliasing distortion* may be minimized by selecting a filter with sufficient stop-band attenuation to prevent excessive aliasing following downsampling. *Amplitude distortion* will occur because any practical filter frequency response will have a finite transition region between the pass and stop bands, and may also have ripple in the pass-band, stop-band, or both. This section will examine the effect of amplitude distortion in the tree-structured search scheme. [1]

There are two strong motivations for picking low order filters which have as few taps as possible. The first is that computational effort increases as the filter grows. The second,

[1] A fourth type of distortion, *quantization error*, arises from finite bit-rate coding and will occur in any practical digital implementation. We will see that there are strong practical reasons in the NILM for using FIR filters with the smallest number of taps possible so that aliasing and amplitude distortion are brought to just tolerable or controllable levels. Profligate use of processing resources, in other words, would not necessarily be helpful in eliminating the effects of amplitude distortion and time variance. The extent and impact of quantization error, on the other hand, is partly a function of "dollars spent" on hardware, and will not be a focus here. Experience seems to indicate, however, that quantization errors may be minimized by working with filters with linear phase characteristics ([143]).

and perhaps more significant concern in a pattern recognition setting, is the desire to project input data into a low-pass space spanned by basis functions (shifted versions of the filter's impulse response) that are as localized or concentrated in time as possible. Why? In many pattern recognition or transient identification problems the v-sections or target transients are step-like, especially at relatively large scales. As shown by the survey in Chapter 2, this situation is clearly true for the target environment of the NILM. Convolving an FIR filter with a step-like input signal gives an output similar to the step response of the filter ([128], [53]). Non-monotonic step responses are characteristic of FIR filters designed by window methods, and also by the optimal *minimax* method described in the next section, which will help develop filters with the best possible localization in frequency – important for minimizing aliasing distortion. If two step-like v-sections are adjacent in the input data, and the support of the FIR impulse response is wide, the step responses from filtering the two v-sections may overlap severly and interfere with the pattern recognition. To maximize the acceptable rate of event generation, it is therefore desirable to use the lowest order FIR filters consistent with tolerable stop-band attenuation.

An FIR filter with a narrowly supported impulse response will have a frequency response with a relatively wide transition region – the uncertainty principle again. A wide transition region exacerbates the amplitude distortion in the output signal by attenuating important frequency content in the tail end of the input spectrum fraction that is to be passed to the next resolution step in the decompositon tree. The amplitude distortion may be thought of as being caused by imperfect modeling of the ideal polyphase filter components in the actual polyphase components of the real filter. This error may be distributed "unevenly" between the polyphase components of the filter. A trivial shift of the input data in time before filtering and downsampling may therefore make a significant difference in the quality of the output waveform.

From an implementation standpoint, it is desirable to have a technique for selecting the input shift which leads to an output that best preserves, with respect to the ideal output, the information content of the input signal. For resolving paths performing 2 to 1 sample rate alteration and operated with either Type I or Type II low-pass FIR filters, this section will demonstrate that the best choice under certain criteria is the one which leads to an output waveform with the largest possible 2-norm. It will also be demonstrated

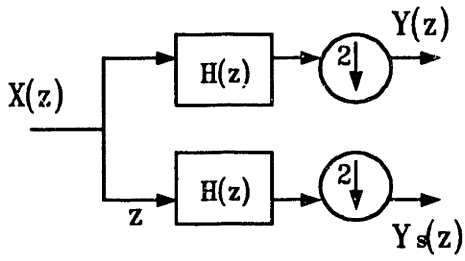


Figure 4.3: Possible Resolving Path Implementations

that differences in output fidelity for resolving paths operated with Type I or Type II filters are related to differences in the input signal polyphase components' magnitude or phase characteristics, respectively. These results will be used to develop an adaptive resolving path selection technique that helps to ensure the highest fidelity output for pattern recognition at each coder stage in the tree-structured decomposition.

4.4.1 Resolving with Type I FIR Filters

Two possible choices for the resolving path in one of the coders of the tree-structured decomposition are shown in Fig. 4.3. The upsampler in the resolving path has been dropped for simplicity since the scale change is 2 to 1 in this example. A best choice of resolving path may be made by analyzing both paths operated with either Type I or Type II FIR filters. First consider the action of a Type I, or odd tap, FIR filter with frequency response $H(z)$. This filter has a polyphase decomposition

$$H(z) = H_e(z^2) + z^{-1}H_o(z^2).$$

The ideal, even polyphase component $\overline{H}_e(z) = \frac{1}{2}$ corresponds to a single non-zero sample value in the even locations of the filter impulse response. There is no problem implementing this ideal polyphase component with a practical polyphase component, so $H_e(z) = \frac{1}{2}$. The ideal odd component is a problem, however, since it corresponds to an infinitely long sequence in time. When a practical Type I filter is designed, as will be demonstrated in the

next section, it is no problem to maintain the *phase* characteristic of $\overline{H}_o(z)$ by preserving the symmetry relationship between the positive and negative odd samples, but it is impossible to implement the exact magnitude response without an infinite number of samples. So, a practical odd polyphase component is $H_o(z) = e^{\frac{j\omega}{2}}|H_o(z)|$ where the function $|H_o(z)|$ will approximate $\frac{1}{2}$ as closely as possible.

Using the familiar two phase polyphase description of the input signal with transform $X(z)$, the output of the top path shown in Fig. 4.3 may be expressed as in the previous section as

$$Y(z) = H_e(z)X_e(z) + z^{-1}H_o(z)X_o(z). \tag{4.19}$$

Substituting in the expressions for the practical Type I polyphase components,

$$Y(z) = \frac{1}{2}X_e(z) + z^{-1}e^{\frac{j\omega}{2}}|H_o(z)|X_o(z) \tag{4.20}$$

where, for convenience and with the reader's indulgence, z-transform and DTFT notation are mixed. If the top path is operated with an ideal filter, the ideal output $\overline{Y}(z)$ can be determined by replacing the filter components with the ideal polyphase components:

$$\overline{Y}(z) = \frac{1}{2}X_e(z) + z^{-1}\frac{1}{2}e^{\frac{j\omega}{2}}X_o(z) \tag{4.21}$$

The error introduced by the practical filter may be quantified by subtracting Eq. 4.20 from Eq. 4.21,

$$\overline{Y}(z) - Y(z) = e^{\frac{-j\omega}{2}}[\frac{1}{2} - |H_o(z)|]X_o(z). \tag{4.22}$$

In the bottom path of Fig. 4.3, the input is first shifted backward one point by the delay z, then filtered and downsampled. The output of this branch operated with a practical Type I filter is

$$Y_s(z) = \frac{1}{2}X_o(z) + e^{\frac{j\omega}{2}}|H_o(z)|X_e(z). \tag{4.23}$$

The output of this branch operated with an ideal low-pass filter is

$$\overline{Y_s}(z) = \frac{1}{2}X_o(z) + \frac{1}{2}e^{\frac{j\omega}{2}}X_e(z). \tag{4.24}$$

The error in this branch with a practical filter may be expressed as the difference of Eqs. 4.23 and 4.24,

$$\overline{Y_s}(z) - Y_s(z) = e^{\frac{j\omega}{2}}[\frac{1}{2} - |H_o(z)|]X_e(z). \tag{4.25}$$

Equations 4.22 and 4.25 already suggest the origin of the time-variance dilemma, that is, the reason why one of the outputs Y or Y_s might be better than the other. In short, the even polyphase component of the analysis filter, H_e, can be implemented error free. The odd component H_o, on the other hand, will always have an amplitude error due to truncation. To expose the difference between the two choices in Fig. 4.3 more clearly, suppose that the magnitude of the odd polyphase component can be expressed as $|H_o(z)| = \frac{1}{2} - \delta(z)$, where $0 \leq \delta(z) \leq \frac{1}{2}$. This is equivalent to assuming that the overshoot in the pass-band ripple is negligible, a reasonable assumption because the filter, as will be shown later in this chapter, will be designed to minimize amplitude distortion in regions of significant input signal frequency content in the pass-band. The main remaining source of amplitude distortion will be in the wide transition region.

A convenient measure of the overall fidelity of the two outputs Y and Y_s is the 2-norm of the error equations in the fundamental frequency region. From Eq. 4.22,

$$\|\overline{Y}(z) - Y(z)\| = \| \, (|\frac{1}{2} - |H_o(z)| \, |)|X_o(z)| \, \|. \tag{4.26}$$

Assuming that the filter has negligible or well controlled pass-band ripple,

$$\|\overline{Y}(z) - Y(z)\| = \| \, \delta(z)|X_o(z)| \, \|. \tag{4.27}$$

Similarly, for the alternate resolving path,

$$\|\overline{Y_s}(z) - Y_s(z)\| = \| \, \delta(z)|X_e(z)| \, \|. \tag{4.28}$$

If the input signal is half-band limited, $|X_e(z)| = |X_o(z)|$ and the errors for either choice of resolving path will be identical. In general this will not be precisely the case and $|X_e(z)| \neq |X_o(z)|$. When this is true *and when searching with a Type I filter*, one of the choices for the resolving path will be better than the other in terms of minimizing the overall error with respect to the output of an ideal filter. For the best possible chance of effectively searching for patterns over many time scales, it would obviously be advantageous to "synchronize" the resolving path in each coder in the tree-structured decomposition so that the minimum error output, $Y(z)$ or $Y_s(z)$, is chosen for every window of input data at every scale step. A real-time test for determining which output is better can be constructed by examining the outputs $Y(z)$ and $Y_s(z)$ more closely.

The squared magnitude of $Y(z)$ may be found by multiplying $Y(z)$ by its complex conjugate:

$$|Y(z)|^2 = Y(z)Y^*(z) = [\frac{1}{2}X_e(z) + z^{-1}H_o(z)X_o(z)][\frac{1}{2}X_e^*(z) + zH_o^*(z)X_o^*(z)]$$

Expanding the right-hand side yields

$$|Y(z)|^2 = \frac{1}{4}|X_e(z)|^2 + |H_o(z)|^2|X_o(z)|^2 + \frac{1}{2}|H_o(z)|[e^{\frac{-j\omega}{2}}X_o(z)X_e^*(z) + e^{\frac{j\omega}{2}}X_o^*(z)X_e(z)],$$

or, recognizing the implicit *cosine* term,

$$|Y(z)|^2 = \frac{1}{4}|X_e(z)|^2 + |H_o(z)|^2|X_o(z)|^2 + |H_o(z)||X_o(z)||X_e(z)|\cos(-\frac{\omega}{2} + \arg(X_o(z)) - \arg(X_e(z))).$$

To visually simplify this equation a bit, define a function $0 \leq \Delta(\omega) \leq 2$ such that

$$\cos(-\frac{\omega}{2} + \arg(X_o(z)) - \arg(X_e(z))) = 1 - \Delta(\omega). \tag{4.29}$$

Now,

$$|Y(z)|^2 = [\frac{1}{2}|X_e(z)| + |H_o(z)||X_o(z)|]^2 - \Delta(\omega)|H_o(z)||X_o(z)||X_e(z)|. \tag{4.30}$$

Recalling the previous definition $|H_o(z)| = \frac{1}{2} - \delta(z)$, Eq. 4.30 may be rewritten as

$$|Y(z)|^2 = \underbrace{[\frac{1}{2}|X_e(z)| + \frac{1}{2}|X_o(z)| - \delta(z)|X_o(z)|]^2}_{a} - \underbrace{\Delta(\omega)|H_o(z)||X_o(z)||X_e(z)|}_{b}. \tag{4.31}$$

The squared magnitude of $Y_s(z)$ may be found by a similar derivation, yielding

$$|Y_s(z)|^2 = \underbrace{[\frac{1}{2}|X_e(z)| + \frac{1}{2}|X_o(z)| - \delta(z)|X_e(z)|]^2}_{c} - \underbrace{\Delta(\omega)|H_o(z)||X_o(z)||X_e(z)|}_{b}. \tag{4.32}$$

The terms marked with underbraces labeled *b* in Eqs. 4.31 and 4.32 are identical. *For the purposes of comparing the norms of $Y(z)$ and $Y_s(z)$, these terms can be dropped, since they make equal contributions to both equations.* Focusing on the terms marked *a* and *c* yields, for $Y(z)$,

$$|Y(z)| \sim \underbrace{\frac{1}{2}|X_e(z)| + \frac{1}{2}|X_o(z)|}_{d} - \delta(z)|X_o(z)|$$

and for $Y_s(z)$,

$$|Y_s(z)| \sim \underbrace{\frac{1}{2}|X_e(z)| + \frac{1}{2}|X_o(z)|}_{d} - \delta(z)|X_e(z)|.$$

The squaring in Eqs. 4.31 and 4.32 has also been dropped because the magnitudes of $X_e(z)$ and $X_o(z)$ are strictly non-negative, and the function $\delta(z)$ is presumed to satisfy $0 \leq \delta(z) \leq \frac{1}{2}$ under the assumption of negligible pass-band ripple in the resolving filter. The terms delimited with underbraces labeled d are common to both equations and can be replaced with a non-negative function $C(z)$ that is everywhere greater than either $\delta(z)|X_e(z)|$ or $\delta(z)|X_o(z)|$. Therefore,

$$|Y(z)| \sim C(z) - \delta(z)|X_o(z)| \tag{4.33}$$

and

$$|Y_s(z)| \sim C(z) - \delta(z)|X_e(z)| \tag{4.34}$$

If, for example, a particular window of input data yielded the result that

$$||Y_s(z)|| > ||Y(z)||,$$

it follows immediately from Eqs. 4.33 and 4.34 that

$$||\delta(z)|X_o(z)||| > ||\delta(z)|X_e(z)|||.$$

From Eqs. 4.27 and 4.28, this is identical to the condition

$$||\overline{Y}(z) - Y(z)|| > ||\overline{Y_s}(z) - Y_s(z)||.$$

Therefore, if $||Y_s(z)|| > ||Y(z)||$, then $Y_s(z)$ is the best choice for the output of the resolving path in terms of minimizing the overall error with respect to the output from an ideal filter. If, on the other hand, $||Y(z)|| > ||Y_s(z)||$, then $Y(z)$ is the best choice. Note, of course, that if the resolving paths were operated with ideal filters, $\delta(z) \equiv 0$, and *the magnitudes of the outputs of the two resolving paths would be identical and ideal.*

From a practical implementation standpoint, the difference between $||\delta(z)|X_o(z)|||$ and $||\delta(z)|X_e(z)|||$ may be relatively small compared to $||X_o(z)||$ and $||X_e(z)||$. In this case, $||Y(z)|| \approx ||Y_s(z)|| \gg |||Y(z)|| - ||Y_s(z)|||$ and it may be easier to numerically distinguish the better resolving path by examining the results of filtering the input signal with the

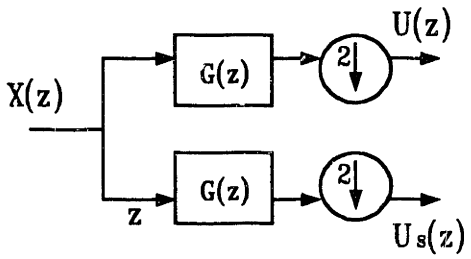


Figure 4.4: High-Pass Indicators for Resolving Path Selection

complementary high-pass filter $G(z)$, defined as for the ideal filter case in Eq. 4.12 and employed as shown in Fig. 4.4. Recapitulating the analysis conducted for the low-pass resolving filter will indicate how the complementary high-pass filter can also be used to select between resolving paths.

The output of the top path shown in Fig. 4.4 may be expressed as

$$U(z) = H_e(z)X_e(z) - z^{-1}H_o(z)X_o(z).$$

Substituting in the expressions for the practical Type I polyphase components,

$$U(z) = \frac{1}{2}X_e(z) - z^{-1}e^{\frac{1\omega}{2}}|H_o(z)|X_o(z) \tag{4.35}$$

The squared magnitude of $U(z)$ may be found by multiplying the right-hand side of Eq. 4.35 by its complex conjugate,

$$|U(z)|^2 = \frac{1}{4}|X_e(z)|^2 + |H_o(z)|^2|X_o(z)|^2 - |H_o(z)||X_o(z)||X_e(z)|(1 - \Delta(\omega)) \tag{4.36}$$

where the function $\Delta(\omega)$ is the same one defined in Eq. 4.29. Simultaneously adding and subtracting a factor of $|H_o(z)||X_o(z)||X_e(z)|$ in Eq. 4.36 will leave the equality unaltered and will permit the equation to be rewritten as

$$|U(z)|^2 = [\frac{1}{2}|X_e(z)| + |H_o(z)||X_o(z)|]^2 - |H_o(z)||X_o(z)||X_e(z)|(2 - \Delta(\omega)). \tag{4.37}$$

Recalling again the definition $|H_o(z)| = \frac{1}{2} - \delta(z)$,

$$|U(z)|^2 = [\frac{1}{2}|X_e(z)| + \frac{1}{2}|X_o(z)| - \delta(z)|X_o(z)|]^2 - \underbrace{|H_o(z)||X_o(z)||X_e(z)|(2 - \Delta(\omega))}_{e}. \quad (4.38)$$

A similar derivation for the bottom path in Fig. 4.4 yields an expression for the squared magnitude of $U_s(z)$:

$$|U_s(z)|^2 = [\frac{1}{2}|X_e(z)| + \frac{1}{2}|X_o(z)| - \delta(z)|X_e(z)|]^2 - \underbrace{|H_o(z)||X_o(z)||X_e(z)|(2 - \Delta(\omega))}_{e}. \quad (4.39)$$

Once again, for purposes of comparing norms, Eqs. 4.38 and 4.39 may be pared of common terms, marked with underbraces labeled e, that are additively identical in both equations. Recalling the function $C(z)$ defined previously, and making simplifications like those used to develop Eqs. 4.33 and 4.34,

$$|U(z)| \sim C(z) - \delta(z)|X_o(z)| \quad (4.40)$$

and

$$|U_s(z)| \sim C(z) - \delta(z)|X_e(z)|. \quad (4.41)$$

If $||U_s(z)|| > ||U(z)||$ for a particular window of input data, then it follows from Eqs. 4.40 and 4.41 that

$$||\delta(z)|X_o(z)||| > ||\delta(z)|X_e(z)|||,$$

indicating that $Y_s(z)$ is the best choice for the output of the resolving path. If the situation were reversed and $||U(z)|| > ||U_s(z)||$, then $Y(z)$ would be the best choice. Again, of course, if the high-pass filters are both ideal, the magnitudes of the outputs of the resolving paths will be identical.

It is clear that the high-pass system in Fig. 4.4 could be used to select the appropriate low-pass resolving path. This approach might be numerically advantageous for input data with relatively little upper half-band energy in the sense that the system in Fig. 4.4 exposes the differences between the low-pass resolving paths as a greater fraction of the overall output energy of the high-pass system. On the other hand, the high-pass system may be more susceptible to noise, and the computation of the high-pass paths is an additional burden. For a given implementation, the decision to use the low-pass paths in Fig. 4.3

directly, or to use the high-pass paths in Fig. 4.4 as an indicator to compute only the optimal low-pass path, is a design trade-off. The high-pass approach may be especially favorable if the high-pass data is useful in other ways, for example as a preprocessor step for pattern recognition or discrimination, in which case the computational burden becomes more tolerable.

Equations 4.33 and 4.34, or equivalently Eqs. 4.40 and 4.41, indicate that the two possible resolving paths in Fig. 4.3 will differ *only* because of magnitude differences in the polyphase components of the input signal. Phase differences are ignored. If an input signal were half-band limited, Eq. 4.17 indicates that there would be no advantage in terms of resolution fidelity in picking one path in Fig. 4.3 over another. In general, however, the input signal will not be half-band limited. If a signal contains high frequency content because it violates Eq. 4.17 only, or significantly relative to the condition in Eq. 4.18, the two possible resolving paths operated *with Type I filters* will differ in the output resolution fidelity. It will be advantageous from a pattern recognition standpoint to be vigilant and select the better path. On the other hand, if the input signal contains high frequency content because it mainly violates Eq. 4.18, the two paths will yield outputs with identical frequency content when operated with Type I filters.

The next section will demonstrate that Type II FIR filters distinguish the case of input data with high frequency content caused by phase deviations from the ideal half-band relationship Eq. 4.18. That is, for input data that mainly violates Eq. 4.18, the two choices in Fig. 4.3 will differ in resolution fidelity when the paths are operated *with Type II filters.*

4.4.2 Resolving with Type II FIR Filters

Type II half-band FIR filters attempt to approximate an ideal filter with polyphase components $\overline{H}_e(e^{j\omega}) = \frac{1}{2}e^{\frac{-j\omega}{4}}$ and $\overline{H}_o(e^{j\omega}) = \frac{1}{2}e^{\frac{j\omega}{4}}$. Both of these ideal polyphase components correspond to infinitely long sequences in time, and, unlike the Type I case, neither can be implemented exactly by an FIR filter. The best that can be done when designing a Type II filter is to preserve the magnitude and phase relationships between the polyphase components. Therefore, for a Type II filter with polyphase components $H_e(e^{j\omega})$ and $H_o(e^{j\omega})$,

$$|H_e(e^{j\omega})| = |H_o(e^{j\omega})| \tag{4.42}$$

and

$$\arg(H_e(e^{j\omega})) = -\arg(H_o(e^{j\omega})) = \alpha(\omega). \tag{4.43}$$

The function $\alpha(\omega) = \arg(H_e(e^{j\omega}))$ is introduced for convenience in the work that follows. One goal of this section is to determine the effect of implementing the two possible resolving paths in Fig. 4.3 with Type II FIR filters. Rather than introducing more notation (e.g., for the outputs of the resolving paths $Y(z)$ and $Y_s(z)$), this section will continue to use the same symbols used in the last section with the understanding that results developed here refer specifically to the action of Type II filters.

The output $Y(z)$ of the top path in Fig. 4.3 is given in Eq. 4.19. The squared magnitude of $Y(z)$ when the filter is a Type II FIR filter may be derived, as in the previous section, by multiplying the right-hand side of Eq. 4.19 by its complex conjugate. Doing so, and substituting in the Type II polyphase filter constraints in Eqs. 4.42 and 4.43, yields:

$$\begin{aligned} |Y(z)|^2 = &\underbrace{|H_e(z)|^2[|X_e(z)|^2 + |X_o(z)|^2]}_{f} + \\ &2|X_e(z)||X_o(z)||H_e(z)|^2 \cos(\arg(X_e(z)) - \arg(X_o(z)) + \omega + 2\alpha(\omega)) \end{aligned} \tag{4.44}$$

The squared magnitude of the bottom path, $Y_s(z)$, may be found by a similar derivation, revealing that

$$\begin{aligned} |Y_s(z)|^2 = &\underbrace{|H_e(z)|^2[|X_e(z)|^2 + |X_o(z)|^2]}_{f} + \\ &2|X_e(z)||X_o(z)||H_e(z)|^2 \cos(\arg(X_e(z)) - \arg(X_o(z)) - 2\alpha(\omega)) \end{aligned} \tag{4.45}$$

For purposes of comparing the norms of the outputs, the terms marked with underbraces labeled f in Eqs. 4.44 and 4.45 contribute equally to both equations and may be ignored. Since the FIR filter will not have an ideal phase characteristic $\alpha(\omega) = \frac{-\omega}{4}$, the *cosine* terms will be different, in general, assuming that the polyphase components of the input signal *do not* obey the half-band limited phase characteristic of Eq. 4.18. For a half-band limited input signal, Eqs. 4.44 and 4.45 yield identical results. Notice that any differences between Eq. 4.44 and 4.45 depend on only the phase characteristic of the input signal and not the magnitude! In general, one of the two outputs will better represent the lower half-band frequency content of the input signal.

To find a numerical indicator of which path is better, consider first the situation when the filters are ideal. If the filters in both paths were ideal, the squared magnitude of the outputs $\overline{Y}(z)$ and $\overline{Y_s}(z)$ would be identical. An exact expression may be found by inserting the ideal filter phase characteristic, $\alpha(\omega) = \frac{-\omega}{4}$, into either Eq. 4.44 or Eq. 4.45:

$$|\overline{Y}(z)|^2 = |\overline{Y_s}(z)|^2 = \underbrace{|\overline{H}_e(z)|^2[|X_e(z)|^2 + |X_o(z)|^2]}_{g} +$$
$$2|X_e(z)||X_o(z)||\overline{H}_e(z)|^2 \cos(\arg(X_e(z)) - \arg(X_o(z)) + \frac{\omega}{2}) \qquad (4.46)$$

Discrepancies in reproducing the contribution of the first term in Eq. 4.46, marked with an underbrace labeled g, will be identical in both Eqs. 4.44 and 4.45 and, as mentioned previously, will not contribute to relative error between Eqs. 4.44 and 4.45. Assume here as in the previous section that the magnitude characteristic of the practical filter has negligible pass-band ripple overshoot and is therefore equal to or less than the ideal magnitude characteristic. In this case, the *cosine* terms in Eqs. 4.44 and 4.45 are premultiplied by an expression, $2|X_e(z)||X_o(z)||H_e(z)|^2$, that can only be equal to or smaller than the ideal term $2|X_e(z)||X_o(z)||\overline{H}_e(z)|^2$. The best choice, therefore, between outputs $Y(z)$ and $Y_s(z)$ is again the output with the larger norm.

Experience from numerical experiments and with the experimental system described in Chapter 5 indicate that when an input signal violates either Eq. 4.17 or Eq. 4.18 more strongly than the other, the best error minimizing results are obtained by operating the resolving path with the FIR filter that distinguishes between magnitude (Type I) or phase (Type II) differences, respectively, and then selecting the best path accordingly. *The best low-pass path is always the one whose output has a larger norm.*

As for the Type I case, the choice of resolving path may also be made in the Type II case by examining the outputs of the structure shown in Fig. 4.4 operated with the complementary Type II high-pass filters. The squared magnitude of the top path is

$$|U(z)|^2 = \underbrace{|H_c(z)|^2[|X_e(z)|^2 + |X_o(z)|^2]}_{h} -$$
$$\underbrace{2|X_e(z)||X_o(z)||H_e(z)|^2 \cos(\arg(X_e(z)) - \arg(X_o(z)) + \omega + 2\alpha(\omega))}_{j} \qquad (4.47)$$

The squared magnitude of the bottom path, $U_s(z)$, may be found by a similar derivation,

revealing that

$$|U_s(z)|^2 = \underbrace{|H_e(z)|^2[|X_e(z)|^2 + |X_o(z)|^2]}_{h} - \underbrace{2|X_e(z)||X_o(z)||H_e(z)|^2 \cos(\arg(X_e(z)) - \arg(X_o(z)) - 2\alpha(\omega))}_{k} \quad (4.48)$$

The terms marked with underbraces labeled h are common to both terms. They make no contribution to differences between the norms of the outputs of the top and bottom paths. As in Eqs. 4.44 and 4.45, the *cosine* terms labeled j and k will generally be different, however, depending on the relative phase characteristics of the polyphase components of the input waveform. Unlike Eqs. 4.44 and 4.45, the terms j and k enter subtractively rather than additively into the equations. Since the *cosine* terms enter into the low-pass equations 4.44 and 4.45 with the opposite sign from the high-pass equations 4.47 and 4.48, $\|Y_s(z)\| > \|Y(z)\|$ when $\|U(z)\| > \|U_s(z)\|$. Similarly, when $\|U_s(z)\| > \|U(z)\|$, then $\|Y(z)\| > \|Y_s(z)\|$. So, the high-pass path with the largest norm indicates that the opposite low-pass path will have the largest output norm and thus represents the best choice of outputs.

4.4.3 Adaptively Selecting Between Resolving Paths

In summary, the tree-structured decomposition could be used to organize a search for a single pattern over many different time scales. For a pattern recognition application, the operation **R** in each resolving path would be selected to be either a Type I or Type II FIR filter, assuming that patterns at a particular scale are well represented by a fraction of the frequency content at a previous, finer (smaller) scale. The linear phase characteristic of the filters minimizes phase distortion. The choice of using either a Type I or Type II filter would be made by examining *a priori* the v-section to be recognized. For a given filter order or tap width, constrained by the desired stop-band attenuation, the type of filter that best minimized the error of the output signal after filtering and downsampling an input signal would be selected. A specific filter of the chosen type would then be designed with minimum pass-band ripple in the regions of significant input frequency content. This design process will be reviewed thoroughly in the next section.

The optimal resolving filter of the selected type is designed to best preserve input

resolution or frequency content for a given prototype input presented in a particular shift. For example, in the case of 2 to 1 decimation at each coder step, prototype input data may be presented with "no shift" or with a "one point shift." Any other shift will be effectively equivalent to one of these, e.g., a two point shift in the input data will be mathematically equivalent to the "no shift" case in terms of error minimization. As will be shown in the next section, a filter is designed to yield optimal results for input data presented with a particular, assumed shift that produces minimum error output. The previous two subsections developed specific mathematical tools for selecting between the two possible sets of output data points indicated in Fig. 4.3.

Unfortunately, of course, real input data will present itself with an arbitrary shift. In practice, resolving path selection would be conducted in between quasi-static regions of the input data. Individual v-sections may be present additively on top of level or near-level waveforms generated by the steady state operation of other loads. Recall, however, that a requirement for successful pattern recognition in the NILM is that v-section supports do not overlap. Only the trivial overlap of a single v-section and a quasi-static, steady state level is permitted. The worst case situation that might still be tractable for nonintrusive monitoring occurs when v-sections appear in a continuous cascade at the service entry, with no overlap in the supports but also no "white space" or constant level regions separating the v-sections. If the service entry is not quite this busy, the opportunity to tune the tree-structured decomposition arises by adaptively selecting resolving paths in between quasi-static regions. After low-pass filtering, a quasi-static region can be detected by searching for a set of adjacent, nearly constant-valued output points. Equivalently, a quasi-static region may also be detected by searching the output of the complementary high-pass filter for adjacent output points of near zero value.

A simple example may serve to illustrate these points clearly. Start with the following sequence of input data:

```
Observations: ...1  1 | 3  3  2  2  2  2  2  2  2 | 4  4  3  3  3  3...
```

The input data contains two transients or v-sections marked with vertical bars (|) to show the start of the transients. Consider the problem of projecting this window of input data into a bigger time scale by filtering and decimating every other point. The data perfectly satisfies

Eq. 4.17 but does not satisfy Eq. 4.18. The wise pattern detective would have therefore selected a Type II half-band FIR filter to bandlimit the resolution before downsampling. To minimize computational effort in this example, let's work with the simplest possible Type II half-band low-pass filter with impulse response [.5 .5]. This filter has very poor stop-band attenuation, too poor to be useful in most practical systems. It will serve well, however, to illustrate key points about resolving path selection.

Before downsampling, the input data window is convolved with the impulse response of the chosen filter. (In a practical implementation, the convolution might be conducted with a computationally efficient fast Fourier transform (FFT). An "overlap and save" scheme would be used to extract from this circular convolution the set of points that coincided with the results of a linear convolution ([102]). Again, for simplicity, such details are ignored here.) The results of this convolution are shown below:

```
Even  Output:      1     3     2     2     2     3     3.5   3
Observations: ... 1  1  3  3  2  2  2  2  2  2  2  4  4  3  3  3  3  ...
Odd   Output:         2     2.5   2     2     2     4     3     3
```

If the window of observations is filtered and downsampled as a contiguous block, the output will be one of the two sequences shown above. The labels `Even` and `Odd` are arbitrary, serving to distinguish the two different output sequences possible because of the time-variant operation of the downsampler. The associated results of convolving with the complementary Type II high-pass filter with impulse response [.5 $-$ .5] are shown below:

```
Even  Output:      0     0     0     0     0     1    -.5    0
Observations: ... 1  1  3  3  2  2  2  2  2  2  2  4  4  3  3  3  3  ...
Odd   Output:         1    -.5    0     0     0     0     0     0
```

These results indicate that neither one of the two low-pass resolving path outputs is wholly satisfactory. In the local neighborhood of the first v-section, the low-pass `Even` output does a better job representing the input data. This is evidenced by the results of convolving with the complementary high-pass filter. In the neighborhood of the first v-section, there is no energy in the `Even` high-pass output, but finite energy in the `Odd` output. On the other hand, in the neighborhood of the second v-section, the low-pass `Odd`

output does a better job representing the input data. If the window of input observations were considered as a whole, either output, **Even** or **Odd**, would seem equally good (or bad!). If the input window had been shifted by one point, the situation would be reversed, i.e., the **Odd** points would best represent the first v-section and the **Even** points would best represent the second. If there had been one more or one less point in the static region separating the two v-sections, both would have been perfectly represented by the **Even** output.

The best answer is to consider each v-section independently and select the appropriate output. In this case, an adaptive selection would presumably be possible because the v-sections are separated by a wide static region. That is, the selected output can be commutated from one path in Fig. 4.3 to the other in the middle of long static or quasi-static regions without destroying the geometry or local resolution of the static section. In this manner, the best output for each v-section in the example could be selected. This is the essence of adaptively selecting optimal resolving path outputs. Naturally, if the v-sections are tightly clustered with no separating static regions, some error may be unavoidable. By keeping the filter impulse response as compact as possible, however, smaller regions will appear static and adaptive selection may occur more often. In other words, compact filters permit optimal resolution preservation even in the presence of high rates of event generation. Given that the filter response will not implement a near ideal low-pass characteristic without a very large number of taps, the implication is that it is best to pick a filter with the smallest number of taps consistent with tolerable stop-band attenuation. Then, amplitude distortion can be minimized by adaptively selecting the best resolving path.

4.5 Filter Construction

Recapitulating briefly, the previous sections have demonstrated that the tree-structured decomposition is a potentially efficient scheme for organizing a pattern search over many time scales. Assuming that patterns at one scale are well represented by a fraction of the frequency content at a previous scale, the resolving path operation **R** could reasonably be implemented as a linear filter. At each coder stage in the tree, three sources of error must be controlled to prevent severe corruption of unidentified patterns: namely phase distortion, aliasing distortion, and amplitude distortion. In the process of designing filters to minimize

the impact of these distortions, the following constraints are identified:

- The filter should have a linear phase characteristic to minimize phase distortion. Therefore, only FIR filters, which are uniquely capable of implementing an exactly linear phase characteristic, will be considered ([29]).
- The impulse response of the filter should be sufficiently wide to provide adequate stop-band attenuation, minimizing aliasing distortion.
- The transition region of the frequency response of the filter will necessarily be of finite width, because the impulse response is finite in extent. Amplitude distortion induced by the imperfect magnitude response in the transition region of a practical filter will be minimized in two ways. First, in the filter design process, the transition region will be made as narrow as possible. Second, adaptive resolving path selection will be implemented so that shift variance in the downsampling action of the sub-band coders will not be permitted to exacerbate the amplitude distortion.
- The pass-band ripple in the filter's frequency response should be minimized to further avoid amplitude distortion and permit adaptive path selection.

For any particular application of sample rate alteration, there will be an ideal filter magnitude characteristic that perfectly limits the bandwidth of the input data to eliminate aliasing and imaging in the output data. There are two broad approaches for designing practical filters to emulate the ideal characteristic. The first is to design the practical filter to best approximate globally the ideal characteristic according to some criteria, with no regard for the potential input data. The second is to design the practical filter according to weighted criteria, where appropriate weighting has been selected based on anticipated input data. By enforcing strict design tolerances only where needed, this second approach permits a potentially higher level of performance, for example, by meeting design specifications with a smaller impulse response than would be necessary to meet strict tolerances everywhere.

A number of FIR filter design techniques have been proposed, including the windowing, frequency-sampling, and equiripple methods, with an innumerable number of variations. A spectrum of approaches may be found in [29], [31], [49], [53], [59], [62], and [120]. For an interpolator, i.e., an upsampler followed by an anti-imaging or interpolating filter, it is possible to independently associate the polyphase components of the output signal with the action of individual polyphase components of the interpolating filter that follows upsampling. This decomposition makes it possible to individually craft the polyphase components of the interpolating filter to minimize an error criterion with repect to the output computed

by ideal polyphase filter components. Custom weighting is easily incorporated to minimize error for an anticipated input signal. Techniques for accomodating a number of error criteria, including least-squared error, *minimax* or minimum ∞-norm error in frequency, and minimax error in time, have been developed for interpolators; see, for example, [29], [30], [98] and [124].

For a decimator, i.e., an anti-aliasing filter followed by a downsampler, it is *not* generally possible to associate output samples with the action of the individual polyphase components of the anti-aliasing filter. This has discouraged the development of filter construction algorithms for decimation applications that tailor the construction of the filter to minimize errors for a particular input signal ([29], [111]). Filters for decimators are typically designed with only global error constraints, using a standard method like windowing or equiripple.

For the special case of 2 to 1 decimation with a Type I FIR filter, Eqs. 4.22 and 4.25 indicate that it is, in fact, possible to optimize the design of the individual polyphase components of a resolving filter for an anticipated input. Since the even polyphase component of the Type I filter can precisely implement the ideal filter characteristic, no output error is associated with the action of this polyphase component. The error in the output of a resolving path decimator with a Type I filter is due solely to imperfections in the magnitude characteristic of the odd polyphase component of the filter, as shown in Eqs. 4.22 and 4.25.

Because of the symmetry of the Type I filter impulse response, the "compressed" sequence $h_o[n]$, which has z-transform $H_o(z)$ and which corresponds to the odd samples of the filter impulse response, has the form of the impulse response of a Type II FIR filter. From Section 4.3, the transform of the ideal odd polyphase component $\overline{H}_o(e^{j\omega}) = \frac{1}{2}e^{\frac{j\omega}{2}}$. The practical odd polyphase component attempts as best as possible to emulate this ideal, all-pass characteristic.

So, the process of designing an appropriate Type I filter for a resolving path can apparently be approached in two steps ([3]). First, an appropriate Type II filter for the odd sequence component of the impulse response would be designed using some filter construction algorithm. Second, the full Type I impulse response would be constructed by upsampling the odd sequence and inserting the single non-zero sample that composes the even sequence. A benefit of this approach is that the design of the odd sample sequence

can be conducted with the goal of minimizing an input-dependent or input-weighted error criterion as in Eq. 4.22. An additional advantage of designing a Type I impulse response with this two-step method is that the filter produced is guaranteed to have zeros in all but one of the even sample locations. Many methods do not precisely ensure that this time domain constraint will be met (see the examples in [103], for example), potentially sacrificing the computational advantage of having a sparse Type I impulse response ([3]). One concern with this technique is that any pass and stop-band ripples will be symmetrical in amplitude. Many applications can tolerate more ripple in the pass-band than in the stop-band. If independent control of pass and stop-band ripples is required, the filter may be designed as a Type I filter, but the independent weighting of error in the polyphase components will generally be lost.

Of the many methods that could be used to design the Type II filter for the odd sequence component of the full Type I filter, the equiripple method has been chosen for this work for three reasons. Equiripple FIR filters are optimal in the sense that they have the smallest ripple for a given transition region width. Also, as the name implies, the unweighted ripple in an equiripple filter has a constant variation throughout the frequency response of the filter. That is, the unweighted crests and troughs of the ripple are identical in magnitude variation from the ideal filter characteristic throughout the frequency response. Filter order can be increased systematically as needed during the design stage to provide acceptably small pass-band ripple. Finally, robust and speedy algorithms are available for the design of equiripple filters.

The difference or error between the ideal and practical filter frequency responses expressed in Eq. 4.22 or Eq. 4.25 is precisely in a form amenable to an application of the *alternation theorem* ([23], [103], [118]). The alternation theorem states necessary and sufficient conditions for minimizing the maximum (the "minimax" method) error or value of the difference in Eq. 4.22 or Eq. 4.25. The Remez algorithm ([23]) – also called the Parks-McClellan algorithm when applied to the problem of FIR filter design ([105], [114], [111]) – provides a stable, convergent scheme for finding an equiripple magnitude characteristic, $|H_o(z)|$, that meets the minimax conditions set forth in the alternation theorem for the difference in Eq. 4.22 or Eq. 4.25.

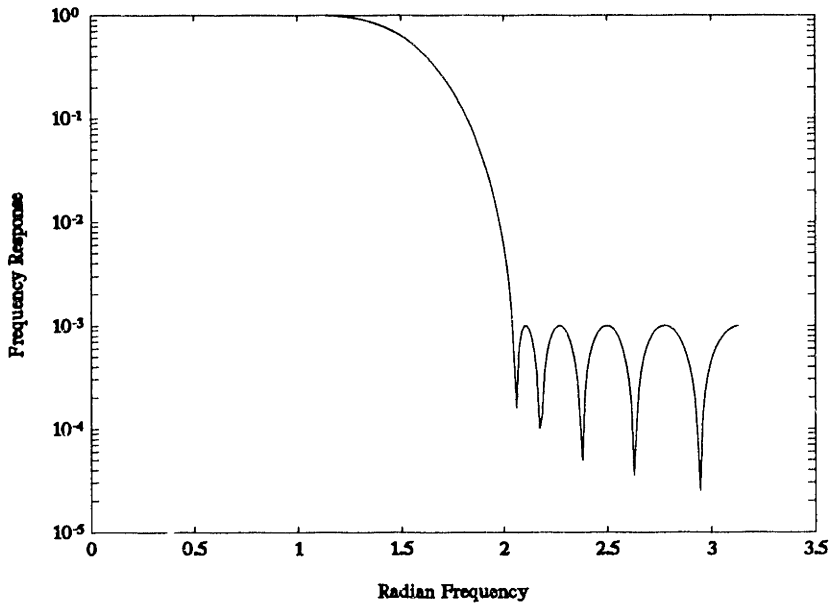


Figure 4.5: Frequency Response of a 23 Tap, Type I Filter

A MATLAB script, *remz*, which implements the Parks-McClellan algorithm for designing both Type I and Type II FIR filters is presented in Appendix D. The ideal filter characteristic to be emulated and also the weighting or template function, which corresponds to either the even or odd polyphase input components, $X_o(z)$ or $X_e(z)$ in Eq. 4.22 or Eq. 4.25, respectively, may be entered by the user as MATLAB scripts. These scripts are called by *remz* in the process of designing a filter that satisfies the optimality conditions specified in the alternation theorem. As an example, the frequency response of a 23 tap, Type I half-band, low-pass filter with uniform weighting of pass and stop-band ripple is shown in Fig. 4.5. The filter was designed by *remz* using the two-step technique described above. The impulse response of the filter is shown in Fig. 4.6, and the pass and stop-band ripple are shown in Fig. 4.7.

For a practical pattern recognition problem, the error term to be minimized, either Eq. 4.22 or Eq. 4.25, would be selected by first determining the magnitudes of the even and odd polyphase components of the multiscale v-sections to be resolved. Clearly, it would be best in terms of minimizing overall error to pair the imperfect, odd polyphase component of the resolving filter with the polyphase input component having the least frequency content in the imperfect transition region. In practice, adaptive resolving path

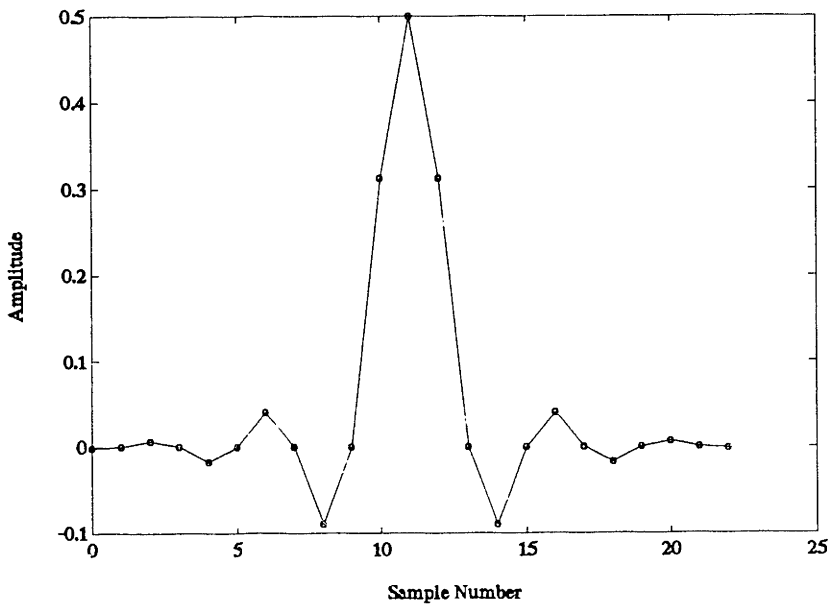


Figure 4.6: Impulse Response of a 23 Tap, Type I Filter

selection will generally ensure that the imperfect, odd polyphase filter component will be paired with the appropriate component of the input.

In a multiscale application, the weighting function indicated by $X_o(z)$ in Eq. 4.22 or by $X_e(z)$ in Eq. 4.25 might be a composite that accounted for the frequency content of all of the potential v-sections present at the finest scale. That is, the ripple characteristic could be graduated so that v-sections to be passed further down the tree would receive the least ripple distortion. For example, the frequency response in Fig. 4.8 with ripple error shown in Fig. 4.9 has a ripple characteristic with a ramped magnitude envelope in the pass-band. The ripple is small at low frequencies for frequency content that will travel through several more filters in successive coder stages. It is larger at frequencies near the half-band point, since input frequency content in this region will not be needed after comparatively few coder stages. Complicated weighting functions tailored to the anticipated multiscale input frequency content could be used in the filter design process. The only requirement, necessary to ensure convergence of the Remez algorithm, is that the weighting function be continuous in the pass and stop bands.

For 2 to 1 decimation with a Type II FIR filter, Eq. 4.42 reveals that magnitude modeling errors will be present symmetrically in the polyphase components of the practical

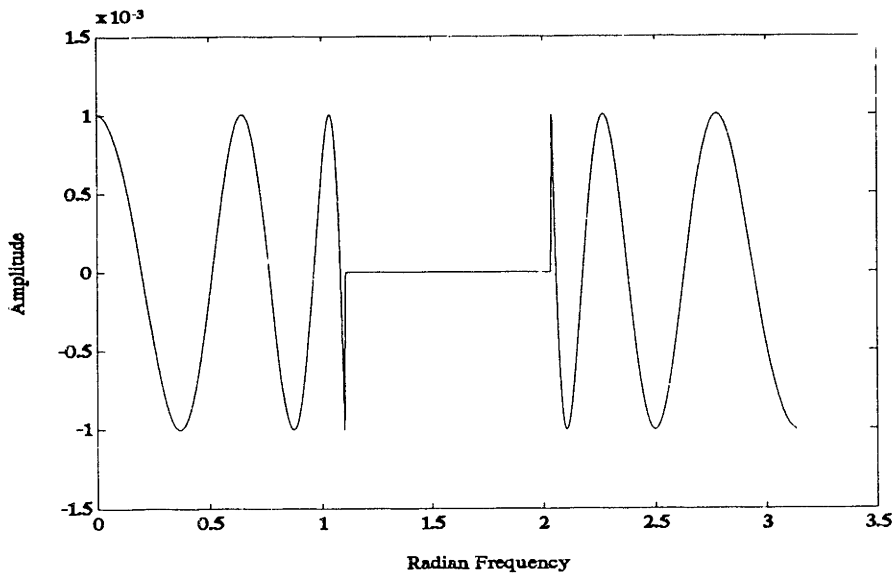


Figure 4.7: Pass-Band and Stop-Band Ripple

filter and Eq. 4.43 indicates that phase errors will occur anti-symmetrically. There is no benefit, therefore, in attempting to tailor the design of individual polyphase filter components based on the content of the polyphase components of the anticipated input signal. Taken as a whole, the complete Type II filter may, of course, be designed to minimize an error constraint based on the total input frequency content. The Remez algorithm may be used, for example, to design a practical filter with a weighted minimax error based on the frequency content of a particular v-section. The *remz* software will perform this design if provided with the frequency content of a v-section, or an aggregated profile of a multiscale set of v-sections, for use as a weighting template in the appropriate MATLAB script.

4.6 Generalized Scale Alteration

It may be desirable to operate the resolving paths in the tree-structured decomposition in Fig. 4.2 at other than 2 to 1 scale change ratios. If neither the upsampling rate l or the downsampling rate m is unity, and operation $\mathbf{R}$ is a linear filter, the filter serves a dual role, limiting the effects of both imaging and aliasing. It will be convenient in this section to think of operation $\mathbf{R}$ as two separate filters or filter banks, one for anti-imaging and the other for anti-aliasing. Naturally, this cascade of filters could be implemented as a single

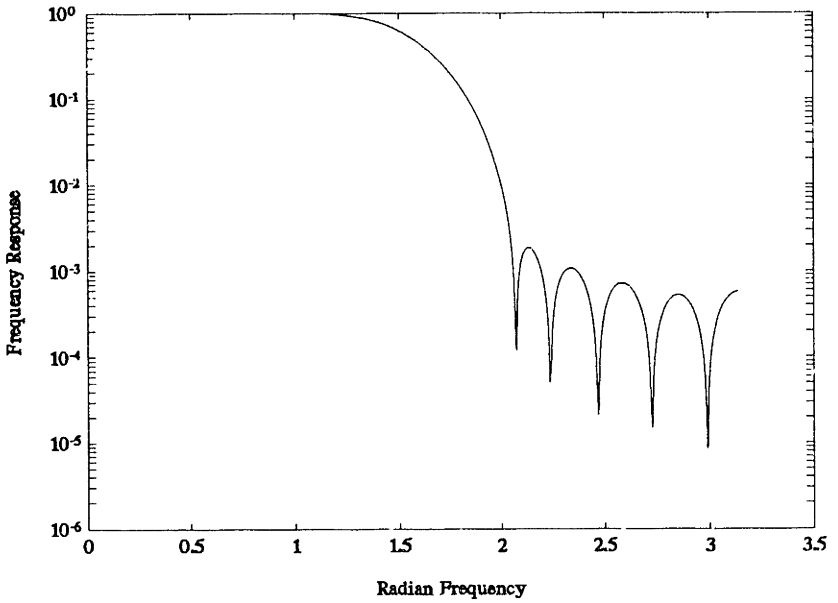


Figure 4.8: Frequency Response with Ramped Ripple

filter in practice.

Consider the case of an L-fold interpolator, i.e., an L-fold upsampler followed by a low-pass filter that attempts to bandlimit the output signal to have frequency content only in the region $-\frac{\pi}{L} < \omega < \frac{\pi}{L}$. The polyphase components of the practical filter that follows upsampling will not implement the exact ideal components necessary to perfectly reject all imaging effects in the output signal. As mentioned in the previous section, for an interpolator it is possible to independently associate errors in the output signal with the action of individual polyphase components in the anti-imaging filter. In particular, the filter may be constructed to pass the original input samples unchanged. For the interpolated samples in between the original samples, the filter may be constructed to minimize a variety of error criteria between the actual interpolated samples and their ideal values. For example, the minimization of mean-squared error is considered in [29] and [111], and a minimax metric is considered in [106]. Interpolation, therefore, can be conducted with no loss in information content in comparison with the original signal. Thus, there is no dual result for interpolating that would be analogous to the resolving path selection criteria needed for decimating.

The second half of a resolving path will consist of an M-fold decimator, i.e., an M-fold downsampler preceded by an M-th band anti-aliasing filter that bandlimits the input

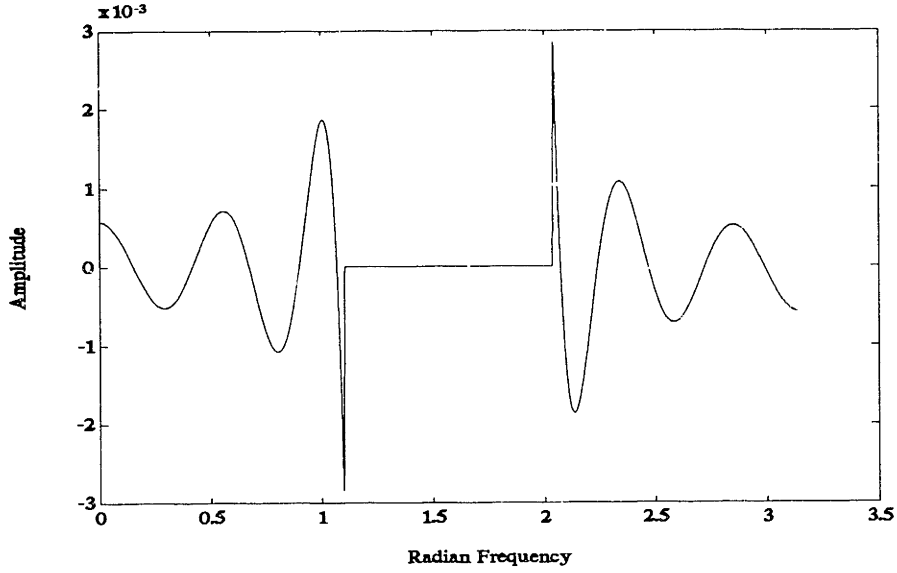


Figure 4.9: Ramped Pass-Band and Stop-Band Ripple

signal to the region $-\frac{\pi}{M} < \omega < \frac{\pi}{M}$. As has been shown for the case of 2 to 1 decimation, the time-variance of the downsampler is a significant problem for pattern recognition applications. This problem persists for M to 1 decimation because, again, the input resolution will not typically be preserved. In general, there will be M possible input shifts to consider for the decimation operation, or M possible choices for the output of a particular resolving path. Assuming that the M-th band filter has negligible pass-band ripple, or, more specifically, that at no frequency does the magnitude response of the practical M-th band filter significantly exceed the magnitude response of the ideal prototype filter, the resolving path selection criteria developed in Section 4.4 extend directly to the case of M-fold decimation. From among the M significant input shifts, the best output – which minimizes the norm of the difference between the actual output and a theoretically ideal output – will be the output with the largest norm.

Unfortunately, it is not generally possible to associate errors in individual polyphase components of the output of an M-fold decimator with polyphase components of the anti-aliasing filter. For a practical half-band Type I filter, the fact that one of the *two* polyphase components of the filter precisely implements the ideal component results in a concentration of error in the other polyphase filter component, as shown in Eq. 4.22 for example. This fact

was exploited in the previous section to develop a design technique for minimizing filter error especially in regions of significant anticipated input frequency content. For higher Type I M-band filters, e.g., a third-band filter for a 3 to 1 decimation application, one of the M polyphase filter components may be ideal, but the other $M-1$ components will only approximate ideal characteristics. These components are not in a form suitable for design by the Remez algorithm, so the design trick in the last section will not be applicable. It may be possible to adapt linear programming or other optimization techniques to the design of these components. The value of such an effort will be highly application dependent. For Type II M-band filters, no such trick was available even for the half-band case.

For large decimation ratios (and/or for large interpolating ratios), it has been shown that it is computationally advantageous to implement the decimation as a cascade of decimators with individually smaller downsampling rates ([29], [143]). For example, a scale change of 32 to 1 might be implemented very efficiently by a resolving path that consisted of a cascade of five 2 to 1 decimators. This approach has an additional benefit. In a pattern recognition problem, it seems likely that there will be some flexibility in the selection of scale change rates at each coder step. That is, if a 31 to 1 scale change is desired, a 32 to 1 scale change is likely to be equally useful. So, many useful scale change rates may be implemented with cascades of decimators individually designed and employed with the adaptive path selection techniques and filter design tools developed in the previous sections.

In closing, and on a somewhat different note, remember that all of the work so far has been predicated on the "resolution assumption" that v-sections at one scale are well represented by a fraction of the frequency content at a previous scale. In this case, a linear filtering operation is satisfactory for ensuring that only the appropriate resolution is passed to successive coder stages. Empirically, this assumption appears to be a good one for the NILM, and the results developed so far will be pivotal in the experimental implementation in the next chapter.

If, for a particular pattern recognition problem, the resolution assumption does not seem to be a good one, other approaches to resolution alteration could be considered. One technique considered in the early experimentation for this thesis involved implementing operation **R** with a *median filter*. The median filter is a nonlinear filter that has been successfully analyzed in terms of its action on input signal geometry ([5], [7], [67], [116]).

Leaving any attempt at rigor entirely to the references, the median filter may be thought of as passing signals that consist of edges and static regions unchanged. Signal features that are impulsive or oscillatory are removed or reduced. The extent or "geometric bandwidth" of signal features that are unchanged by median filtering depends on the order or size of the median filter. If an underlying signal to be pattern-matched can be expressed as a collection of edges and constant neighborhoods whose extents vary with scale, the median filter is a reasonable choice for operation **R**. This type of signal could conceivably be generated, for example, by a pattern recognition system with an edge detector as a preprocessor, or with a preprocessor that employed multi-rate sample-and-hold operations. Signal structures that will not be edges or constant neighborhoods after downsampling will be eliminated by the median filter before the scale is altered.

The results of experiments with a hardware median filter, potentially useful for a real-time implementation of a tree-structured decomposition with nonlinear resolution alteration, are presented in Appendix E. Other nonlinear operators with different resolution altering properties are discussed in references [86], [87], and [109]. Since standard frequency domain analysis techniques are not typically useful for understanding the operation of these nonlinear operators, it is difficult to make precise or general statements about the use of such operators in a pattern recognition setting. Fortunately, linear filtering appears to be quite adequate for the NILM, and will be the focus of the remainder of this chapter and the hardware implementation in the next chapter.

4.7 Time Domain Observations

All of the mathematical tools necessary to implement a multiresolution event detector for the NILM have now been presented. This section develops tools that can be used to connect the results derived previously with time domain interpretations. Some new results will also be presented that are not essential to the development of the prototype in the next chapter, but which serve to connect some of this work with results and observations in the wavelet and signal processing literatures.

When operation **R** is a linear filter, the convolutions in the tree-structured decomposition might be implemented as circular convolutions in practice ([128]). To help make the

work in this section mathematically tractable and still meaningful from an implementation standpoint (e.g., no infinite dimensional systems or matrices), circular convolutions will be assumed for the filtering operations in the tree-structured decomposition. Circular convolutions are useful in the wavelet transform, for example, to preserve the degrees of freedom going from the data to the transform space and back again. That is, the circular convolution provides precisely as many transform coefficients as input data points ([84], [45]). Circular convolutions are useful in the signal processing and pattern recognition settings even when reconstruction is not necessarily a concern, because the FFT can be used to implement the circular convolution operation efficiently.

Recalling that the circular convolution corresponds to assuming that the window of input data repeats periodically, all of the derivations with z-transforms presented in the previous sections are still valid. Only a discrete set of frequencies will be relevant in the z-transform of a discrete, periodic sequence, and therefore the complex variable z will now have the property that $z^N = 1$ where N is the number of points in the data window. This is equivalent to "sampling" the DTFT of a nonperiodic input waveform which consists of a single period of the repeating discrete-time sequence, or computing the discrete Fourier transform of the data ([102]). Edge effects or "wrap-around" distortions are typically minimized by ensuring that the support of the filter impulse response is small compared to the width of the input data window. Several algorithms for computing a linear convolution with a cascade of circular convolutions have been developed, such as the "overlap and save" algorithm described in [102] and [103]. Hence, even when the input data is not actually periodic, the circular convolution may be used repetitively as a tool for computing the linear convolution.

This work proceeds as in [84] by assuming that the input data really is periodic, and that wrap-around distortion is not a concern. Naturally, this will not be the case for the NILM, but, again, the distortion may not be significant when the filter impulse response is small. Modifications necessary to moderate the impact of wrap-around distortion will also be presented. The tree-structured decomposition operating with 2 to 1 scale changes is considered first, followed by generalizations for other sample rate alterations.

In the time domain, a circular convolution is equivalent to multiplying a vector of input data by a circulant matrix, whose rows might, for example, consist of shifted versions

of a filter impulse response ([132]). Assuming circular convolution, the action of either of the resolving paths in Fig. 4.3 may be expressed as

$$y = Mx, \tag{4.49}$$

where x is an input vector of N sample points and y is an output vector containing $\frac{N}{2}$ sample points that will serve as the input vector for the next coder in the tree. The matrix M is a *decimated* circulant operator with $\frac{N}{2}$ rows and N columns. For a causal filter with impulse response $h[n]$ with, say, four non-zero samples, e.g., a Type II FIR filter, the matrix M will have the form

$$M = \begin{bmatrix} h[0] & h[1] & h[2] & h[3] & 0 & 0 & 0 & 0 & \\ 0 & 0 & h[0] & h[1] & h[2] & h[3] & 0 & 0 & \\ 0 & 0 & 0 & 0 & h[0] & h[1] & h[2] & h[3] & \\ & & & & & & & & \ddots \end{bmatrix}$$

for the *top* resolving path in Fig. 4.3. For example, for a 4×8 matrix M in Eq. 4.49 that produces a 4 point output vector from an 8-point input, the matrix would be

$$M = \begin{bmatrix} h[0] & h[1] & h[2] & h[3] & 0 & 0 & 0 & 0 \\ 0 & 0 & h[0] & h[1] & h[2] & h[3] & 0 & 0 \\ 0 & 0 & 0 & 0 & h[0] & h[1] & h[2] & h[3] \\ h[2] & h[3] & 0 & 0 & 0 & 0 & h[0] & h[1] \end{bmatrix} \tag{4.50}$$

For the *bottom* resolving path in Fig. 4.3, the input vector would be shifted by a single sample location. Since the input is assumed to be periodic in a circular convolution, this process could be equally well implemented by shifting all of the rows in the convolution matrix by one sample location. This corresponds to replacing the matrix M in Eq. 4.49 with a matrix M_s,

$$M_s = \begin{bmatrix} 0 & h[0] & h[1] & h[2] & h[3] & 0 & 0 & 0 \\ 0 & 0 & 0 & h[0] & h[1] & h[2] & h[3] & 0 \\ h[3] & 0 & 0 & 0 & 0 & h[0] & h[1] & h[2] \\ h[1] & h[2] & h[3] & 0 & 0 & 0 & 0 & h[0] \end{bmatrix},$$

which computes the output y_s from the input vector x. The matrices M and M_s are related by an $N \times N$ *permutation matrix* P, which serves to rotate the first and last columns of M to create M_s, i.e.,

$$MP = M_s.$$

Permutation matrices are orthonormal [133], [48]. For the 4 × 8 case, for example,

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Rather than clutter the variables with subscripts to indicate the dimensions of a particular convolution or permutation matrix, context will be relied upon to imply appropriate dimensions.

The top and bottom paths in Fig. 4.4 may be implemented by an equation similar to Eq. 4.49:

$$u = Wx, \tag{4.51}$$

For the *top* path, the $\frac{N}{2}$-point high-pass output u, is found by multiplying the input data vector x by an $\frac{N}{2} \times N$ matrix W, for example,

$$W = \begin{bmatrix} g[0] & g[1] & g[2] & g[3] & 0 & 0 & 0 & 0 \\ 0 & 0 & g[0] & g[1] & g[2] & g[3] & 0 & 0 \\ 0 & 0 & 0 & 0 & g[0] & g[1] & g[2] & g[3] \\ g[2] & g[3] & 0 & 0 & 0 & 0 & g[0] & g[1] \end{bmatrix}.$$

For the *bottom* path, the matrix W in Eq. 4.51 is replaced by an $\frac{N}{2} \times N$ matrix W_s,

$$W_s = \begin{bmatrix} 0 & g[0] & g[1] & g[2] & g[3] & 0 & 0 & 0 \\ 0 & 0 & 0 & g[0] & g[1] & g[2] & g[3] & 0 \\ g[3] & 0 & 0 & 0 & 0 & g[0] & g[1] & g[2] \\ g[1] & g[2] & g[3] & 0 & 0 & 0 & 0 & g[0] \end{bmatrix}$$

which computes the $\frac{N}{2}$-point output vector u_s from the input data. Both W and W_s are decimated circulant matrices whose rows consist of shifted versions of the impulse response $g[n]$ of a complementary high-pass filter. They are related by the same permutation matrix P as the identically dimensioned matrices M and M_s – i.e., $WP = W_s$.

The norms of the output vectors y, y_s, u and u_s are proportional to the norms of the discrete Fourier transforms of these vectors by virtue of Parseval's relationship ([128]). A reasonable suspicion, therefore, is that time-domain versions of the tests in Section 4.4,

acting on the norms of the output vectors, will indicate the best choice of resolving path. This is indeed the case, as will be shown. The derivation of these time domain tests not only yields useful tools for a practical implementation, but also affords an opportunity to examine other schemes, like the wavelet transform, in a way different from typical presentations. The first step, taken in the next section, will be to present a decomposition technique for decimated circulant matrices. Among other uses, this technique will expose a computationally efficient means for determining the singular values ([48]) of a matrix like M. The singular values will play an important role in understanding the relative benefits of one resolving path over another.

4.7.1 Condition Numbers of Decimated Circulant Matrices

The decomposition developed here for decimated circulant matrices is based on a known technique for using the discrete Fourier transform (DFT) matrix to diagonalize an undecimated circulant matrix ([132]). First, a few definitions will be needed. Let the matrix F represent the $m \times m$ DFT matrix, shown below for the case $m = 4$,

$$F = \frac{1}{\sqrt{m}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \alpha & \alpha^2 & \alpha^3 \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 \\ 1 & \alpha^3 & \alpha^6 & \alpha^9 \end{bmatrix},$$

where

$$\alpha = e^{-j2\pi/m}.$$

Again, context, rather than an explicit notation, will be relied upon to indicate the conformal DFT matrix dimensions. The DFT matrix is unitary. Therefore, if the superscript $*$ indicates a conjugate transpose,

$$F^*F = FF^* = I$$

where I is the $m \times m$ identity matrix. The DFT of an h-point sequence w is an h-point sequence $\hat{w}$, defined as

$$\hat{w}[k] = \sum_{j=0}^{h-1} w[j]\alpha^{jk} \tag{4.52}$$

for $0 \leq k \leq h-1$. All of the examples in this section will be illustrated with the 2 to 1 decimated, $m \times n$ circulant matrix M shown in Eq. 4.50, with $m = 4$ and $n = 8$. Extensions

to other dimensions should be obvious. Results for other scale change ratios will be discussed at the end of this subsection.

The product of the matrices F and M,

$$FM = \frac{1}{\sqrt{m}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \alpha & \alpha^2 & \alpha^3 \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 \\ 1 & \alpha^3 & \alpha^6 & \alpha^9 \end{bmatrix} \begin{bmatrix} h[0] & h[1] & h[2] & h[3] & 0 & 0 & 0 & 0 \\ 0 & 0 & h[0] & h[1] & h[2] & h[3] & 0 & 0 \\ 0 & 0 & 0 & 0 & h[0] & h[1] & h[2] & h[3] \\ h[2] & h[3] & 0 & 0 & 0 & 0 & h[0] & h[1] \end{bmatrix},$$

may be written as

$$FM = \overbrace{\begin{bmatrix} \hat{h}_e[0] & 0 & 0 & 0 \\ 0 & \hat{h}_e^*[1] & 0 & 0 \\ 0 & 0 & \hat{h}_e[2] & 0 \\ 0 & 0 & 0 & \hat{h}_e^*[3] \end{bmatrix}}^{D_e} \overbrace{\frac{1}{\sqrt{m}} \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & \alpha & 0 & \alpha^2 & 0 & \alpha^3 & 0 \\ 1 & 0 & \alpha^2 & 0 & \alpha^4 & 0 & \alpha^6 & 0 \\ 1 & 0 & \alpha^3 & 0 & \alpha^6 & 0 & \alpha^9 & 0 \end{bmatrix}}^{G_0} +$$

$$\underbrace{\begin{bmatrix} \hat{h}_o[0] & 0 & 0 & 0 \\ 0 & \hat{h}_o^*[1] & 0 & 0 \\ 0 & 0 & \hat{h}_o[2] & 0 \\ 0 & 0 & 0 & \hat{h}_o^*[3] \end{bmatrix}}_{D_o} \underbrace{\frac{1}{\sqrt{m}} \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & \alpha & 0 & \alpha^2 & 0 & \alpha^3 \\ 0 & 1 & 0 & \alpha^2 & 0 & \alpha^4 & 0 & \alpha^6 \\ 0 & 1 & 0 & \alpha^3 & 0 & \alpha^6 & 0 & \alpha^9 \end{bmatrix}}_{G_1}. \quad (4.53)$$

The sequence $\hat{h}_e[k]$ is the m-point DFT of the sequence $h_e[j]$ constructed from all of the even samples of the first row of M. For the matrix M in the example above, the samples $h_e[j]$ are $[h[0]\ h[2]\ 0\ 0]$ for $j = 1 \ldots m$. Similarly, $\hat{h}_o[k]$ is the DFT of the m point sequence $h_o[j]$. The sequence $h_o[j]$ is $[h[1]\ h[3]\ 0\ 0]$ in the example for $j = 1 \ldots m$. The superscript $*$ indicates simple conjugation when applied to a scalar. Equation 4.53 is perhaps best verified by evaluating the products and sums and comparing the results to multiplying F and M directly. For convenience, the matrices in Eq. 4.53 have been marked with labeled braces that permit the equation to be written in the compact form

$$FM = D_e G_0 + D_o G_1 \quad (4.54)$$

Two useful observations about the matrices G_0 and G_1 are that, because of the structural location of zeros in the two matrices,

$$G_0 G_1^* = G_1 G_0^* = [0], \quad (4.55)$$

where $[0]$ represents an $m \times m$ matrix filled with zeros, and that, because F is unitary,

$$G_0 G_0^* = G_1 G_1^* = I, \quad (4.56)$$

where I is the $m \times m$ identity matrix. The decomposition in Eq. 4.54 will prove to be remarkably useful and illuminating. One immediate application is to provide an easy means for determining the condition number of the matrix M.

The development here is stimulated by the use of the Rayleigh quotient ([133]) in [90] for the determination of asymptotic bounds on the condition number of non-decimated circulant matrices ([132]). Let the symbol $\kappa(M)$ represent the condition number of M, i.e., the ratio of the largest to the smallest of the singular values of M. The singular value decomposition ([48]) may be used to verify that

$$\kappa(M) = \sqrt{\kappa(MM^T)}.$$

Because F is unitary,

$$\kappa(M) = \sqrt{\kappa(MM^T)} = \sqrt{\kappa(FMM^TF^*)}. \tag{4.57}$$

Because the matrix FMM^TF^* is symmetric, its singular values are identical to its eigenvalues, and therefore

$$\kappa(FMM^TF^*) = \frac{\lambda_{max}}{\lambda_{min}} \tag{4.58}$$

where λ_{max} and λ_{min} refer to the largest and smallest eigenvalues of FMM^TF^*, respectively.

The Rayleigh quotient may be used to determine the eigenvalue λ_{max},

$$\lambda_{max} = \max_{x \neq 0} \frac{x^*FMM^TF^*x}{x^*x} \tag{4.59}$$

where x may be any non-zero m-dimensional vector. Similarly, the smallest eigenvalue λ_{min} is

$$\lambda_{min} = \min_{x \neq 0} \frac{x^*FMM^TF^*x}{x^*x}. \tag{4.60}$$

Now, the decomposition in Eq. 4.54 may be used to simplify the matrix FMM^TF^* as follows:

$$FMM^TF^* = (FM)(FM)^* = [D_eG_0 + D_oG_1][D_eG_0 + D_oG_1]^*$$

Expanding the right-hand side of this equation,

$$FMM^TF^* = D_eG_0G_0^*D_e^* + D_oG_1G_1^*D_o^* + D_eG_0G_1^*D_o^* + D_oG_1G_0^*D_e^*.$$

This may be simplified further by recalling Eqs. 4.55 and 4.56 to yield

$$FMM^TF^* = D_eD_e^* + D_oD_o^* = D$$

where D is a diagonal, $m \times m$ matrix which consists of the sums of the squared magnitudes of the even and odd frequency components $\hat{h}_e$ and $\hat{h}_o$. For the 4×4 case, for example,

$$FMM^TF^* = D = \begin{bmatrix} |\hat{h}_e[0]|^2 + |\hat{h}_o[0]|^2 & 0 & 0 & 0 \\ 0 & |\hat{h}_e[1]|^2 + |\hat{h}_o[1]|^2 & 0 & 0 \\ 0 & 0 & |\hat{h}_e[2]|^2 + |\hat{h}_o[2]|^2 & 0 \\ 0 & 0 & 0 & |\hat{h}_e[3]|^2 + |\hat{h}_o[3]|^2 \end{bmatrix}$$

or simply

$$D = \begin{bmatrix} (\hat{d}[0])^2 & 0 & 0 & 0 \\ 0 & (\hat{d}[1])^2 & 0 & 0 \\ 0 & 0 & (\hat{d}[2])^2 & 0 \\ 0 & 0 & 0 & (\hat{d}[3])^2 \end{bmatrix} \tag{4.61}$$

where

$$\hat{d}[i] = \sqrt{|\hat{h}_e[i]|^2 + |\hat{h}_o[i]|^2}.$$

The Rayleigh quotients for λ_{max} and λ_{min} may be simplified by substituting Eq. 4.61 into Eqs. 4.59 and 4.60, respectively. For λ_{max},

$$\lambda_{max} = \max_{x \neq 0} \frac{x^*Dx}{x^*x}.$$

Since D is a diagonal matrix,

$$\lambda_{max} = \hat{d}_{max}^2$$

where

$$\hat{d}_{max} = \max_i \hat{d}[i].$$

For λ_{min},

$$\lambda_{min} = \min_{x \neq 0} \frac{x^*Dx}{x^*x} = \hat{d}_{min}^2$$

where

$$\hat{d}_{min} = \min_i \hat{d}[i].$$

Substituting these expressions for λ_{max} and λ_{min} into Eq. 4.58 reveals that

$$\kappa(FMM^TF^*) = \frac{\hat{d}^2_{max}}{\hat{d}^2_{min}}$$

So, from Eq. 4.57, it is possible to conclude that

$$\kappa(M) = \frac{\hat{d}_{max}}{\hat{d}_{min}} \tag{4.62}$$

This neat result makes it possible to evaluate the conditioning of a proposed analysis or analysis-synthesis system which makes use of decimated circulant matrices, simply by examining the DFT of the analysis filter used to construct the matrix. Notice that the technique is applicable to either Type I or Type II filters. The impulse response used to construct the rows of M may be turned into a Type I, odd tap response, for example, by simply setting $h[3] = 0$. Of course, the DFT of the sequence h_o will have to be recomputed, but the decomposition in Eq. 4.54 will remain structurally unchanged.

The decomposition is easily extended to decimated circulant matrices with other decimation or scale change ratios. The only difference is that the entries $\hat{d}[i]$ of the diagonal matrix D will each consist of the sum of squares of a different number of polyphase components. With a scale change of 3 to 1, for instance, the entries of D will each consist of the sum of squares derived from three polyphase sequences h_a, h_b, and h_c, which will be different in general from the two polyphase sequences used in the example above.

4.7.2 Time Domain Tests for Resolving Path Selection

A number of insights follow from the decomposition in Eq. 4.54 and the condition number computation in Eq. 4.62. Consider first the action of an ideal half-band filter in a circulant, 2 to 1 decimation system. Recall that the DFT of the polyphase components of an ideal half-band filter, $\hat{h}_e[k]$ and $\hat{h}_o[k]$, will consist of samples of the DTFT of the polyphase components of the ideal filters examined in Section 4.3. For either a Type I or a Type II precursor, discrete-time, discrete-frequency, ideal half-band filter, the magnitudes of the polyphase components will all be equal to a constant, c. For the scaling used in Section 4.3, $c = \frac{1}{2}$. For an $\frac{n}{2} \times n$ circulant matrix $\overline{M}$ whose rows consist of circularly shifted versions of the globally supported (i.e., n point) impulse response of an ideal half-band filter, Eq. 4.62 indicates that $\kappa(\overline{M}) = 1$. The rows of $\overline{M}$, therefore, form an orthogonal basis for the

"low-pass half-band" space that contains the exact half-band component of the input data. That is, if an n-point input vector were known to be perfectly half-band limited, it could be expressed exactly as a linear combination of the $\frac{n}{2}$ vectors of length n comprising the row space of $\overline{M}$.

The complementary Type I and Type II high-pass, half-band filters also have polyphase components with constant magnitudes. The collection of shifted impulse responses of these filters, i.e., the row space of a matrix $\overline{W}$, form an orthogonal basis for a "high-pass half-band" space which is the orthogonal complement of the low-pass space. If an input vector consisted solely of high frequency components in the upper half-band, it could be expressed perfectly as a linear combination of the row space of $\overline{W}$.

The conclusion was made in Section 4.4 that, operated on with ideal filters, the outputs of the resolving paths in Fig. 4.3 would be identical in magnitude. This may be demonstrated again with the help of the decomposition in Eq. 4.54, by comparing the norms of the two output vectors $\overline{y}$ and $\overline{y}_s$ after filtering an input vector x with ideal half-band decimated circulant matrices $\overline{M}$ and $\overline{M_s}$, respectively. For the matrix equivalent of the top path in Fig. 4.3, the output vector $\overline{y}$ is

$$\overline{y} = \overline{M}x.$$

The norm of the output vector y is

$$\|\overline{y}\| = \|\overline{M}x\| = \|F\overline{M}x\|$$

where the last equality is possible because the DFT matrix F is unitary. Employing Eq. 4.54 yields

$$\|\overline{y}\| = \|[D_eG_0 + D_oG_1]x\|$$

so the squared norm of $\overline{y}$ is

$$\|\overline{y}\|^2 = x^T[D_eG_0 + D_oG_1]^*[D_eG_0 + D_oG_1]x.$$

Expanding the right-hand side,

$$\|\overline{y}\|^2 = x^T \underbrace{[G_0^*D_e^*D_eG_0 + G_1^*D_o^*D_oG_1 + G_0^*D_e^*D_oG_1 + G_1^*D_o^*D_eG_0]}_{A} x \tag{4.63}$$

or, identifying the sum of matrices marked with an underbrace labeled A as a single real, symmetric matrix A,

$$\|\overline{y}\|^2 = x^T A x. \tag{4.64}$$

For the matrix equivalent of the *bottom* path in Fig. 4.3, the output vector $\overline{y}_s$ is

$$\overline{y}_s = \overline{M_s} x = \overline{M} P x \tag{4.65}$$

where P is a permutation matrix. Before computing the norm of $\overline{y}_s$, the introduction of a few identities will be helpful. By examining the matrix definitions in Eq. 4.53, notice that

$$G_0 P = G_1 \tag{4.66}$$

and also that

$$G_1 P = S G_0 \tag{4.67}$$

where the *shift* matrix S is an $\frac{n}{2} \times \frac{n}{2}$ diagonal matrix whose diagonal entries consist of the elements of the vector

$$[1 \; \alpha^{-1} \; \alpha^{-2} \ldots \alpha^{-\frac{n}{2}+1}].$$

For example, for the $\frac{n}{2} = 4$ case considered previously,

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \alpha^{-1} & 0 & 0 \\ 0 & 0 & \alpha^{-2} & 0 \\ 0 & 0 & 0 & \alpha^{-3} \end{bmatrix}.$$

Finally, refer back to the ideal filter polyphase components presented in Section 4.3. Notice that for an ideal, linear phase half-band filter, the diagonal matrices D_e and D_o will contain diagonal entries that consist of frequency samples of either the DTFT ideal polyphase components c and $ce^{j\frac{\omega}{2}}$ (Type I precursor) or components $ce^{-j\frac{\omega}{4}}$ and $ce^{j\frac{\omega}{4}}$ (Type II precursor). For the 4×4 case, for example, the two matrices D_e and D_o might be *either*

$$D_e = cI \;\text{ and }\; D_o = cS^{-\frac{1}{2}} \tag{4.68}$$

or

$$D_e = cS^{\frac{1}{4}} \;\text{ and }\; D_o = cS^{-\frac{1}{4}}. \tag{4.69}$$

In *either* case,

$$D_e^* D_o S = D_e D_o^* \quad \text{and} \quad S^* D_o^* D_e = D_o D_e^* \tag{4.70}$$

Also,

$$D_e^* D_e = D_o^* D_o = c^2 I \tag{4.71}$$

Now, returning to Eq. 4.65, the norm of the output vector $\overline{y}_s$ is

$$\|\overline{y}_s\| = \|\overline{M} P x\| = \|F \overline{M} P x\|.$$

Employing the decomposition in Eq. 4.54 and the identities in Eqs. 4.66 and 4.67,

$$\|\overline{y}_s\| = \|[D_e G_1 + D_o S G_0] x\|$$

so the squared norm of y_s is

$$\|\overline{y}_s\|^2 = x^T [D_e G_1 + D_o S G_0]^* [D_e G_1 + D_o S G_0] x.$$

Expanding the left-hand side of this equation yields

$$\|\overline{y}_s\|^2 = x^T \underbrace{[G_1^* D_e^* D_e G_1 + G_0^* S^* D_o^* D_o S G_0 + G_1^* D_e^* D_o S G_0 + G_0^* S^* D_o^* D_e G_1]}_{B} x \tag{4.72}$$

With the help of the identities in Eqs. 4.70 and 4.71, the sum of matrices marked with an underbrace labeled B may be identified as the conjugate transpose of the real, symmetric matrix A. Therefore,

$$\|\overline{y}_s\|^2 = x^T A^* x = x^T A x = \|\overline{y}\|^2. \tag{4.73}$$

Since the squared norms of $\overline{y}$ and $\overline{y}_s$ are equal, so are the norms themselves, producing the expected and previously derived result that, for ideal half-band, linear phase filters, the two resolving paths in Fig. 4.3 produce outputs with identical spectral magnitudes. The norms of $\overline{u}$ and $\overline{u}_s$, the outputs of the top and bottom paths in Fig. 4.4 implemented with ideal high-pass matrices $\overline{W}$ and $\overline{W}_s$, will also be identical.

For a practical implementation, the matrices M and M_s, or W and W_s, which are used to implement the resolving paths, will not typically be constructed from ideal filter impulse responses. In this case, the outputs y and y_s, or u and u_s, may differ in the fidelity with which they reflect the appropriate half-band frequency content of the input

data. In Section 4.4, error minimizing critieria for selecting the better resolving path were developed. For both Type I and Type II low-pass filters, the better path displayed the larger output norm, $\|Y(z)\|$ or $\|Y_s(z)\|$, for a given window of data. As mentioned previously, an equivalent statement may be made immediately for the norms of the time domain output vectors by virtue of Parseval's relationship. That is, the optimal low-pass path will produce an output vector with a larger norm than the alternate path.

The resolving path selection criteria could also have been developed directly for the time domain outputs by examining the decomposition in Eq. 4.54. Without engaging in detailed derivations of time-domain versions of all of the results in Section 4.4, consider, as an example, implementing the resolving paths with matrices M and M_s constructed from a Type I FIR filter. A filter impulse response would be selected so that, for instance, the even samples of the response were all zero except for a single non-zero sample of value c. Then the diagonal matrix D_e in the decomposed form of M would agree perfectly with the ideal diagonal matrix D_e in Eq. 4.68. For the same reasons discussed in Section 4.4, the filter used to construct the row space of the $\frac{n}{2} \times n$ matrix M will not usually be selected to have the "global," n-point support required for an ideal low-pass filter. In the case of a more localized, Type I impulse response, the matrix D_o will have diagonal elements with the same phase or argument as those of the matrix for the ideal filter in Eq. 4.68, but not the same constant magnitude c for every element. Ignoring pass-band ripple, most of the magnitudes will be nearly the same, but a few of the higher frequency elements will have diminished magnitudes corresponding to a transition region of non-zero width.

The assumption was made previously that the optimal resolving path is the one whose output most closely resembles the output of the appropriate ideal resolving path. In general, the norms of the outputs of the resolving paths when operated with practical Type I filters will be smaller than the ideal norm because of the diminished magnitudes of some of the elements in the implementation of D_o. By examining the decomposition in Eq. 4.53, notice that the norm $\|Mx\| = \|FMx\|$ reflects the contribution of the even samples of the input vector ideally, but diminishes the magnitude of the contribution of the odd samples. Again, because the phases of the elements of the two diagonal matrices D_e and D_o will be identical to those of the ideal matrices, a simple examination of the element magnitudes reveals how error will creep into the convolution in this case. For the alternate path operating with

matrix $M_s = MP$, the magnitude error in D_o will instead weight the contribution of the even samples of the input vector. Thus, if the magnitudes of the norms of the even and odd sample sequences in the input vector are not the same, the best approximation to the ideal result will come from the path in which the matrix D_e weights the contribution of whichever sample sequence (even or odd) has the larger norm. This increases the overall norm of the output, and therefore, the better path is the one with the larger output norm, as expected. If, for example, the sample sequences are identical, the two paths will deliver results of equal fidelity compared to the ideal outputs.

If the resolving path computations have been implemented with circular convolutions but non-periodic input data, there is a risk that end effects or wrap-around distortion will corrupt the comparison of the output norms and lead to an incorrect selection of optimal resolving path. This problem may be minimized by ensuring that the input data window is large compared to the filter impulse response support. The problem may be eliminated by windowing the output data to eliminate distorted output values before computing norms for comparison purposes.

Another interesting observation can be made by examining the matrices M and M_s, and the associated high-pass matrices W and W_s. For the Type I case, the better low-pass resolving path delivers an output with a larger norm. Alternatively, the better low-pass resolving path could also be found by finding the *complementary high-pass path with a larger norm.* For example, if the output of the low-pass path with matrix M yields a larger output norm than the path with M_s, then the high-pass path with matrix W also yields an output with a larger norm than the "cross" path with W_s.

For the Type II case, the better low-pass resolving path still delivers an output with a larger norm. However, the complementary high-pass path has an output with a *smaller* norm then the alternative, "cross" high-pass path. For example, if the output of the low-pass path with matrix M comprised of Type II impulse responses yields an output with a larger norm than the path with M_s, than the high-pass path with matrix W yields a *smaller* norm than the high-pass path with W_s.

The reason for this difference is the different structure of the two types of impulse responses. For the Type I case with, say, a five-tap filter impulse response, a row of the

low-pass matrix M_s might be:

$$[h[2]\ 0\ h[1]\ h[0]\ h[1]\ 0\ h[2]\ 0\ 0\ 0]$$

Departing from the previous convention of numbering the sequence values consecutively, the symmetries in the Type I response have been acknowledged in this row vector. In the same location in the complementary high-pass matrix W_s, the row is

$$[-h[2]\ 0\ -h[1]\ h[0]\ -h[1]\ 0\ -h[2]\ 0\ 0\ 0]$$

and in the "cross" high-pass matrix W, the row is

$$[0\ -h[2]\ 0\ -h[1]\ h[0]\ -h[1]\ 0\ -h[2]\ 0\ 0].$$

Because of the structural location of zeros and the symmetry in the odd sequence of the impulse response, the row of M_s is orthogonal to the row of W, and to all 2-point circular shifts of the row, *regardless of the values of the impulse response.* In general, the inner product of the row of M_s and of W_s will not be zero, that is, these rows are not orthogonal. Thus, the row space of W spans the orthogonal complement of the low-pass space spanned by the rows of M_s. Hence, minimizing the output norm of the *cross* high-pass path ensures, in the Type I case, that the input signal energy projected into the high-pass or "error" space is minimized.

For the Type II case, on the other hand, the orthogonal complement of the row space of, say, the matrix M is spanned by the rows of the complementary high-pass matrix W. A row of the matrix M in the Type II case with a four tap impulse response, for example, might be:

$$[h[0]\ h[1]\ h[1]\ h[0]\ 0\ 0\ 0\ 0]$$

The corresponding row of the complementary high-pass matrix W is

$$[h[0]\ -h[1]\ h[1]\ -h[0]\ 0\ 0\ 0\ 0]$$

The first row is orthogonal to all 2-point shifts of the second row, regardless of the sample values $h[0]$ and $h[1]$. So, in the Type II case, minimizing the norm or the energy of the output of a high-pass path ensures that the *complementary* low-pass path will be better than the "cross" low-pass path.

With ideal filters, the impulse response values are global in extent and are carefully selected so that the row space of $\overline{W}$ is the orthogonal complement of the row space of *both* $\overline{M}$ and $\overline{M}_s$. The rows of $\overline{W}_s$ *also* span the orthogonal complement of the row space of *both* $\overline{M}$ and $\overline{M}_s$. The norms of the outputs of both low-pass paths, or both high-pass paths, should therefore be identical, as is indeed the case.

One final observation will be made from Eq. 4.62. It has already been demonstrated that the ideal half-band filters produce 2 to 1 decimated circulant matrices with orthogonal rows, i.e., with matrices like $\overline{M}$ for which $\kappa(\overline{M}) = 1$. If it is desirable to construct other shift orthogonal filter impulse responses that lead to matrices with unity condition numbers, Eq. 4.62 reveals that a necessary condition on the polyphase components of the filter impulse response is that

$$|\hat{h}_e[i]|^2 + |\hat{h}_o[i]|^2 = a \tag{4.74}$$

where a is a constant. This celebrated condition was used by Daubechies ([32]), Mallat ([85]), and others ([143], [117], [131]) in various forms to produce and analyze families of compactly supported orthogonal bases, or FIR filter impulse responses. As mentioned previously, it has been proven that all but the most trivial of these orthogonal filters must have nonlinear phase. Also, Strang has shown in [134] that these orthogonal filters require approximately twice as many taps as a nonorthogonal filter to implement the same stop-band attenuation. For these reasons, shift orthogonal filter responses, which are valuable in perfect reconstruction coding and compression schemes, are less useful in a pattern recognition setting, where reconstruction is usually not an issue.

4.7.3 Discrete-Time, Discrete-Frequency Orthogonal Bases

This chapter closes with two final observations about discrete-time shift orthogonal functions. Meyer ([89]), Mallat ([85]), and others have observed that if the set of functions $\phi(x-k), k \in Z$, where Z is the set of all integers, forms an orthonormal basis for a subspace of square-integrable functions, the continuous-time Fourier transform $\hat{\phi}(\omega)$ of the function $\phi(x)$ will satisfy the relationship

$$\sum_{k=-\infty}^{\infty} |\hat{\phi}(\omega - 2k\pi)|^2 = 1. \tag{4.75}$$

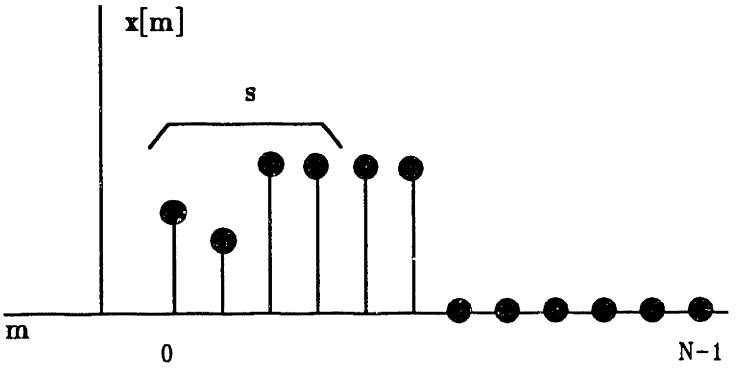


Figure 4.10: Sample Sequence

This observation has been useful in connecting different expressions for a DTFT form of Eq. 4.74 in [134] and [32]. Equation 4.75 has been used, for example, to identify certain constraints on orthogonal wavelets as identical to design constraints on quadrature mirror filters.

A DFT analogue of Eq. 4.75 will be proven in this section. This exercise will provide a useful tool for extending some of the results in the references to discrete-time, discrete-frequency situations, i.e., practical implementations using circulant transforms. Also, the proof of Eq. 4.75 is difficult to find in the relevant literature, and the proof for the DFT case will provide to the interested reader an outline for proving Eq. 4.75. This section will close with an application of this DFT analogue to the development of periodic, discrete-time orthonormal bases.

A sample discrete-time sequence $x[m]$ defined over a region of N points is shown in Fig. 4.10. The sequence has a DFT $\hat{X}[k]$. Suppose that the inner product of x and a circularly shifted copy of x, i.e., $x([m-sp])_N$, is such that

$$\sum_{m=0}^{N-1} x[m]x([m-sp])_N = \begin{cases} 1 & \text{if } p = 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.76}$$

where p is any integer, s is an integer smaller than N, and the notation $()_N$ indicates an

argument taken *modulo* N. For convenience, assume here that N is an integer multiple of the "shift interval" s. The following calculations will demonstrate that Eq. 4.76 will be true for a given sequence x if and only if

$$\sum_{p=0}^{\frac{N}{s}-1} |\hat{X}[k - \frac{N}{s}p]|^2 = 1 \tag{4.77}$$

for all k where the argument $k - \frac{N}{s}p$ is taken modulo N. The arguments of the sequences and transforms in the derivation which follows will henceforth be understood to be taken modulo N without explicit notation.

To begin, notice that because of the shift property of the DFT, the transform of $x[m - sp]$ may be written as

$$\mathcal{F}(x[m - sp]) = e^{\frac{-j2\pi ksp}{N}}\hat{X}[k] = e^{\frac{-j2\pi ksp}{N}}\hat{X}^*[-k]$$

where the last equality is possible because x is assumed to be a real sequence. The transform of $x[m]x[m - sp]$ is equivalent to the circular convolution of the transforms of $x[m]$ and $x[m - sp]$. So,

$$\mathcal{F}(x[m]x[m - sp]) = \sum_{m=0}^{N-1} x[m]x[m - sp]e^{\frac{-j2\pi km}{N}} = \sum_{l=0}^{N-1} \hat{X}[k - l]\hat{X}^*[-l]e^{\frac{-j2\pi lsp}{N}}. \tag{4.78}$$

Now, multiplying both sides of Eq. 4.78 by

$$\sum_{p=0}^{\frac{N}{s}-1} e^{\frac{-j2\pi k_o sp}{N}}$$

where k_o is an arbitrary integer yields

$$\sum_{p=0}^{\frac{N}{s}-1} e^{\frac{-j2\pi k_o sp}{N}} \sum_{m=0}^{N-1} x[m]x[m-sp]e^{\frac{-j2\pi km}{N}} = \sum_{p=0}^{\frac{N}{s}-1} e^{\frac{-j2\pi k_o sp}{N}} \sum_{l=0}^{N-1} \hat{X}[k-l]\hat{X}^*[-l]e^{\frac{-j2\pi lsp}{N}}. \tag{4.79}$$

Evaluating both sides of Eq. 4.79 at $k = 0$,

$$\sum_{p=0}^{\frac{N}{s}-1} e^{\frac{-j2\pi k_o sp}{N}} \underbrace{\sum_{m=0}^{N-1} x[m]x[m - sp]}_{a} = \sum_{p=0}^{\frac{N}{s}-1} e^{\frac{-j2\pi k_o sp}{N}} \sum_{l=0}^{N-1} \hat{X}[-l]\hat{X}^*[-l]e^{\frac{-j2\pi lsp}{N}}. \tag{4.80}$$

Now, if the sequence x is s-point shift orthonormal, Eq. 4.76 reveals that the term in Eq. 4.80 marked with an underbrace labeled a is unity for $p = 0$ and zero for all other p. Thus,

$$1 = \sum_{p=0}^{\frac{N}{s}-1} e^{\frac{-j2\pi k_o sp}{N}} \sum_{l=0}^{N-1} \hat{X}[-l]\hat{X}^*[-l]e^{\frac{-j2\pi lsp}{N}},$$

or, more compactly,

$$\sum_{p=0}^{\frac{N}{s}-1} \sum_{l=0}^{N-1} |\hat{X}[-l]|^2 e^{\frac{-j2\pi sp}{N}(k_o+l)} = 1 \tag{4.81}$$

Now, making a variable substitution $u = k_o + l$, Eq. 4.81 becomes

$$\sum_{p=0}^{\frac{N}{s}-1} \sum_{u=0}^{N-1} |\hat{X}[k_o - u]|^2 e^{\frac{-j2\pi usp}{N}} = 1. \tag{4.82}$$

Moving the summation over p inside the equation yields

$$\sum_{u=0}^{N-1} |\hat{X}[k_o - u]|^2 \underbrace{\sum_{p=0}^{\frac{N}{s}-1} e^{\frac{-j2\pi usp}{N}}}_{b} = 1. \tag{4.83}$$

The term in Eq. 4.83 marked with an underbrace labeled b is nonzero only when u is an integer multiple of $\frac{N}{s}$. Therefore,

$$\sum_{p=0}^{\frac{N}{s}-1} |\hat{X}[k_o - \frac{N}{s}p]|^2 = 1, \tag{4.84}$$

thus proving half of the assertion made at the beginning of this section. All of the steps in this derivation are reversible, i.e., the converse statement that Eq. 4.84 is a sufficient condition for Eq. 4.76 can be demonstrated. This is left to the interested reader. The transforms of the ideal half-band FIR filter impulse responses, Daubechies' compactly supported orthonormal wavelets, and the FIR quadrature mirror filters of Smith and Barnwell ([131]) and Vaidyanathan ([143]) will all obey Eq. 4.84.

One consequence of Eq. 4.84 is a technique for generating shift orthogonal bases for discrete-time subspaces of a finest resolution space or vector containing N points. Suppose that the $\frac{N}{s}$ sequences generated by s-point cyclically shifting a sequence $g[m]$ a total of $\frac{N}{s}$ times yield a *non-orthonormal* basis for a particular subspace. A shift *orthonormal* basis

sequence $\phi[m]$ for the same subspace may be expressed as a linear combination of the basis generated by the sequence $g[m]$:

$$\phi[m] = \sum_{i=0}^{\frac{N}{s}-1} \alpha[i] g[m-i]$$

where the $\alpha[i]$ are appropriate but as yet undetermined constants. The function $\hat{\phi}[k]$, the DFT of $\phi[m]$, may therefore be expressed as

$$\hat{\phi}[k] = \hat{M}[k]\hat{g}[k] \tag{4.85}$$

where $\hat{g}[k]$ is the DFT of $g[m]$ and

$$\hat{M}[k] = \sum_{n=0}^{\frac{N}{s}-1} \alpha[n] e^{\frac{-j2\pi kns}{N}}. \tag{4.86}$$

Since $\phi[m]$ is shift orthonormal,

$$\sum_{p=0}^{\frac{N}{s}-1} |\hat{\phi}[k - \frac{N}{s}p]|^2 = 1$$

or, substituting Eq. 4.85,

$$\sum_{p=0}^{\frac{N}{s}-1} |\hat{M}[k - \frac{N}{s}p]|^2 |\hat{g}[k - \frac{N}{s}p]|^2 = 1 \tag{4.87}$$

Equation 4.87 may be simplified by noting that

$$\hat{M}[k - \frac{N}{s}p] = \sum_{n=0}^{\frac{N}{s}-1} \alpha[n] e^{\frac{-j2\pi(k-\frac{N}{s}p)ns}{N}} = \sum_{n=0}^{\frac{N}{s}-1} \alpha[n] e^{\frac{-j2\pi kns}{N}} \underbrace{e^{j2\pi np}}_{c}$$

The term c is unity regardless of the value of the integers n and p. Recalling Eq. 4.86,

$$\hat{M}[k - \frac{N}{s}p] = \sum_{n=0}^{\frac{N}{s}-1} \alpha[n] e^{\frac{-j2\pi kns}{N}} = \hat{M}[k]. \tag{4.88}$$

Substituting Eq. 4.88 into Eq. 4.87,

$$|\hat{M}[k]|^2 \sum_{p=0}^{\frac{N}{s}-1} |\hat{g}[k - \frac{N}{s}p]|^2 = 1$$

or

$$|\hat{M}[k]|^2 = (\hat{M}[k])(\hat{M}[k])^* = \left[\sum_{p=0}^{\frac{N}{s}-1} |\hat{g}[k - \frac{N}{s}p]|^2\right]^{-1}.$$

The square root of the right-hand side of this equation is a real number, i.e., it equals its conjugate, so

$$\hat{M}[k] = \left[\sum_{p=0}^{\frac{N}{s}-1} |\hat{g}[k - \frac{N}{s}p]|^2\right]^{-\frac{1}{2}}. \tag{4.89}$$

Finally, substituting Eq. 4.89 into Eq. 4.85 yields an expression for the transform of the orthonormal basis,

$$\hat{\phi}[k] = \frac{\hat{g}[k]}{\left[\sum_{p=0}^{\frac{N}{s}-1} |\hat{g}[k - \frac{N}{s}p]|^2\right]^{\frac{1}{2}}}. \tag{4.90}$$

Equation 4.90 may be used to generate a (generally globally supported) orthonormal basis for a particular subspace from any non-orthonormal "template" basis function $g[m]$. This orthogonalization procedure has been shown to have certain symmetry preserving properties different from common methods like the Gram-Schmidt procedure ([132]); see [85] and [58], for example. Because this technique generally produces globally supported basis functions, it would not be useful for generating filter responses for application in the resolving path. It may, however, be useful in the discriminating path for certain pattern recognition techniques ([107], [141]).

Chapter 5

Prototype Construction and Performance

This chapter reviews the construction and testing of a prototype multiscale event detector. This detector is suitable for use in an advanced NILM, intended for application in demanding commercial and industrial settings. The v-section pattern recognition techniques introduced in Chapter 3 and the tree-structured search scheme developed in Chapter 4 for efficiently organizing the pattern search are the basis for the development of the prototype. In practice, the event detector developed here would be conjoined with the minimum information error correction and event matching algorithms reviewed and referenced in Chapter 1 and Appendix A, to complete the implementation of a new NILM with broader applicability and functionality than that of the residential monitor.

Extensive details of the hardware and software employed in the prototype, including schematics and code listings, are presented in Appendix D. The presentation of experimental results in this chapter is extensive and should assist in understanding the practical operation of the multiresolution event detector. The experiments conducted with the prototype are powerful demonstrations of the flexibility and potential of transient event detection for nonintrusive monitoring. It is important to remember that the hardware platform described in this chapter is expedient for quick implementation and testing. It is not necessarily an economical or compact commercial product. Based on experience with the current implementation, considerations for a dedicated hardware implementation are discussed later in this chapter.

5.1 Testing and Validation

Testing of the prototype event detection algorithm proceeded in three steps. The test program ranged from simulations entirely in software to a real-time implementation on an advanced digital signal processing system. Experiments with the early software platforms were instrumental in developing the results in Chapters 3 and 4. Later tests with the signal processing system and real electrical loads exposed the performance of the algorithm in a practical monitoring environment. The real-time implementation and testing will be the primary focus of this chapter, following a brief review of the complete, three step testing schedule.

Initial experimentation was conducted off-line in software on an IBM compatible personal computer. The software implementation is very similar to the *NILMscope* real-time program to be described shortly. These off-line tests challenged the algorithm with transient data generated by the RAPID simulator described in Appendix B. The flexibility of the RAPID simulator and the off-line monitoring software permitted testing and tuning of the v-section recognition algorithm and tree-structured decomposition in a relatively benign, noise-free environment.

In the second phase of testing, the real-time components of the software were ported to a Texas Instruments TMS320C30 digital signal processing (DSP) card installed in an Intel 80486-based, IBM compatible personal computer. The DSP card provides analog-to-digital conversion circuitry and remarkable real-time, floating point computational performance. The personal computer (PC) provides a convenient front-end with a graphical video display and inexpensive mass storage. All software for both the DSP card and the PC is written in the *C* programming language. In the second phase, the event detector was tested with the extended duration waveform synthesizer (EDWS) developed for this thesis.

The EDWS is described in detail in Appendix C. In brief, the EDWS creates analog signals from stored digital samples on four independent channels. A page-interleaved memory architecture employed in the EDWS permits the synthesizer to generate analog waveforms, from data stored on the hard disk of a personal computer, without interruption at a sample rate of 2000 Hertz per channel. The synthesizer will operate for long periods of time (over three hours with a 200 megabyte hard disk drive), limited only by the size of

the mass storage device providing digital samples.

The four channels of the EDWS were used to simulate signals from current transformers around a four wire utility service (three phases and a neutral wire) in an imaginary building. The sampled data for the EDWS was provided by RAPID. So, while experiments using the EDWS did not raise new concerns about the performance of the event detection algorithm, the EDWS did provide an important, controlled environment for initial testing of the hardware. Since the synthesizer can also be used to replay data recorded from a test site with a conventional storage oscilloscope or other portable data acquisition tool, the EDWS may in the future prove helpful for testing a prototype NILM *prior* to its installation at the site.

In the third phase of testing, which will be the primary focus of the remainder of this chapter, an analog preprocessor was constructed so that the DSP system could monitor real loads under actual operating conditions. The preprocessor contains current and voltage sensors configured to monitor the three phase currents and voltages on a 120 volt RMS AC line-to-neutral, three phase service. It also contains circuitry (described in more detail in the next section and in Appendix D) that computes estimates of the envelopes of real and reactive power, and also third harmonic content in the current waveform, on each of the three phases. The analog preprocessor, DSP card, and PC comprise the complete, real-time, prototype event detector. The third testing phase focused on conducting experiments with this detector monitoring actual loads.

The test stand constructed for experimentation with the event detector is shown schematically in Fig. 5.1. The box labeled `Multiscale Event Detector` in Fig. 5.1 represents the three components of the real-time prototype: the analog preprocessor, DSP card, and PC. The event detector monitors the voltage and current waveforms on a three phase electrical service that powers a collection of loads representative of important load classes in typical, medium to large size commercial and industrial buildings. The electrical hookup to the loads is routed through an electronically switched circuit breaker panel. [1] This panel can support a total of eight devices: six single phase loads and two three phase loads. A dedicated computer, labeled `PC` in Fig. 5.1, controls the operation of each load

[1]The author gratefully acknowledges the valuable assistance of Kurt Levens and Professor Les Norford in constructing the electronically switched circuit breaker panel.

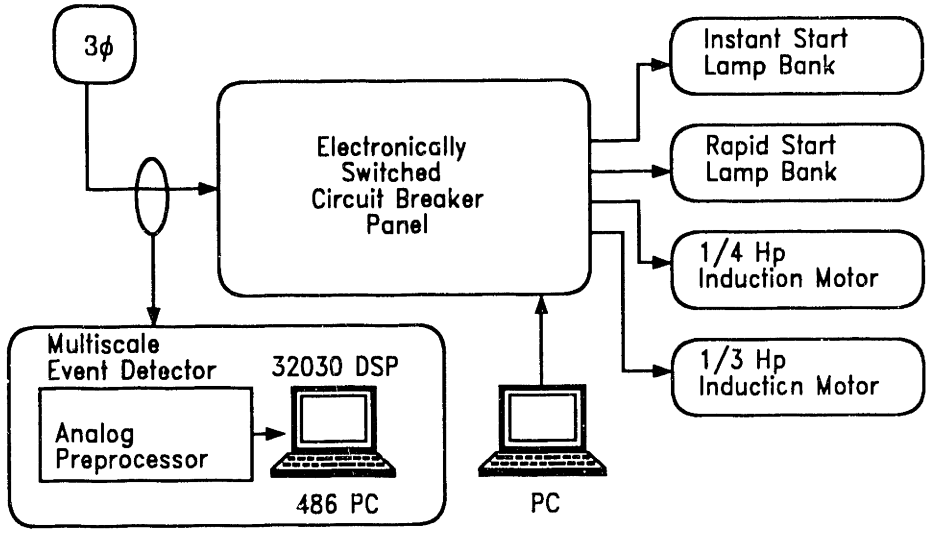


Figure 5.1: Hardware Test Facility

connected to the circuit breaker panel through a bank of relays. This personal computer runs software that can turn the loads on and off in any order, and with great flexibility in the relative timing of the turn–on and turn–off events. It can be programmed to activate the loads to simulate a variety of hypothetical end-use scenarios and – especially important for repeatable testing and demonstration – it can ensure that turn–on events for different loads will overlap, if desired.

The prototype event detector monitors the "utility service entry," i.e., the main three phase electrical service of this "mock" building. It is used to identify the turn–on time and type of the various loads. The event detector has, of course, **no** *a priori* knowledge of the operating schedule for the loads. Experiments with the test facility in Fig. 5.1 are not only easily verified, since the load scheduling is known independently of the event detector, but also are reasonably representative of conditions that might be found on part of the wiring harness of an actual building. Four loads, shown in Fig. 5.1, were selected for inclusion in the test facility: a bank of instant start fluorescent lamps, a bank of rapid start fluorescent lamps, a $\frac{1}{4}$ horsepower induction motor, and a $\frac{1}{3}$ horsepower induction motor. These loads were chosen because they represent load classes that account for a great deal of electric power consumption. The two different induction motors provide a set of loads

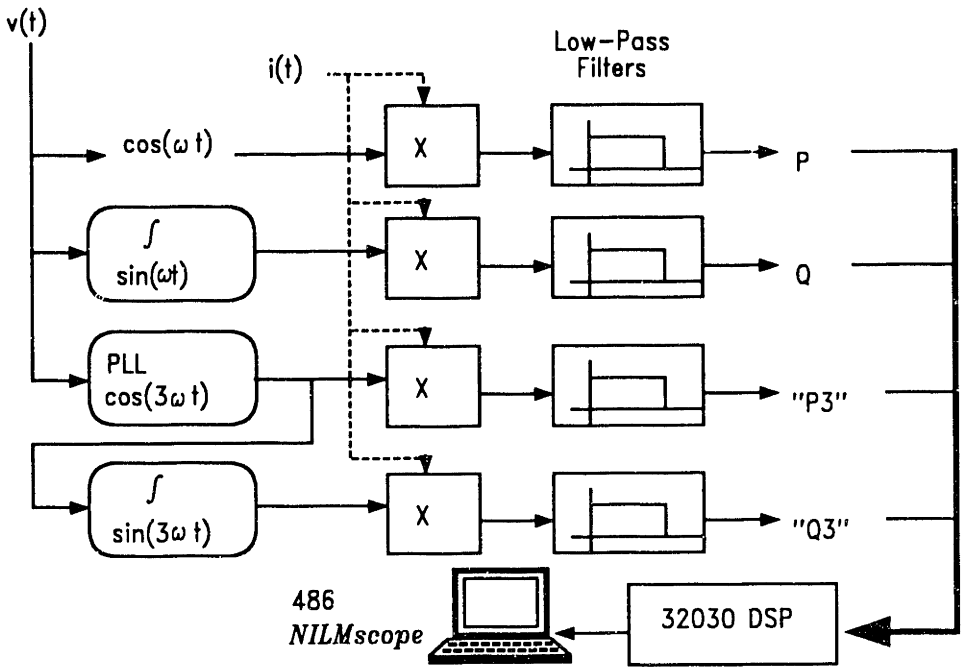


Figure 5.2: Prototype Event Detector

for testing multiscale aspects of the event detector. The loads are scaled relatively in power consumption to represent typical components in the lighting and ventilation systems of a medium size (between 10,000 and 100,000 square feet) commercial building ([97]).

5.2 Hardware Overview

This section provides an overview of the prototype event detection hardware. This review is primarily intended to aid understanding of the empirical results presented later in this chapter. Circuit schematics, source code listings, and other in-depth implementation details are included in Appendix D.

The multiscale event detector is shown in Fig. 5.2. Special emphasis has been placed on the analog preprocessor in the figure, since the next section reviews the detection algorithm that runs on the DSP system in more detail. Four channels of analog monitoring circuitry for only a single phase are shown in Fig. 5.2. These four channels compute estimates of the envelopes of real power, reactive power, in-phase third harmonic content in the current waveform, and quadrature third harmonic content in the current. The actual

prototype contains two additional copies of this circuitry for monitoring the voltage and current on the other two legs of the three phase service.

The variables v(t) and i(t) in Fig. 5.2 represent analog signals that are scaled and isolated versions of the actual voltage and current waveforms in a particular phase. These scaled and isolated waveforms are produced by Hall-effect voltage and current sensors that are not shown in the figure. The voltage is approximately a sinusoid at 60 Hertz line frequency, and is arbitrarily assumed to be a cosine wave, as shown in Fig. 5.2. The assumptions about frequency and phase are for convenience and do not impact on the generality of the techniques or results presented here. Looking across the top of the figure, the first channel in the analog preprocessor computes an estimate of the envelope of real power. A multiplier computes the instantaneous product of voltage and current, i.e., instantaneous power. The waveform of instantaneous power is filtered by a second order Butterworth low-pass filter with a cutoff frequency of 20 Hertz. This filter rejects the "carrier wave" 120 Hertz fundamental component of instantaneous power; i.e., it computes an approximate time average of the instantaneous power, or real power, represented by the variable P in Fig. 5.2. It also serves as an anti-aliasing filter so that the waveform P may be sampled accurately by the DSP system.

The second channel in the analog preprocessor computes an estimate of the envelope of reactive power. A sinusoid that lags the voltage waveform by 90 degrees is generated by integrating the voltage waveform. This quadrature sine wave is mixed with the current and low-pass filtered to compute Q, the estimate of reactive power. The third and fourth channels compute estimates of the third harmonic content in the current waveform. A phase-locked loop circuit, labeled PLL in Fig. 5.2, locks to the line frequency of the voltage waveform and generates a cosine wave at 180 Hertz, three times the frequency of the voltage waveform. This triplen frequency cosine is again mixed with the current waveform and the result is low-pass filtered to generate an envelope of the third harmonic content in phase with the voltage waveform. This envelope is labeled P3 in Fig. 5.2, where quotation marks have been included to emphasize that this envelope is not actually a variable of power in any conventional sense. Finally, the fourth channel integrates the 180 Hertz cosine wave to generate a 180 Hertz sine wave. This high frequency sine wave is mixed with the current and the product is low-pass filtered to generate an envelope of quadrature third harmonic

content in the current, labeled Q3 in the figure.

These techniques of integrating to generate quadrature waveforms and employing phase-locked loops to generate higher harmonics could be applied to compute envelopes of harmonic content for any frequency of interest, not just third harmonic. In Chapter 3, it was observed that it is generally advantageous to have as many informative channels of input data as possible, since this tends to increase the diversity of the v-section sets associated with different load classes. Third harmonic is a well-known component in the current waveforms of many loads: for example, computers containing power supplies that rectify the line voltage without power factor or harmonic content correction. Hence, third harmonic was chosen as the example for the analog preprocessor in this thesis. Other harmonics will probably be useful for identifying other load types, and will be clearly desirable in further field tests.

Note that the voltage and current waveforms could be sampled directly, and the envelopes of power and harmonic content could be computed by the DSP system. For these experiments, the TMS320C30 DSP system provided a surfeit of numerical performance. However, the experimental system proved to be limited in memory, and costly to expand. The decision was made, therefore, to relieve the DSP system of the burden of computing the envelopes, leaving extra memory free for the identification routines. Since some analog interface circuitry was necessary for the voltage and current sensors and anti-aliasing filters regardless of the chosen approach, the burden of implementing the envelope computations in analog circuitry was not excessive. For a commercial NILM, this trade-off would have to be examined carefully.

The four channels or envelope estimates for each of the three phases of the utility service can be sampled by the DSP system, as shown for the four channels of one phase in Fig. 5.2. In the current implementation, only two analog channels are available on the DSP system. The prototype event detector is currently configured to observe the envelopes of real and reactive power on a single phase that services all four loads in the test stand. The sample rate employed in the prototype is 180 Hertz on each channel. After sampling the input data streams, the DSP system scans for known transients with the *NILMscope* event detection software. An interactive component of the *NILMscope* software permits the DSP system to communicate any positive identifications or contacts to the 80486 PC. This

interface software presents results to the user through a convenient, graphical interface.

5.3 *NILMscope* Software

The event detection algorithm implemented in the *NILMscope* software demonstrates most of the key features outlined in Chapters 3 and 4. The software is implemented in two components. One component implements the event detection algorithm and is executed on the DSP system. The other is a user interface responsible for initializing the DSP system and for report generation. This second component is executed on the 80486 PC. To make some of the results presented in Chapters 3 and 4 more concrete, and to assist in appreciating the results presented in the next section, this section provides an overview of the event detection algorithm implemented in the *NILMscope* software, which is included in Appendix D.

A flow chart of the *NILMscope* event detection algorithm is shown in Fig. 5.3. This flow chart describes the analysis procedure conducted after the DSP system has acquired a window of samples from the input data streams. In the current implementation, a window contains 330 sample points, or approximately 1.83 seconds of data. For convenience in the presentation and analysis of results, the current implementation operates much like a digital storage oscilloscope. The user *arms* the system by activating a graphical button on the PC display. The DSP system samples the input data streams while waiting for a triggering event to occur on one of the channels. This event is any change in value between successive points that is greater than a user-specified value. When a triggering event is observed, the DSP system saves a window of data for analysis with the event detection algorithm.

With sufficient memory, the DSP system can execute the event recognition algorithm without halting its sampling, so that the search for events is conducted continuously. This would be essential in a practical NILM. In the test phase, however, a brief pause is helpful to provide time for the user to evaluate the analysis of each data window collected. The triggering software guarantees that this data window will contain interesting transient events. Of course, the window size has been selected to be sufficiently large to accomodate the largest transients of interest. The user may collect successive windows of data by arming the event detector after each window collected.

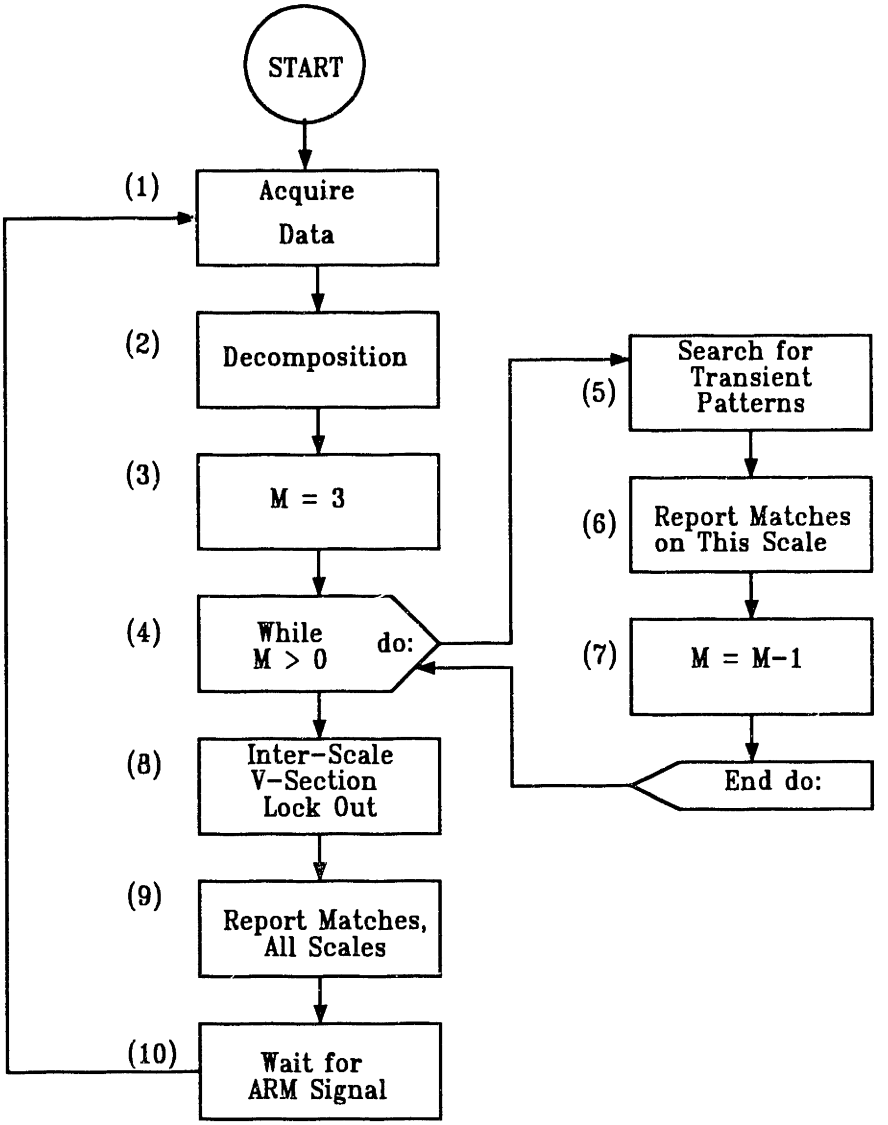


Figure 5.3: The *NILMscope* Event Detection Algorithm

Figure 5.3 summarizes the sequence of actions taken by the *NILMscope* software after the user has initialized the code on the PC, as indicated by the START bubble in the flow chart. Each action that is part of the data acquisition and recognition algorithm is numbered in the Fig. 5.3 flow chart. The remainder of this section will review each step in more detail. Each numbered item in the following list is a description whose number corresponds to a numbered step in the flow chart.

1. Data Acquisition: After the user initializes the *NILMscope* software on the PC, the DSP system is armed. During this data acquisition step, the DSP system waits for a triggering event. When such an event occurs, the DSP system saves 30 "pretrigger" points before the triggering event, acquires 300 new samples after the triggering event, and then suspends sampling to identify any known transient patterns in the data window using the event detection algorithm. The code for the data acquisition routines is contained in the source file *nilm* in Appendix D.

2. Tree-Structured Decomposition: Once a full window or vector of samples has been acquired, the DSP system performs a tree-structured decomposition on the data, *with adaptive resolving path selection*, as described in Chapter 4. In the current implementation, the input data for each coder step in the tree is computed before any pattern discrimination occurs at any scale step. The decomposition is conducted with sample rate changes of 2 to 1. A tree structure with a total of three coder or scale steps proved sufficient for identifying all of the transients associated with the loads in the test stand shown in Fig. 5.1. The code for the routines that perform the tree-structured decomposition is contained in the source file *nilm2* in Appendix D.

3. Set Scale Steps: Next, the DSP system searches at each scale for all three of the transient types that could appear. There are three scale steps, and a scale step variable M is initialized to the value of 3 to count down the search. When $M = 3$, the finest scale step, which corresponds to the initial vector of 330 sample points, will be searched for transient patterns. Later, the middle and coarse scales will be searched when $M = 2$ and $M = 1$, respectively.

4. Initiate Pattern Search: A loop in the program flow at this point in the *NILMscope*

software controls the search for patterns over all three scale steps. Transient templates for the three load classes included in the test stand were collected during a one-time training period prior to the start of on-line experimentation with the test stand and event detector. The three load classes represented in the test stand are rapid start fluorescent lamp banks, instant start fluorescent lamp banks, and induction motors. (The transient templates for these three classes will henceforth be referred to by the shorthand notation *Rapid*, *Instant*, and *Motor*, respectively.) The two induction motors in the test stand have significantly different rotor inertias. This difference causes the turn–on transient associated with the $\frac{1}{3}$ horsepower motor to last approximately four times as long as the transient of the $\frac{1}{4}$ horsepower motor. The long motor transient should therefore be identified at the coarse scale step, while the short motor transient should be identified at the fine scale step along with the two lamp banks. The load mix selected for the test stand therefore tests not only the v-section discrimination techniques on a single scale as outlined in Chapter 3, but also the application of these techniques in a multiscale setting with the tree-structured decomposition as outlined in Chapter 4.

The transient templates used during experimentation with the event detector are included in Appendix D. In order to emphasize the remarkable power of the v-section pattern recognition approach adopted here, very little "tuning" was performed to perfect the transient templates. During the training phase, the $\frac{1}{4}$ horsepower motor and the rapid start lamp bank were each turned on a single time. The DSP system was employed to capture the transients associated with these one-time turn–on events, to use as pattern templates. A change-of-mean detector implemented in MATLAB was used to segment these transient templates. The induction motor is represented by a time pattern of four v-sections, and the rapid start bank is represented by a set of three v-sections. *No data at all was collected for the larger induction motor.* The v-section set for the small motor was assumed to be representative of the v-section shapes for *both* motors, but on two different time scales and with larger amplitudes for the larger motor. Generating the template for the instant start bank required slightly more effort. The instant start bank is represented by a single v-section in

real power. This v-section exhibited some variation in shape depending on the precise turn–on time with respect to the line voltage waveform. Hence, a collection of 7 turn–on events was averaged to generate a representative v-section for the instant start bank.

5. Hierarchical Pattern Search with V-Section Lock Out on Scale M: During each pass through the pattern search loop, the DSP system searches for the v-sections associated with all three of the known transient events, *Motor*, *Rapid*, and *Instant*. Each iteration of the loop focuses on a single scale. The first pass through the loop, when $M = 3$, corresponds to a search on the finest scale. The pattern search is hierarchical, in that the DSP system searches for the most complex transient patterns first. Thus, the system searches first for the *Motor* pattern, with four v-sections, and last for the *Instant* pattern, with one v-section. When all of the v-sections for a pattern are found with both the shape and amplitude transversal filtering operations, the complete transient pattern is presumed to be present in the input data, and an event is recorded.

   To permit testing of the v-section lock out scheme described in Chapter 3, the numerical match range in the code presented in Appendix D is identical for all of the v-sections associated with each of the three patterns. The match range is wide enough for matching the v-section with the greatest possible observable variation about its template vector, in this case, the *Instant* v-section. Occasional false matches in v-sections for the more complex patterns due to the low discrimination tolerance will not cause erroneous event detections, because the entire v-section set must be identified for a positive event detection. In general, different match tolerances for each pattern and possibly several template patterns for problematic v-sections could be employed to further enhance the robustness of the event detector. Experimentation conducted with v-section specific match regions indicates that the adaptive resolving path selection employed in the tree-structured decomposition is instrumental in ensuring the tightest possible match region tolerances across multiple time scales.

   A v-section lock out is performed at each scale. If a complex pattern is found in the input data, the location of the v-sections of the pattern are recorded. The iden-

tification of any subsequent, less complex patterns will not be permitted based on the detection of v-sections at the previously recorded, "locked out" locations. As described in Chapter 3, the rationale for this lock out stems from the assumption that v-sections should not overlap. If a very complicated v-section set associated with a certain pattern is found in the data, the likelihood that the transient associated with this pattern is actually present in the data is relatively high. If a less complicated pattern is subsequently matched based on one or more v-sections that overlap with v-section locations of a previously discovered, more complicated pattern, the match for the less complicated pattern is presumed to be erroneous. In the current implementation, the lock out has been simplified to just the first v-sections of each of the three patterns. The remaining v-sections in each of the patterns were sufficiently different in shape that false v-section matches were not observed, and no lock out was necessary.

The source code for the pattern search routines is presented in the file *nilm3* in Appendix D.

6. Report Generation for Scale M: If all of the v-sections are found for a particular pattern, the transient pattern is presumed to be present in the data at the current scale M, and an event type and time is communicated to the interface component of the *NILMscope* software running on the PC.

7. Decrement Scale Step: The scale counter M is decremented, and the pattern detection loop is repeated until all remaining coarser scales have been searched for all three pattern types.

8. V-Section Lock Out Over All Scales: The PC component of the *NILMscope* software performs a final check of all matches found on all scales. The DSP system has already performed a hierarchical pattern search on each scale. This final check ensures that v-sections from a complex but coarse scale pattern were not used to match a less complicated, finer scale pattern. The source code for this check is in the file *lock* in Appendix D.

9. Final Report Generation: The PC component of the *NILMscope* software collates a

final report of the type and time of occurrence of all positive event detections; this report is recorded on a mass storage device and also to the graphical user interface for review.

10. Standby: After reporting any contacts, the PC waits for the user to issue an arming command. Once the system has been armed, the program flow returns to the first step in the flow chart. The DSP system waits for another triggering event to engage the complete data acquisition and event detection process again.

5.4 Review of Results

This section reviews ten experiments with the prototype event detector and the test stand illustrated in Fig. 5.1. The empirical results of these experiments are representative of several hundred similar trials conducted with the event detector. The tests reviewed in this section are not "hand picked" or representative of specially conducted or tuned tests. The performance of the prototype strongly supports the theoretical results presented in Chapters 3 and 4.

Figures 5.4 through 5.13 show screen prints from the PC running the *NILMscope* user interface software during ten of the experiments conducted with the test stand. The figures illustrate the performance of the prototype detector through a battery of progressively more complicated tests. To understand these figures, focus first on Fig. 5.4. Along the left-hand side of the display is a collection of video "buttons" that permit the user to control the operation of the event detector with a mouse connected to the PC. In particular, notice the button labeled `Arm`, which engages the event detector by starting the sequence of program steps shown in Fig. 5.3. Once a triggering event has occurred and windows of data have been collected on the input streams, the two graph windows labeled `Real Power, A` and `Reactive Power, A` are updated to display the captured data. The top graph shows an estimate of the envelope of real power in watts on one phase of the three phase service (arbitrarily labeled phase `A`). The bottom graph shows an estimate of the envelope of reactive power in VAr on the same phase. In Fig. 5.4, the two graph windows display the data collected during the turn–on transient of the instant start lamp bank in the test stand.

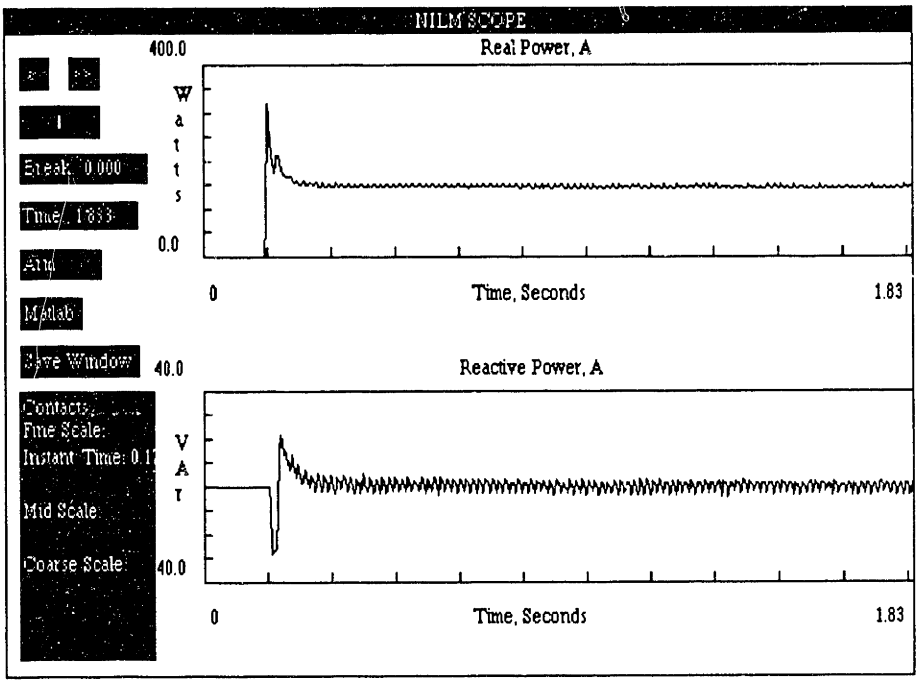


Figure 5.4: *NILMscope* Contact Report: *Instant*

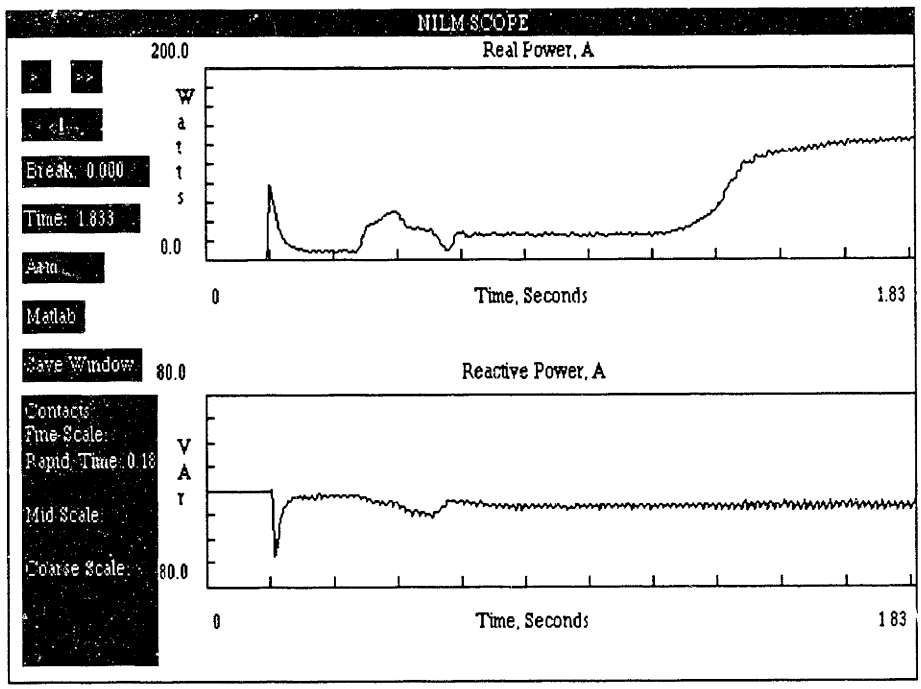


Figure 5.5: *NILMscope* Contact Report: *Rapid*

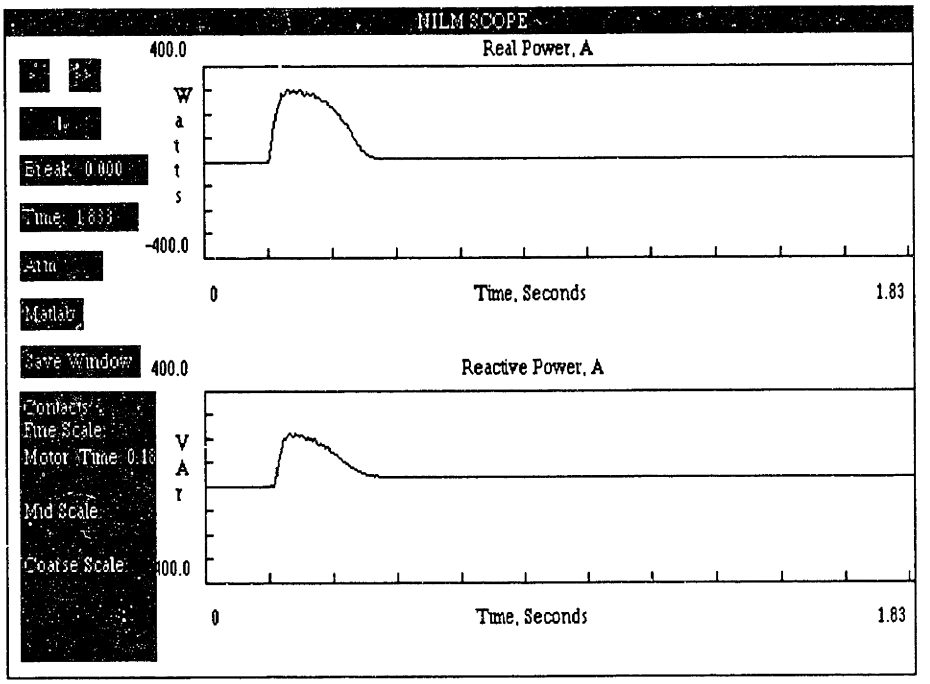


Figure 5.6: *NILMscope* Contact Report: *Motor*

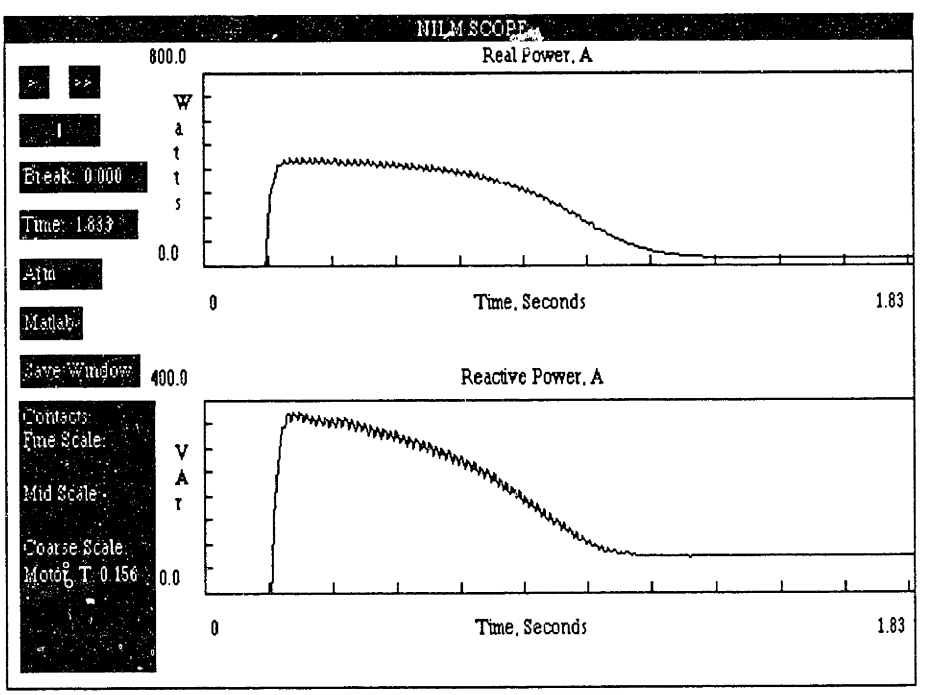


Figure 5.7: *NILMscope* Contact Report: *Motor*

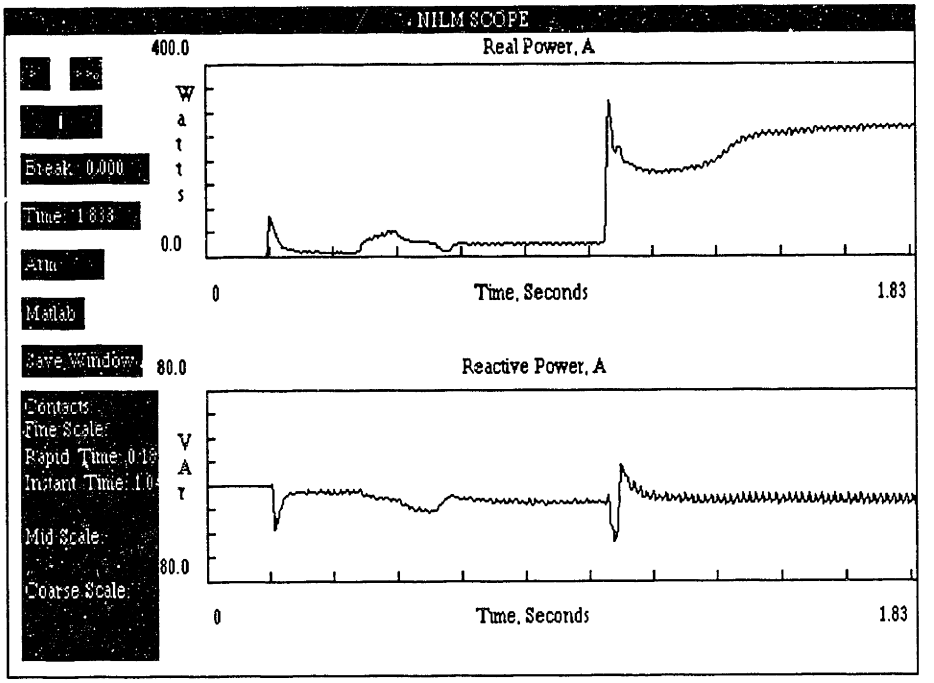


Figure 5.8: *NILMscope* Contact Report: *Rapid, Instant*

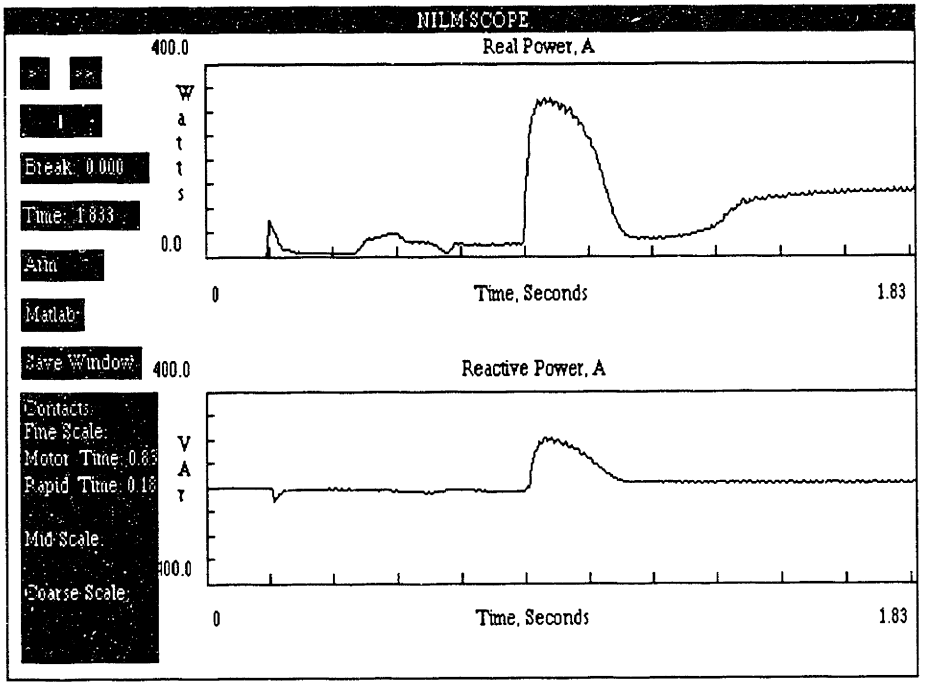


Figure 5.9: *NILMscope* Contact Report: *Rapid, Motor*

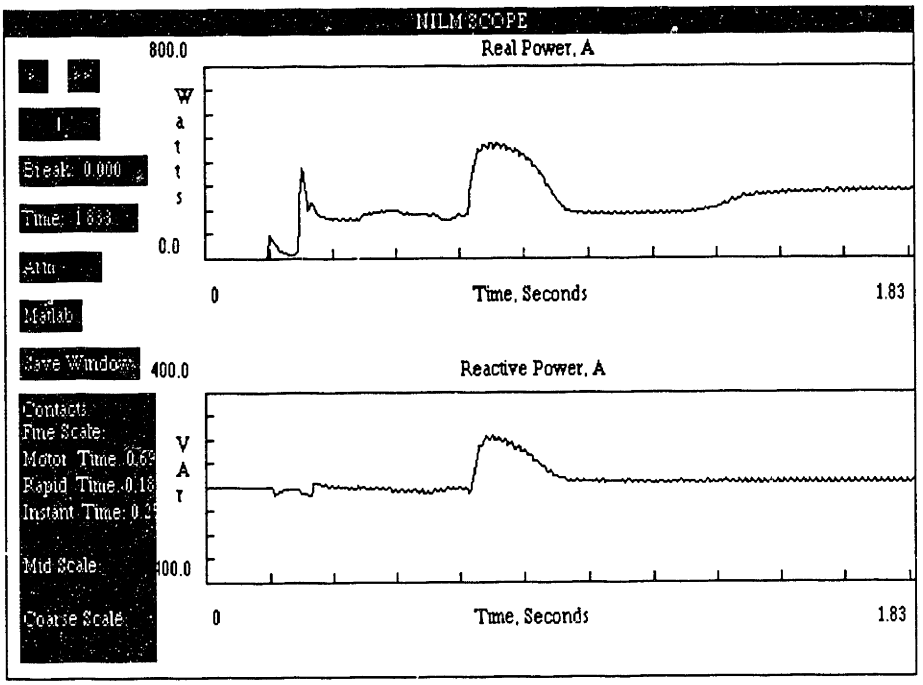


Figure 5.10: *NILMscope* Contact Report: *Rapid*, *Instant*, *Motor*

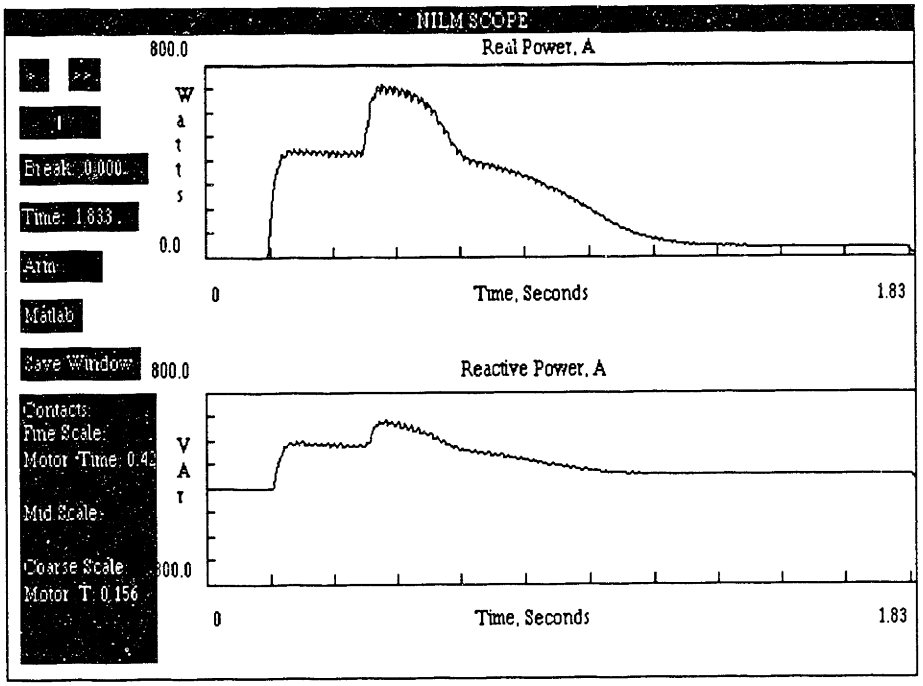


Figure 5.11: *NILMscope* Contact Report: Two *Motors*

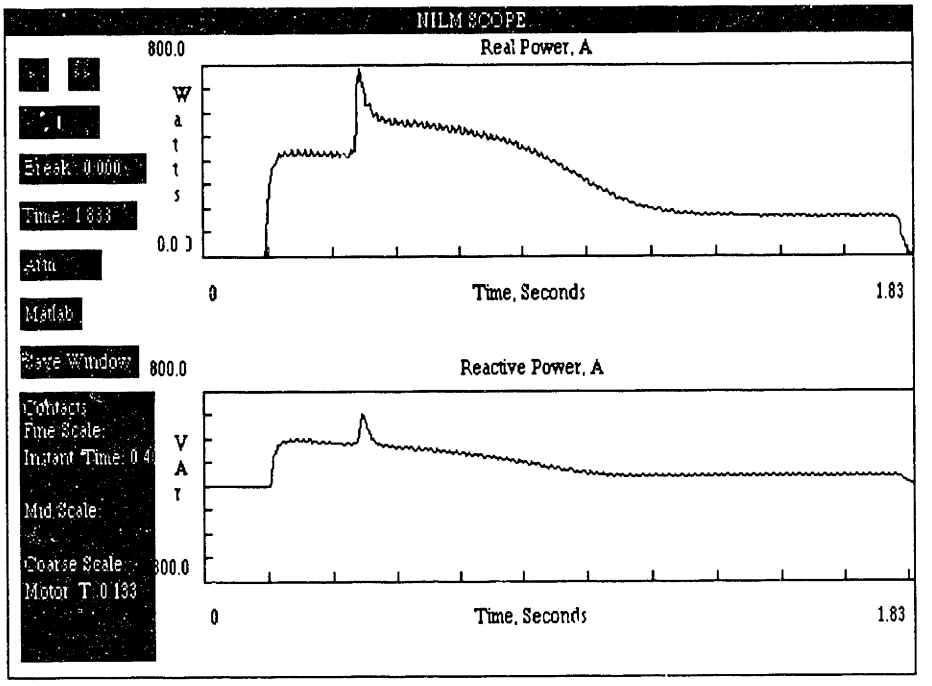


Figure 5.12: *NILMscope* Contact Report: *Motor, Instant*

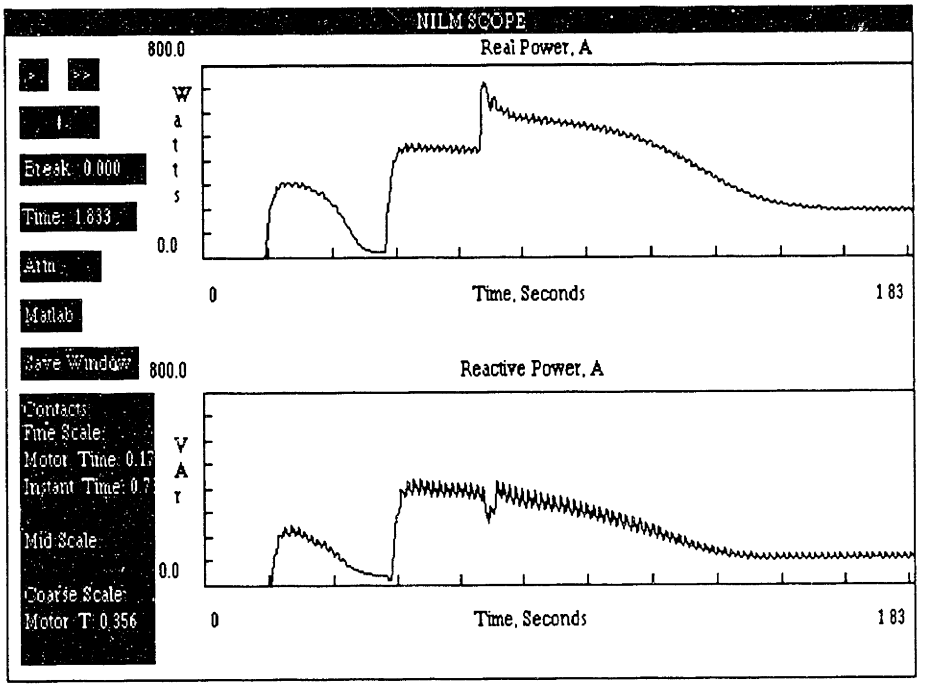


Figure 5.13: *NILMscope* Contact Report: Two *Motors, Instant*

In the lower left-hand corner of Fig. 5.4, the window labeled `Contacts` reports any transient events that the event detector has been able to identify in the course of examining the data displayed in the two graph windows. Events or contacts are reported by identity, time of occurrence, and scale. The identity of every recognizable, complete transient event found in the data windows is listed in the contact window. The load type responsible for the transient event is denoted by the designated shorthand notation, i.e., `Motor` for induction motors, `Rapid` for rapid start fluorescent lamp banks, and `Instant` for instant start fluorescent lamp banks. Following the load type responsible for each event listed in the contact window is the time of occurrence of the event. This time is listed in the form `Time` T, where T is the time of occurrence with respect to the origin of the two graph windows. The origins of the graph windows are arbitrarily assigned to be time zero. Finally, each event and time of occurrence is listed under the time scale on which it occurred.

There are three scales listed in the contact window: fine, mid, and coarse. Any event that was identified by a known v-section set at the initial, highest sampling rate, i.e., at the first coder stage in the tree-structured decomposition, will be listed directly under the heading `Fine Scale` in the contact window. By design, it is anticipated that events associated with the small motor and both lamp banks will be listed as fine scale events when they appear, assuming that the event detector functions properly. Events found at the next coder stage in the tree-structured decomposition conducted by the event detector will be listed immediately under the heading `Mid Scale` in the contact window. None of the loads in the test stand generate transient events that should appear under this heading. Immediately under the heading `Coarse Scale`, events found at the third and final coder stage in the decomposition will be listed. These events correspond to the longest duration or "stretched" versions of the finest scale transient event templates that can be identified by the prototype. Any events recognized by the properly working detector which are caused by the turn–on of the large induction motor should appear under this heading.

The first four tests shown in Figs. 5.4 through 5.7 record the performance of the prototype when challenged individually with the turn–on events of each of the four loads in the test stand. The graph windows in Fig. 5.4 show the envelopes of real and reactive power during the turn–on transient of the instant start lamp bank. The event has been properly identified in the contact window. Similarly, Fig. 5.5 shows the turn–on transient

of the rapid start lamp bank. Again, the event type, time of occurrence, and scale of occurrence have been correctly identified by the prototype event detector. Figures 5.6 and 5.7 emphasize the multiscale nature of the event detection scheme. Figure 5.6 shows the turn–on transient event associated with the small induction motor. Under the heading `Fine Scale`, the prototype has correctly identified the transient event type as `Motor` in the contact window. Figure 5.7 shows the turn–on event for the large motor. Again, the prototype has correctly identified the event type as `Motor`, but this time on the coarse scale, as anticipated.

Recall that the template for the turn–on transient of event type `Motor` was generated from a single example of the small motor *only*. Nevertheless, the event detector correctly classified both the small and large motors. This lends further credence to the notion that certain classes of loads are well represented by a single transient shape scaled in amplitude and time.

Carefully note the transient shapes of each of the four load transients shown in Figs. 5.4 through 5.7. Referring to these shapes may be helpful when examining the following experiments with overlapping transient events.

Figures 5.8 and 5.9 record two instances of two loads turning on such that the transient events overlap. In both cases, no key v-sections overlap with each other. In the example shown in Fig. 5.8, the rapid start bank turned on, followed by the turn–on of the instant start bank. Both events are properly identified in the contact window. In Fig. 5.9, the rapid start bank again turned on first, this time followed by the small induction motor. The contact window correctly contains the record of both events at the finest scale.

Figure 5.10 shows an example where three loads turn on so that all three transient events overlap. Again, no key v-sections overlap with each other. First, the rapid start bank turns on, closely followed by the turn–on transient of the instant start bank, and then the small induction motor. All three events are correctly recorded at the finest time scale in the contact window.

The last three figures show overlapping transient events that occur on different time scales. In Fig. 5.11, the large induction motor turns on, followed by the small induction motor. The small induction motor finishes its turn–on transient before the large motor finishes. No key v-sections overlap with each other, and both events are correctly identified

at the appropriate time scales in the contact window. In Fig. 5.12, the large motor turns on, followed by the instant start lamp bank. In the final experiment shown in Fig. 5.13, the small induction motor turns on and completes its transient, followed by the turn–on transient of the large induction motor and the instant start lamp bank. All of the events are correctly identified at the appropriate time scales.

5.5 Summary

The examples reviewed in the previous section are representative of a larger set of experiments conducted with the test stand. Provided the assumptions made in the development of the event detection algorithm are satisfied, the prototype detector performs remarkably well. This performance is perhaps more impressive in light of the fact that fairly little effort was made to "tune" the detector for the loads in the test stand. The templates for the small motor and rapid start bank were developed from a single transient capture during a one-time "walk through" of the test stand. The large motor was consistently identified only with data taken from the small motor. Only the template for the instant start bank required special consideration, i.e., the averaging of several turn–on transients to develop a generally representative prototype template.

The traces for real and reactive power in Fig. 5.4 for the instant start turn–on transient suggest that this load could be represented by two v-sections, one in real power and one in reactive power. The reactive power envelope exhibited annoying variation, however, over successive events. Also, for test purposes, it was desired to have a very sparse v-section set to seriously challenge the prototype detector. For both of these reasons, the decision was made to represent the instant start bank with only a single v-section in real power. The sparsity of this pattern resulted in an occasional false match with part of the large induction motor pattern. *This never resulted in an incorrect final event contact report, however, because the v-section lock out recognized and eliminated the false match.* So, the complete event detection algorithm always produced reliable final contact reports.

In practice, the robustness of the detector could be enhanced by working with more complicated v-section pattern sets for each load. For example, including the reactive power v-section for the instant start bank would tend to eliminate false matches. Variation could

be handled by working with more than one transversal filter template for this particular v-section. A match from any candidate v-section shape over the range of possible shapes would constitute a match for the v-section in the reactive power data stream. Searching more data streams, such as the third harmonic streams already generated by the analog preprocessor, would result, in general, in more complicated v-section pattern sets for many loads of interest. Experience with the prototype supports the assumption made in Chapter 3 that employing more complicated sets of interesting v-sections for each significant load tends to enhance the reliability of the event detection algorithm.

Searching for more v-sections – motivated by the desire for more complicated patterns for each significant load, or the desire to monitor more loads, or both – will require more computational capability. The commercial DSP system employed in the prototype was selected precisely for its high numerical performance and flexible development tools. It is not necessarily the most cost effective solution, however. Fortunately, the pattern search loop in the event detection algorithm, represented by the steps labeled four through seven in Fig. 5.3, can be conducted in parallel. Specifically, the search for each v-section could be conducted independently by a relatively low cost processor. Furthermore, the search for each v-section on each scale could also be conducted independently. This strongly suggests that a commercial NILM based on the event detection algorithm developed in this thesis could be based on a parallel processing architecture composed of inexpensive microprocessors or microcontrollers. The performance of the event detector in such a NILM would always be easily expandable for more challenging monitoring sites by simply adding more processors to search for more v-sections, and possibly over more time scales.

Chapter 6

Conclusions

Summary

The need for reliable load usage statistics has become increasingly apparent as demand and the introduction of new load classes, with potentially detrimental impact on the quality of the utility service, have increased practically unchecked. The desire for accurate load usage data is not limited to electric utilities. Commercial and industrial consumers, seeking to schedule energy consumption during economical billing times, have a growing appetite for load usage information. Conventional load monitoring schemes require separate metering equipment or connections for every load of interest. The costs of purchasing this equipment, installing it at a target site, and collating the collected data have traditionally been limiting factors in the acquisition of usage data. The residential nonintrusive load monitor developed at MIT was a remarkable first step toward reducing the cost and time required to collect and summarize usage data from relatively uncomplicated target sites.

The purpose of this thesis has been to propose methods that extend the applicability of the nonintrusive approach. A load survey was conducted which revealed that common commercial and industrial techniques, which homogenize steady state load behavior in order to minimize costs and local degradation of the utility service, failed in general to mask telltale turn–on transient signatures. These signatures appear to be reliable bases for associating patterns observed at or near the utility service entry with their propagating loads. The mathematical tools necessary to implement a multiscale transient event detector have been developed and presented. These tools permit the detector to work even in environ-

ments with very high rates of event generation and where transient signatures may overlap to some extent. The ability of the detector to tolerate high rates of event generation increases its "nonintrusiveness," i.e., the maximum allowable distance on the wiring harness between the detector and the loads of interest.

A prototype event detector based on these tools was constructed and demonstrated. This prototype tracked actual loads in a realistic operating environment. It performed successfully even when challenged with multiple, overlapping transient signatures on different scales.

Directions for Further Work

In practice, an advanced NILM would be constructed by conjoining the multiscale transient event detection techniques with the syntactic pattern recognition and error correct algorithms that were developed for the residential monitor and which were reviewed in the first chapter. The pattern recognition tools introduced in this thesis could be applied in graded steps to develop a spectrum of monitoring capabilities and costs. Most loads in a residential environment, for example, exhibit transient behavior that spans comparatively few scales. The full multiscale search scheme would not be essential in the residential environment. While the residential monitor has operated successfully with a simple event detector, its flexibility and power could be enhanced by incorporating the v-section signature recognition techniques described in Chapter 3. For busier target sites with less differentiation in steady state load behavior but a rich diversity of transient signatures, the complete multiscale search technique, incorporating tools outlined in Chapters 3 and 4 and demonstrated in Chapter 5, would be necessary to provide reliable, efficient event detection.

For the NILM to be able to bound the operating times of any particular load, the turn–off transients of individual loads would have to be sufficiently diverse to permit reliable association of turn–on events with turn–off events. In general, turn–off transient signatures are much less complicated than turn–on signatures. For the complete determination of the operating schedules of different loads, some differentiation between the steady state power levels of different loads must therefore be present, as in the case of the residential monitor.

Strictly from a load monitoring standpoint, the transient event detector for nonintru-

sive monitoring provides two primary, novel benefits. First, the multiscale event detector drastically reduces or altogether eliminates the need for the service entry to settle to a quiet steady state level before an event is generated. Second, the detector provides immediate identification of the load class responsible for the transient event. If the event detector is "preloaded" with the signatures of important loads, it may even be possible to install an advanced NILM and begin detailed monitoring, including load identification and operating schedule determination, without ever examining the loads installed at the target site.

One important focus for future work, therefore, will no doubt be a detailed load survey of different types of sites of potential interest for monitoring. A more thorough understanding of the variation of load transient signatures will suggest the best approach for configuring the NILM for extended monitoring of different types of target sites. Some sites may require a one-time training period when the NILM is installed. For many sites and load classes, it may be possible to develop a complete set of transient templates from a signature library and simulations like those provided by RAPID.

This is particularly true if the monitoring objectives are relatively limited. For example, one application that has been suggested during the course of this experimentation would be to use the NILM to track the energy consumption of lighting loads, especially fluorescent lamp banks. The transient signatures for most fluorescent lamp banks, at least within a subclass such as rapid start banks, tend to be very similar and very repeatable. It seems likely that a dedicated NILM could be constructed to monitor such light banks from a library of known transient signatures. Sufficient commonalities and differences in the transients for rapid start banks, for example, have been observed so that banks of lamps with iron core ballasts or with integrated circuit ballasts could be tracked together or separately for many manufacturers' ballasts, without any on-site training period. As suggested in the previous chapter, parallel hardware implementations and "minimally configured" or dedicated hardware architectures for specific monitoring applications are interesting and important avenues for future research.

The availability of a transient event detector opens the door to many other related monitoring applications. In Chapter 2 it was observed that the ongoing quest for efficiency and the introduction of new load classes has led to a proliferation of loads that incorporate solid state power supplies or power conditioning utility interfaces. Important examples

include personal computers, energy efficient lighting, and variable speed drives in ventilating systems. Poorly designed or inexpensive power electronic interfaces may introduce undesired line current harmonics, which adversely affect the utility voltage waveform seen by all of the loads in the neighborhood of the offending interface.

Currently, a power quality investigation of a load or class of loads at a target site usually involves hooking each suspect load to an expensive, dedicated spectrum analyzer. A NILM with a transient event detector is uniquely capable of reducing the expense of examining the power quality at a site. The NILM hardware platform is certainly capable of performing a spectral analysis of line current harmonics. By associating changes in harmonic content at the service entry with the activation of different loads, the NILM is uniquely situated for identifying power quality "offenders."

For facilities managers, the advanced NILM offers a potentially inexpensive source of information for exercising control over the electrical loading in a building. The NILM could make available detailed, almost instantaneous knowledge of the power demand and impact on power quality of individual loads. This information could, for example, be used to reschedule the operating time of different loads or alter the location of different loads to balance a local network or to respond quickly to constraints imposed by utility spot pricing schemes. Power quality offenders could be identified and operated at times and locations that minimize the impact of undesired line harmonic content.

Finally, an advanced NILM with a transient event detector may be a remarkably valuable platform for conducting nonintrusive diagnostic evaluations of critical loads in a building. Fascinating results are reported in [147] and [148] in which multirate estimators are used to determine the state space parameters of an induction motor strictly from measurements made at the electrical terminals. It may be possible to extend this technique to other types of rotating electric machinery, and other loads. By tracking trends in parameters over time, it may be possible to make some statements concerning the health of these loads. The ability to presage the need for maintenance could make the NILM invaluable in many commercial and industrial settings.

The tools and results in this thesis provide a strong foundation for the implementation of a versatile and widely applicable commercial nonintrusive load monitor.

Bibliography

[1] R.E. Abbot and S.C. Hadden, "Requirements for an Advanced Utility Load Monitoring System," EPRI Final Report CU–6623, December 1989.

[2] I. Aleksander, *Artificial Vision for Robots*, Chapman & Hall, 1983.

[3] R. Ansari, "Satisfying the Haar Condition in Halfband FIR Filter Design," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 36, No. 1, January 1988, pp. 123–124.

[4] G.R. Arce and N.C. Gallagher, Jr.,"BTC Image Coding Using Median Filter Roots," *IEEE Transactions on Communications*, June 1983, pp. 784–793.

[5] G.R. Arce, N.C. Gallagher, and T.A. Nodes, "Median Filters: Theory for One– and Two–dimensional filters," *Advances in Computer Vision and Image Processing*, JAI Press, Greenwich, CN, 1986.

[6] G.R. Arce and P.J. Warter, "A Median Filter Architecture Suitable for VLSI Implementation," *23rd. Allerton Conf. on Commun., Cont., and Comput.*, October 1984, pp. 172–181.

[7] G.R. Arce, P.J. Warter, R.E. Foster, "Theory and VLSI Implementation of Multilevel Median Filters," *International Symposium on Circuits and Systems*, June 1988.

[8] A.R. Barron and R.L. Barron, "Statistical Learning Networks: A Unifying View," *Computing Science and Statistics: 1988 Proceedings of the 20th Symposium on the Interface*, 1988, pp. 192–203.

[9] R.L. Barron et. al., "Applications of Polynomial Neural Networks to FDIE and Reconfigurable Flight Control," National Aerospace Electronics Conference, May 1990.

[10] R.L. Barron and D.W. Abbott, "Use of Polynomial Networks in Optimum, Real–Time, Two–Point Boundary–Value Guidance of Tactical Weapons," Military Computing Conference, May 1988.

[11] M. Basseville and A. Benveniste, *Detection of Abrupt Changes in Signals and Dynamical Systems*, Springer–Verlag, 1980.

[12] M.J. Bastiaans, *Gabor's Expansion of a Signal into Gaussian Elementary Signals, Proceedings of the IEEE*, Vol. 68, No. 4, April 1980, pp. 538–540.

[13] B. Boashash and P. O'Shea, "A Methodology for Detection and Classification of Some Underwater Acoustic Signals Using Time–Frequency Analysis Techniques," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 38, No. 11, November 1990, pp. 1829–1841.

[14] A. Bouloutas, G. Hart, and M. Schwartz, "Extending the Viterbi Algorithm to Correct Strings Generated by a Finite State Machine," preprint, NSF Grant # CDR 88–11111, CAT Grant # CU01125801–006, and EPRI Grant # 8000–32, 1990.

[15] A. Bouloutas, G. Hart, and M. Schwartz, "Identification of a Finite State Machine Using Unreliable Partially Observed Data Sequences," preprint, NSF Grant # CDR 88–11111, CAT Grant # CU01125801–006, and EPRI Grant # 8000–32, 1990.

[16] J. Bryant, "A Fast Classifier for Image Data," *Pattern Recognition*, Vol. 22, No. 1, 1989, pp. 45–58.

[17] P. Burt and E. Adelson, "The Laplacian Pyramid as a Compact Image Code," *IEEE Transactions on Communication*, Vol. 31, 1983, pp. 482–540.

[18] S.L. Campbell and C.D. Meyer, Jr., *Generalized Inverses of Linear Transformations*, Dover Publications, 1979.

[19] J.F. Canny, "Finding Edges and Lines in Images," Artificial Intelligence Laboratory, Technical Report TR.720, June 1983.

[20] C.S. Carman and M.B. Merickel, "Supervising Isodata with an Information Theoretic Stopping Rule," *Pattern Recognition*, Vol. 23, No. 1/2, 1990, pp. 185–197.

[21] G.A. Carpenter and S. Grossberg, "ART 2: Self–Organization of Stable Category Recognition Codes for Analog Input Patterns," *Applied Optics*, Vol. 26, No. 23, December 1987, pp. 4919–4930.

[22] M. Chen and P. Yan, "A Multiscaling Approach Based on Morphological Filtering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 7, July 1989, pp. 694–700.

[23] E.W. Cheney, *Introduction to Approximation Theory*, McGraw–Hill, 1966.

[24] D.G. Childers and M. Pao, "Complex Demodulation for Transient Wavelet Detection and Extraction," *IEEE Transaction on Audio and Electroacoustics*, Vol. AU–20, No. 4, October 1972, pp. 295–308.

[25] C.K. Chui, *An Introduction to Wavelets*, Academic Press, 1992.

[26] J.J. Clark, "Authenticating Edges Produced by Zero–Crossing Algorithms," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 1, January 1989, pp. 43–57.

[27] R. Courant and D. Hilbert, *Methods of Mathematical Physics*, Interscience Publishers, 1953.

[28] E.J. Coyle, "The Theory and VLSI Implementation of Stack Filters," *VLSI Signal Processing*, IEEE Press, New York, NY, 1986.

[29] R.E. Crochiere and L.R. Rabiner, *Multirate Digital Signal Processing*, Prentice–Hall, 1983.

[30] R.E. Crochiere and L.R. Rabiner, "Optimum FIR Digital Filter Implementations for Decimation, Interpolation, and Narrow–Band Filtering," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP–23, No. 5, October 1975, pp. 444–456.

[31] A.W. Crooke and J.W. Craig, "Digital Filters for Sample–Rate Reduction," *IEEE Transactions on Audio and Electroacoustics*, Vol. AU–20, No. 4, October 1972, pp. 308–315.

[32] I. Daubechies, "Orthonormal Bases of Compactly Supported Wavelets," *Communications on Pure and Applied Mathematics*, Vol. XLI, 1988, pp. 909–996.

[33] W.B. Davenport, Jr., and W.L. Root, *An Introduction to the Theory of Random Signals and Noise*, IEEE Press, 1987.

[34] M. Dolson, "The Phase Vocoder: A Tutorial," *Computer Music Journal*, Vol. 10, No. 4, Winter 1986, pp. 14–27.

[35] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, John–Wiley and Sons, 1973.

[36] C. Fahn, J. Wang, and J. Lee, "An Adaptive Reduction Procedure for the Piecewise Linear Approximation of Digitized Curves," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 9, September 1989, pp. 967–973.

[37] A.E. Fitzgerald, C.K. Kingsley, Jr., and S.D. Umans, *Electric Machinery*, McGraw–Hill, 1983.

[38] B. Friedlander and B. Porat, *Detection of Transient Signals by the Gabor Representation*, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP–37, No. 2, February 1989, pp. 169–180.

[39] B. Friedlander and B. Porat, *Performance Analysis of Transient Detectors Based on Linear Data Transforms*, for U.S. Army Missile Command, Contract No. DAAH01–90–C–0521, sponsored by the Defense Advanced Research Projects Agency, 28 February 1991.

[40] M. Frisch and H. Messer, "Detection of a Transient Signal of Unknown Scaling and Arrival Time Using the Discrete Wavelet Transform," *ICASSP*, 1991, pp. 1313–1316.

[41] K.S. Fu, *Sequential Methods in Pattern Recognition and Machine Learning*, Academic Press, 1968.

[42] K.S. Fu, *Syntactic Methods in Pattern Recognition*, Academic Press, 1974.

[43] K.S. Fu, *Syntactic Pattern Recognition and Applications*, Prentice–Hall, 1982.

[44] K. Fukushima, "Neural Network Model for Selective Attention in Visual Pattern Recognition and Associateive Recall," *Applied Optics*, Vol. 26, No. 23, 1 December 1987, pp. 4985–4930.

[45] S. A Golden, *Identifying Multiscale Statistical Models Using the Wavelet Transform*, Center for Intelligent Control Systems, CICS–TH–294, April 1991.

[46] L. Goldfarb, "A Unified Approach to Pattern Recognition," *Pattern Recognition*, Vol. 17, No. 5, 1984, pp. 575–582.

[47] D.B. Goldgof, T.S. Huang, and H. Lee, "A Curvature–Based Approach to Terrain Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 11, November 1989, pp. 1213–1217.

[48] G.H. Golub and C.F. Van Loan, *Matrix Computations*, 2nd ed., The Johns Hopkins University Press, 1989.

[49] D.J. Goodman and M.J. Carey, "Nine Digital Filters for Decimation and Interpolation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP–25, No. 2, April 1977, pp. 121–126.

[50] J. Goutsias and J.M. Mendel, "Simultaneous Optimal Segmentation and Model Estimation of Nonstationary Noisy Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 9, September 1989, pp. 990–998.

[51] A. Grossman and J. Morlet, "Decomposition of Hardy Functions into Square Integrable Wavelets of Constant Shape," *SIAM J. Math. Anal.*, Vol. 15, no. 4, July 1984, pp. 723–736.

[52] A. Grossman, "Wavelet Transforms and Edge Detection," in *Stochastic Processes in Physics and Engineering*, 1988, pp. 149–157.

[53] R.W. Hamming, *Digital Filters*, Prentice–Hall, 1989.

[54] G.W. Hart, "Identification of Multi–State Appliances," MIT LEES Technical Report TR–87–012, July 1987.

[55] G.W. Hart, "A Method for Estimating the Operating History of a Known Appliance Given Slightly Imperfect Data," MIT LEES Progress Report, EPRI Contract RP2568-2, August 1987.

[56] G.W. Hart, "Minimum Information Estimation of Structure," M.I.T. Ph.D. Dissertation, Department of Electrical Engineering and Computer Science, June 1987.

[57] G.W. Hart, "Prototype Nonintrusive Appliance Load Monitor," MIT LEES Progress Report, EPRI Contract RP2568-2, September 1985.

[58] C.E. Heil and D.F. Walnut, "Continuous and Discrete Wavelet Transforms," *SIAM Review*, Vol. 31, No. 4, December 1989, pp. 628–666.

[59] H.D. Helms, "Digital Filters with Equiripple or Minimax Responses," *IEEE Transactions on Audio and Electroacoustics*, Vol Au–19, No. 1, March 1971, pp. 87–93.

[60] R.D. Hippenstiel and P.M. De Oliveira, "Time–Varying Spectral Estimation Using the Instantaneous Power Spectrum (IPS)", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 38, No. 10, October 1990, pp. 1752–1759.

[61] F. Hlawatsch and G.F. Boudreaux-Bartels, "Linear and Quadratic Time-Frequency Signal Representations," *IEEE Signal Processing Magazine*, April 1992, pp. 21–67.

[62] E.B. Hogenauer, "An Economical Class of Digital Filters for Decimation and Interpolation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol ASSP–29, No. 2, April 1981, pp. 155–162.

[63] P.A. Humblet, *Design of Optical Matches Filters*, Center for Intelligent Control Systems, CICS–P–302, August 1991.

[64] F. Itakura, "Minimum Prediction Residual Principle Applied to Speech Recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol ASSP–23, February 1975, pp. 67–72.

[65] L.B. Jackson, *Digital Filters and Signal Processing*, Kluwer Academic Publishers, 1989.

[66] A.K. Jain and R.C. Dubes, *Algorithms for Clustering Data*, Prentice–Hall, 1988.

[67] W.C. Karl, S.B. Leeb, L.A. Jones, J.L. Kirtley, and G.C. Verghese, "Applications of a Class of Nonlinear Filters to Problems in Power Electronics," *IEEE Power Electronics Specialists Conference*, June 1990, pp. 35–42.

[68] B. Kartikeyan and A. Sarkar, "Shape Description by Time Series," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 9, September 1989, pp. 977–984.

[69] M.J. Katz, *Templets and the Explanation of Complex Patterns*, Cambridge University Press, 1986.

[70] A. Kendric, "Implementation of Fast Wavelet Transform Multiscale Edge Identification Algorithm," S.B. Thesis, MIT Department of Electrical Engineering and Computer Science, May 1992.

[71] J.L. Kirtley, "Polyphase Networks," M.I.T. 6.061 Supplementary Notes 3, June 1988.

[72] T. Kohonen, *Content–Addressable Memories*, Springer–Verlag, 1980.

[73] P.C. Krause, *Analysis of Electric Machinery*, McGraw–Hill, 1986.

[74] P.C. Krause and C.H. Thomas, "Simulation of Symmetrical Induction Machinery," *IEEE Transactions Power Apparatus and Systems*, Vol. 84, November 1965, pp. 1038–1053.

[75] A. Kusko, *Solid-State DC Motor Drives*, The MIT Press, 1979.

[76] C.F. Lam and D. Kamins, "Signature Recognition Through Spectral Analysis," *Pattern Recognition*, Vol. 22, No. 1, 1989, pp. 39–44.

[77] R.S. Ledley, "High–Speed Automatic Analysis of Biomedical Pictures," *Science*, Vol 146, 1964, p. 216.

[78] Y.H. Lee, S. Ko, and A.T. Fam, "Efficient Impulsive Noise Suppression Via Nonlinear Recursive Filtering," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 37, No. 2, February 1989, pp. 303–306.

[79] S.B. Leeb, A. Ortiz, and J.L. Kirtley, Jr., "Real-Time Median Filtering with a Fast Hardware Sorter," *IEEE APEC Conference Record*, March 1991, pp. 254–260.

[80] S.B. Leeb, J.L. Kirtley, Jr., D.S. Woodruff, and M.S. LeVan, "Building–Level Power Network Analysis," *IEEE Computer Applications in Power*, Vol. 5, No. 1, January 1992, pp. 30–34.

[81] S.B. Leeb, J.L. Kirtley, G.C. Verghese, L.K. Norford, and R.D. Tabors, "Nonintrusive Load Monitoring: Status Report," MIT LEES, 11 May 1992.

[82] J.J. Liang and V. Clarson, "A New Approach to Classification of Brainwaves," *Pattern Recognition*, Vol. 22, No. 6, 1989, pp. 767–774.

[83] M.S. Longmire, A.F. Milton, and E.H. Takken, "Simulation of Mid–Infrared Clutter Rejection, Part 1: One–dimensional LMS Spatial Filter and Adaptive Threshold Algorithms," *Applied Optics*, Vol. 21, No. 21, November 1982, pp. 3819–3833.

[84] S.G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 7, July 1989.

[85] S.G. Mallat, "Multiresolution Approximations and Wavelet Orthonormal Bases of $L^2(R)$," *Transactions of the American Mathematical Society*, Vol. 315, No. 1, September 1989.

[86] P. Maragos and R.W. Schafer, "Morphological Filters – Part II: Their Relations to Median, Order-Statistic, and Stack Filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, August 1987, pp. 1170–1184.

[87] P.A. Maragos and R.W. Schafer, "Morphological Skeleton Representation and Coding of Binary Images," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, October 1986, pp. 1228–1244.

[88] H.G. Martinez and T.W. Parks, "A Class of Infinite–Duration Impulse Response Digital Filters for Sampling Rate Reduction," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP–27, No. 2, April 1979, pp. 154–162.

[89] Y. Meyer, "Orthonormal Wavelets," *Wavelets: Time–Frequency Methods and Phase Space*, Proceedings of the International Conference, Marseille, France, December 14–18, 1987, Springer–Verlag, pp. 21–37.

[90] F. Milinazzo, C. Zala, and I. Barrodale, "On the Rate of Growth of Condition Numbers for Convolution Matrices," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP–35, No. 4, April 1987, pp. 471–475.

[91] M. Minsky and S. Papert, *Perceptron: An Introduction to Computational Geometry*, The MIT Press, 1969.

[92] B.A. Miwa, D.M. Otten, and M.F. Schlecht, "High Efficiency Power Factor Correction Using Interleaving Techniques," *Proceedings of the APEC*, February 1992.

[93] G.V. Moustakides and S. Theodoridis, "Fast Newton Transversal Filters – A New Class of Adaptive Estimation Algorithms," *IEEE Transactions on Signal Processing*, Vol. 39, No. 10, October 1991, pp. 2184–2193.

[94] A.N. Mucciardi and E.E. Gose, "An Automatic Clustering Algorithm and Its Properties in High–Dimmensional Spaces," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC–2, No. 2, April 1972, pp. 247–254.

[95] A.N. Mucciardi, "Self–Organizing Probability State Variable Parameter Search Algorithms for Systems that Must Avoid High–Penalty Operating Regions," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC–4, No. 4, July 1974, pp. 350–362.

[96] National Fire Protection Association, *National Electrical Code*, 1990.

[97] L. Norford, P. Hinteregger, R. Little, and N. Mabey, "Application of Nonintrusive Load Monitoring to Commercial Buildings," Final Report to Johnson Controls, Inc., December 1992.

[98] G. Oetken, T.W. Parks, and H.W. Schussler, "New Results in the Design of Digital Interpolators," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP–23, No. 3, June 1975, pp. 301–309.

[99] K. Oflazer, "Design and Implementation of a Single-Chip 1–D Median Filter," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, October 1983, pp. 1164–1168.

[100] A.V. Oppenheim, ed., *Applications of Digital Signal Processing*, Prentice–Hall, 1978.

[101] A.V. Oppenheim, A.S. Willsky, and I.T. Young, *Signals and Systems*, Prentice–Hall, 1983.

[102] A.V. Oppenheim and R.W. Schafer, *Digital Signal Processing*, Prentice–Hall, 1975.

[103] A.V. Oppenheim and R.W. Schafer, *Discrete–Time Signal Processing*, Prentice–Hall, 1989.

[104] A. Ortiz, "Flash Filter: Median Filtering by Insertion Sort," S.B. Thesis, MIT Department of Electrical Engineering and Computer Science, May 1990.

[105] T.W. Parks and J.H. McClellan, "Chebyshev Approximation for Nonrecursive Digital Filters with Linear Phase," *IEEE Transactions on Circuit Theory*, Vol CT–19, No. 2, March 1972, pp. 189–194.

[106] T.W. Parks and D.P. Kolba, "Interpolation Minimizing Maximum Normalized Error for Band–Limited Signals," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP–26, No. 4, August 1978, pp. 381–384.

[107] E.A. Patrick, *Fundamentals of Pattern Recognition*, Prentice–Hall, 1972.

[108] Y. Pao, *Adaptive Pattern Recognition and Neural Networks*, Addison–Wesley, 1989.

[109] I. Pitas and A.N. Venetsanopoulos, *Nonlinear Digital Filters – Principles and Applications*, Kluwer Academic Publishers, Boston, 1990.

[110] *Power Pollution Caused by Fluorescent Lighting Fixtures*, Power Quality Laboratory Techbrief, California Polytechnic, San Luis Obispo, Vol 1., No. 12, September 1990.

[111] *Programs for Digital Signal Processing*, Digital Signal Processing Committee (eds.), IEEE Acoustics, Speech, and Signal Processing Society, IEEE Press, 1979.

[112] T.F. Quatieri and R.J. McAulay, "Speech Transformations Based on Sinusoidal Representation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP–34, No. 6, December 1986, pp. 1449–1464.

[113] L.R. Rabiner, "Linear Program Design of Finite Impulse Response (FIR) Digital Filters," *IEEE Transactions on Audio and Electroacoustics*, Vol AU–20, No. 4, October 1972, pp. 280–288.

[114] L.R. Rabiner, J.H. McClellan, and T.W. Parks, "FIR Digital Filter Design Techniques Using Weighted Chebyshev Approximation," *Proceedings of the IEEE*, Vol. 63, No. 4, April 1975, pp. 595–610.

[115] S. Ranganath, "Image Filtering Using Multiresolution Representations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 13, No. 5, May 1991, pp. 426–439.

[116] D.S. Richards, "VLSI Median Filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, January 1990, pp. 145–153.

[117] O. Rioul and M. Vetterli, "Wavelets and Signal Processing," *IEEE SP Magazine*, October 1991, pp. 14–38.

[118] T.J. Rivlin, *An Introduction to the Approximation of Functions*, Dover, 1969.

[119] W. Roecker, "The Application of Digital Filters for Moving Target Indication," *IEEE Transactions on Audio and Electroacoustics*, Vol. AU–19, No. 1, March 1971, pp. 72–77.

[120] D.W. Rorabacher, "Efficient FIR Filter Design for Sample Rate Reduction or Interpolation," *IEEE Proceedings '75 ISCAS*, 1975, pp. 396–399.

[121] M. Ruskai, et. al., *Wavelets and Their Applications*, Jones and Bartlett, 1992.

[122] D.C. Saha and G.P. Rao, *Identification of Continuous Dynamical Systems*, Springer–Verlag, 1983.

[123] A.C. Sanderson and A.K.C. Wong, "Pattern Trajectory Analysis of Nonstationary Multivariate Data," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC–10, No. 7, July 1990, pp. 384–392.

[124] T. Saramaki, "A New Class of Linear–Phase FIR Filters for Decimation, Interpolation, and Narrow–Band Filtering," *IEEE Int. Symp. Circuits Syst.*, May 10–12, 1982, pp. 801–811.

[125] G.N. Saridis and R.F. Hofstadter, "A Pattern Recognition Approach to the Classification of Nonlinear Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC–4, No. 4, July 1974, pp. 362–371.

[126] J.A. Scheuer, "Digital Waveform Synthesizer," M.I.T. S.B. Thesis, Department of Electrical Engineering and Computer Science, May 1991.

[127] W.A.C. Schmidt, "Modified Matched Filter for Cloud Clutter Suppression," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, No. 6, June 1990, pp. 594–600.

[128] W.M. Siebert, *Circuits, Signals, and Systems*, McGraw–Hill, 1986.

[129] G.R. Slemon, "Electric Machines and Drives," Addison–Wesley, 1992.

[130] G.R. Slemon, "Magnetoelectric Devices: Transducers, Transformers, and Machines," John–Wiley and Sons, 1966.

[131] M.J.T. Smith and T.P. Barnwell, III, "Exact Reconstruction Techniques for Tree-Structured Subband Coders," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP–34, No. 3, June 1986, pp. 434–441.

[132] G. Strang, *Introduction to Applied Mathematics*, Wellesley–Cambridge Press, 1986.

[133] G. Strang, *Linear Algebra and Its Applications*, Academic Press, 1980.

[134] G. Strang, "The Optimal Coefficients in Daubechies Wavelets," Center for Intelligent Control Systems Technical Report CICS–P–308, October, 1991.

[135] G. Strang, "Wavelets and Dilation Equations: A Brief Introduction," *SIAM Review*, Vol. 31, No. 4, December 1989, pp. 614–627.

[136] F. Sultanem, "Using Appliance Signatures for Monitoring Residential Loads at Meter Panel Level," *IEEE/PES 1991 Winter Meeting*, paper no. 91WM016–6PWRD, February 3, 1991.

[137] R. Tabors, et. al., "Residential Nonintrusive Appliance Load Monitor," EPRI final report, to appear.

[138] J.D. Taft and N.K. Bose, "Quadratic–Linear Filters for Signal Detection," *IEEE Transaction on Signal Processing*, Vol. 39, No. 11, November 1991, pp. 2557–2559.

[139] E.H. Takken, D. Friedman, A.F. Milton, and R. Nitzberg, "Least–Mean–Square Spatial Filter for IR Sensors," *Applied Optics*, Vol. 18, No. 24, 15 December 1979, pp. 4210–4222.

[140] T. Taxt, P.J. Flynn, and A.K. Jain, "Segmentation of Document Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 12, December 1989, pp. 1322–1329.

[141] J.T. Tou and R.C. Gonzalez, *Pattern Recognition Principles*, Addison–Wesley, 1974.

[142] P.P. Vaidyanathan, "Lattice Structures for Optimal Design and Robust Implementation of Two–Channel Perfect–Reconstruction QMF Banks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 36, No. 1, January 1988, pp. 81–94.

[143] P.P. Vaidyanathan, "Multirate Digital Filters, Filter Banks, Polyphase Networks, and Applications: A Tutorial," *Proceedings of the IEEE*, Vol. 78, No. 1, January 1990, pp. 56–93.

[144] P.P. Vaidyanathan, "Quadrature Mirror Filter Banks, M–Band Extensions and Perfect–Reconstruction Techniques," *IEEE ASSP Magazine*, July 1987, pp. 4–20.

[145] P.P. Vaidyanathan, "Theory and Design of M–Channel Maximally Decimated Quadrature Mirror Filters with Arbitrary M, Having the Perfect–Reconstruction Property," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP–35, No. 4, April 1987, pp. 476–492.

[146] B. Van Der Pol and H. Bremmer, *Operational Calculus*, Cambridge University Press, 1950.

[147] M. Velez–Reyes, "Speed and Parameter Estimation for Induction Machines," M.I.T. S.M. Thesis, Department of Electrical Engineering and Computer Science, May 1988.

[148] M. Velez–Reyes, K. Minami, and G.C. Verghese, "Recursive Speed and Parameter Estimation for Induction Machines," *Proceedings of IEEE Industry Applications Society Annual Meeting*, 1989, pp. 607–611.

[149] R.R. Verderber and O. Morse, "Performance of Electronic Ballasts and Lighting Controllers with 34-W Fluorescent Lamps," EPRI Final Report EM-5888, June 1988.

[150] G.N. Wassel and J. Sklansky, "Training a One–Dimensional Classifier to Minimize the Probability of Error," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC–2, No. 4, September 1972, pp. 533–541.

[151] P.D. Wasserman, *Neural Computing*, Van Nostrand Reinhold, 1989.

[152] J.F. Waymouth, *Electric Discharge Lamps*, The MIT Press, 1971.

[153] P. Whittle, *Prediction and Regulation*, University of Minnesota Press, 1983.

[154] B. Wilkenson, "Power Factor Correction and IEC 555-2," *Powertechnics Magazine*, February 1991, pp. 20–24.

[155] J.L. Wyatt and M. Ilic, "Time–Domain Reactive Power Concepts for Nonlinear, Nonsinusoidal, or Nonperiodic Networks," *ISCAS*, 1990, pp. 387–390.

[156] W. Zettler, J. Huffman, and D.C.P. Linden, "Application of Compactly Supported Wavelets to Image Compression," *SPIE Image Procssing Algorithms and Techniques*, Vol. 1244, 1990, pp. 150–160.

Appendix A

Multi-State Event Correction Algorithm

The code in this appendix implements a version of the multistate event correction algorithm introduced in [55]. The algorithm is reviewed in some detail in Chapter 1. This appendix does not contain the support code which implements the user interface for the event correction algorithm. Only the routines necessary to the implementation of the algorithm itself have been included. This program may be compiled with the version 5.1, Microsoft *C* compiler. It may be executed on an IBM PC compatible personal computer.

A.1 Event Correction Algorithm: *multi*

This header file declares global variables and structures used in the main body of the event correction code.

```
#include <stdio.h>
#include <graph.h>
#include <window.h1>
#include <math.h>
#include <stdlib.h>
#include <dos.h>

#define CURS 219
#define SPACE 32
#define NUL
#define MAXSTATE 25
#define MAXARC 40
#define NORM 7
#define HLIGHT 15
#define ID 1
#define INV 0

#ifdef MAIN
```

```
#define EXTERN NUL
#else
#define EXTERN extern
#endif

EXTERN int sc;
EXTERN int mousepresent;
EXTERN int u,v;

typedef struct {int num_states,num_arcs,u,v,state_tab[MAXSTATE+1][MAXSTATE+1];
                int id_tab[MAXSTATE+1][MAXARC+1], coord[MAXSTATE+1];
                char state_name[MAXSTATE+1][20],fsm_name[20];} FSM, *PFSM;

typedef struct {float arc_val[MAXARC+1];} ARC, *PARC;
```

A.2 Event Correction Algorithm: *mfsm*

The source code which follows implements the multistate event correction algorithm.

```
#define MAIN 1
#include <multi.h>

main()
{
  int nevs,fnevs;
  int i,j,k;
  char filename[30];
  float evs[1000][2],fixevs[1000][2];
  double pan[401], pbn[401], pcn[401],val;
  FSM  workf;
  PFSM workfsm;
  ARC  worka;
  PARC workarc;
  FILE *pf,*fopen();


  workfsm = &workf;
  workarc = &worka;
  strcpy(filename,"input");

  /* load the fsm that describes the appliance of interest */
  init_fsm(workfsm,workarc);
  edit_fsm(workfsm,workarc);
  pack_fsm(workfsm,workarc);
  gen_idtab(workfsm,workarc);

  /* load event data from mass storage */
```

```
  read_event_history(&nevs,evs,filename);
  /* find anomalous event sequences */
  segment_events(nevs,evs,workfsm,workarc,workfsm->u);
  /* repair the event stream */
  fix_events(workfsm,workarc,evs,nevs,fixevs,&fnevs);
  /* determine the state history for the corrected event stream */
  segment_events(fnevs,fixevs,workfsm,workarc,workfsm->u);

  /* save the corrected state and event streams */
  /* plot original and corrected streams        */
  j = 4;
  for (i = 0; i <= 400; i++) {
          pan[i] = pbn[i] = pcn[i] = 0.0;
  }
  k = 0;
  for (i = 1; i <= 100; i++) {
          pcn[k] = (double) fixevs[i][0];
          pbn[k] = (double) evs[i][0] + 0.2;
          k += j;
  }
  pf = fopen("output","w");
  fprintf (pf,"event  state\n");
  for (i = 1; i <= fnevs; i++) {
        fprintf (pf,"%g %g\n",fixevs[i][0],fixevs[i][1]);
  }
  fclose(pf);
  _setvideomode(_ERESCOLOR);
  clear();
  i = 400;
  val = 0.0;
  plotit(pan,pbn,pcn,i,val,(double) fnevs,"EVENT PLOT","EVENT"," ");
  j = 4;
  for (i = 0; i <= 400; i++) {
          pan[i] = pbn[i] = pcn[i] = 0.0;
  }
  k = 0;
  for (i = workfsm->u; i <= 100+workfsm->u; i++) {
          pcn[k] = (double) fixevs[i][1];
          pbn[k] = (double) evs[i][1] + 0.2;
          k += j;
  }
  i = 400;
  val = 0.0;
  clear();
  plotit(pan,pbn,pcn,i,val,(double) fnevs,"STATE PLOT","STATE"," ");
  clear();
  _setvideomode(_TEXTC80);
}
```

```
fix_events(workfsm,workarc,evs,nevs,fixevs,fn)
int nevs,*fn;
float evs[1000][2],fixevs[1000][2];
PFSM workfsm;
PARC workarc;
{
  int i,j,k,l,pa,len,flag,reps,offset,noff,fnevs;
  int seq_tab[MAXARC][MAXARC],rep_tab[MAXARC][MAXARC];

  reps = 0;
  for (i = 1; i <= nevs; i++) {
          if (evs[i][1] == 0) {
                  j = i;
                  flag = 0;
                  while(flag == 0) {
                        j++;
                        if (evs[j][1] != 0) flag = 1;
                  }
                  len = j-i+1;
                  pa = 0;
                  fill_seq_tab(workfsm,workarc,seq_tab,
                               (int) evs[j][1],len+2,&pa);
                  k = pick_best(evs,seq_tab,len,pa,i-1,j,workfsm->u);
                  rep_tab[reps][0] = seq_tab[k][0] - workfsm->u;
                  rep_tab[reps][1] = i; /* start of zero seq in evs */
                  rep_tab[reps][2] = j; /* loc of next nonz in the seq     */
                  for (l = 0; l < rep_tab[reps][0]; l++)
                     rep_tab[reps][3+l] = seq_tab[k][rep_tab[reps][0]-l];
                  reps++;
                  i = j;
          }
  }
  i = 0;
  fnevs = nevs;
  offset = 0;
  for (j = 1; j <= nevs; j++) {
                  if (j != rep_tab[i][1])
                          fixevs[j+offset][0] = evs[j][0];
                  else {
                          len = rep_tab[i][2] - rep_tab[i][1] + 1;
                          for (k = j+offset; k <= j+offset+len; k++)
                            fixevs[k][0] = (float) rep_tab[i][k-j-offset+3];
                          offset += (rep_tab[i][0] - len);
                          j += len-1;
                          fnevs += (rep_tab[i][0] - len);
                          if (i < reps) i++;
                  }
  }
```

```
  *fn = fnevs;
}

pick_best(evs,seq_tab,len,paths,start,stop,u)
float evs[1000][2];
int seq_tab[MAXARC][MAXARC];
int len,paths,start,stop,u;
{
  int i, j, l, ok[MAXARC], minsofar;
  int tlen, plen, trailer[40], phrase[40], temphrase[40], temp;

  tlen = 0;
  while(tlen <= u) {
          tlen++;
          trailer[tlen] = (int) evs[start-tlen+1][0];
  }
  plen = 0;
  while(plen <= u+1) {
          plen++;
          temphrase[plen] = (int) evs[start+plen][0];
  }
  for (i = 1; i <= plen; i++) phrase[i] = temphrase[plen-i+1];
  for (i = 1; i <= paths; i++) {
          ok[i] = 0;
          l = 1;
          temp = seq_tab[i][0]-u+1;
          for (j = temp; j <= seq_tab[i][0]; j++)
                  if (seq_tab[i][j] != trailer[j-temp+1]) l = 0;
          ok[i] = l;
  }
  minsofar = 10000; j = 0;
  for (i = 1; i <= paths; i++) {
          if (ok[i] == 1) {
                  temp = eval(seq_tab[i],phrase,plen,u);
                  if (temp < minsofar) {
                          minsofar = temp;
                          j = i;
                  }
          }
  }
  return(j);
}

eval(seq,phrase,plen,u)
int seq[MAXARC],phrase[40];
int plen,u;
{
 int i,j,minsofar;
 int chan,del,ins,seqstart,seqend,index,longer,len1,len2;
```

```
int phrase1[MAXARC], phrase2[MAXARC];

seqstart = 1;
seqend   = seq[0]-u;
if (plen >= seq[0]-u) longer = 1;
else longer = 2;
if (longer == 1) {
        for (i = 1; i <= plen; i++) phrase1[i] = phrase[i];
        for (i = seqstart; i <= seqend; i++) phrase2[i] = seq[i];
        len1 = plen;
        len2 = seqend;
}
else {
        for (i = 1; i <= plen; i++) phrase2[i] = phrase[i];
        for (i = seqstart; i <= seqend; i++) phrase1[i] = seq[i];
        len1 = seqend;
        len2 = plen;
}
index = 0;
chan = del = 0;
minsofar = 10000;
while(len2+index < len1) {
        for (i = 1; i <= len2; i++)
                if (phrase2[i] != phrase1[i+index]) chan++;
        index++;
        if (chan < minsofar) {
                minsofar = chan;
                j = index;
        }
}
del = len1-len2;
return(minsofar+del);
}

segment_events(nevs,evs,workfsm,workarc,strike)
int nevs,strike;
float evs[1000][2];
PFSM workfsm;
PARC workarc;
{
  int i,j,u,sec[200],n,list[100];

  n = 0;
  u = strike;
  if (nevs < u) exit(0);
  for (i = 1; i <= nevs; i++) {
          if (i < u) evs[i][i] = -1;
          else {
                for (j = i; j > i-u; j--) sec[i-j+1] = (int) evs[j][0];
```

```
                j = id(sec,workfsm,workarc);
                evs[i][1] = (float) j;
                if (j == 0 & check(i,list,n,u)) {
                        n++;
                        list[n] = i;
                }
        }
  }
  for (i = u; i < nevs; i++) {
        if (evs[i][1] != 0.0 & evs[i+1][1] != 0.0) {
            j = workfsm->state_tab[(int) evs[i][1]][(int) evs[i+1][1]];
            if (evs[i+1][0] != (float) j & (n == 0 | check(i+1,list,n,u) )){
                n++;
                list[n] = i;
            }
        }
  }
  for (j = 1; j <= n; j++) {
        for (i = 0; i <= u; i++) evs[list[j]+i][1] = 0.0;
  }

}


check(i,list,n,u)
int i,n,list[100];
{
  int j,flag;

  flag = 1;

  if (n > 0) {
        for (j = 1; j <= n; j++)
                if ((i - list[j]) < u+1 & (i - list[j] > -u-1)) flag = 0;
  }
  return(flag);

}

id(sec,workfsm,workarc)
PFSM workfsm;
PARC workarc;
int sec[200];
{
   int i,j,l,u;

   u = workfsm->u;
   for (i = 1; i <= workfsm->v; i++) {
         l = workfsm->id_tab[i][0];
```

```
            for (j = 1; j <= u; j++)
                  if (workfsm->id_tab[i][j] != sec[j]) l = 0;
            if (l != 0) return(l);
      }
      return(0);
}


read_event_history(nevs,evs,filename)
int *nevs;
float evs[1000][2];
char *filename;
{
      int i,ret;
      float num;
      FILE *fopen(),*pf;

      if ((pf = fopen(filename,"r")) == 0) exit(0);
      i = 0;
      nevs[0] = 0;
      while (i == 0) {
            ret = fscanf(pf,"%f",&num);
            evs[nevs[0]+1][0] = num;
            if (ret > 0) nevs[0]++;
            else i = 1;
      }
      fclose(pf);
}



gen_idtab(workfsm,workarc)
PFSM workfsm;
PARC workarc;
{
      int i,j,k,l,u,v,uopt,state,seq,paths;

      u = 1;
      v = 0;
      l = 1;
      i = 0;
      paths = 0;
      uopt = 0;
      seq = 0;

      k = 0;
      while (k == 0) {
          paths = 0;
```

```
      for (state = 1; state <= workfsm->num_states; state++) {
          seq = 0;
          how_many_seqs(workfsm,workarc,&seq,state,u,u);
          fill_id_tab(workfsm,workarc,state,seq,paths,u);
          paths += seq;
      }
      k = check_unique(workfsm,workarc,paths,u);
      if (k == 0) u++;
   }
   workfsm->u = u;
   workfsm->v = paths;
}


check_unique(workfsm,workarc,paths,u)
PFSM workfsm;
PARC workarc;
int paths,u;
{
  int i,j,k,l,flag = 1;

  for (i = 1; i <= paths; i++)   {
         for (j = 1; j <= paths; j++) {
            if (j != i) {
               l = 0;
               for (k = 1; k <= u; k++)
                  if (workfsm->id_tab[i][k] != workfsm->id_tab[j][k]) l = 1;
               if (l == 0) {
                  flag = 0;
                  i = paths+1;
                  j = paths+1;
               }
            }
         }
  }
  return(flag);
}



fill_id_tab(workfsm,workarc,state,seq,paths,u)
PFSM workfsm;
PARC workarc;
int state,seq,paths,u;
{
   int i,d;

   for (i = 1; i <= seq; i++) {
```

```
            workfsm->id_tab[paths+i][0] = state;
    }
    d = seq;
    load_seqs(workfsm,workarc,seq,&d,paths,state,u,u);

}


fill_seq_tab(workfsm,workarc,seq_tab,state,l,p)
PFSM workfsm;
PARC workarc;
int seq_tab[MAXARC][MAXARC];
int state,l,*p;
{
  int i,j,u;
  int back, oback, seq, cseq, paths;

  u = workfsm->u;
  paths = 0;
  for (i = 1+u; i <= l+u; i++) {
          seq = 0;
          how_many_seqs(workfsm,workarc,&seq,state,i,i);
          cseq = seq;
          gen_seqs(workfsm,workarc,seq,&cseq,paths,state,i,i,seq_tab);
          if (paths+seq < MAXARC-1) {
                  paths += seq;
                  for (j = paths-seq+1; j <= paths; j++) {
                          seq_tab[j][0] = i;
                  }
          }
  }
  *p = paths;

}


gen_seqs(workfsm,workarc,seq,cseq,paths,state,back,oback,seq_tab)
PFSM workfsm;
PARC workarc;
int state, back, oback, seq, *cseq, paths;
int seq_tab[MAXARC][MAXARC];
{
  int i,j,q;

  if (back > 0) {
          q = 0;
          for (i = 1; i <= workfsm->num_states; i++) {
                  if(workfsm->state_tab[i][state] != 0) {
                      for (j = 1; j <= cseq[0]; j++) {
```

```
                        if (paths+j < MAXARC)
                                seq_tab[paths+j][oback-back+1] =
                                        workfsm->state_tab[i][state];
                    }
                    gen_seqs(workfsm,workarc,seq,cseq,paths,i,back-1,
                              oback,seq_tab);
                }
        }
  }
  else  cseq[0]--;
}


load_seqs(workfsm,workarc,seq,cseq,paths,state,back,oback)
PFSM workfsm;
PARC workarc;
int state, back, oback, seq, *cseq, paths;
{
  int i,j,q;

  if (back > 0) {
        q = 0;
        for (i = 1; i <= workfsm->num_states; i++) {
                if(workfsm->state_tab[i][state] != 0) {
                    for (j = 1; j <= cseq[0]; j++) {
                        workfsm->id_tab[paths+j][oback-back+1] =
                        workfsm->state_tab[i][state];
                    }
                    load_seqs(workfsm,workarc,seq,cseq,paths,i,back-1,
                              oback);
                }
        }
  }
  else  cseq[0]--;
}



how_many_seqs(workfsm,workarc,seq,state,back,oback)
PFSM workfsm;
PARC workarc;
int state, back, oback, *seq;
{
  int i,q;

  if (back > 0) {
        q = 0;
        for (i = 1; i <= workfsm->num_states; i++) {
                if(workfsm->state_tab[i][state] != 0) {
```

```
                        if (q == 0) {
                              q = 1;
                              if (back == oback) seq[0]++;
                        }
                        else if (q == 1) seq[0]++;
                        how_many_seqs(workfsm,workarc,seq,i,back-1,oback);
                  }
            }
      }
}
```

Appendix B

RAPID

This article describes the Radial Panel Installation Designer (RAPID), a software program for building power network analysis developed at MIT's Laboratory for Electromagnetic and Electronic Systems. The minimum hardware platform for running the program is a PC-AT class computer with an EGA monitor. A mouse is helpful, but not required. Some of the material in this appendix has appeared in [80].

With RAPID, an end-user can quickly and easily create, move, or alter the data which constitutes a building description stored in the program's database. Like a spreadsheet, the program can be used to conduct "what-if" studies on a network. Automatic recalculation permits the creation and modification of power distribution networks far more easily than pencil and paper allow. The program remembers the relationships between the various network elements, e.g., the location and type of transformers, upstream breakers, service entry capacity, etc., and will compute relevant network parameters.

At each stage of the specification process, while loads are being added or altered in the network, the user may enter his own choice of components such as wire, conduit, and breaker size. Or, he may select choices determined by the program on the basis of information the user has entered about the load. If a user's choice is in conflict with the program's, the user is warned. The program determines its choices by referencing tables derived from the National Electrical Code ([96]). These tables are stored in ASCII files and may be modified by a knowledgeable user to reflect relevant local standards.

The Radial Panel Installation Designer is also capable of answering questions about the dynamic operation of parts of the network. Two different types of simulators are available in separate program versions. The first version uses piece-wise linear waveform representations to model the "power profile" of a load. The second uses state space or algebraic models to represent load dynamics.

The first type of simulation is particularly valuable for approximate studies of cold load pick-up transients. In this simulation, a piece-wise linear waveform specified by the user represents the level of apparent power drawn by each load from turn-on to steady-state operation. A simulation is conducted by aggregating these waveforms for all of the loads in a section of interest in the building. Parameters such as peak power drawn on start-up and peak power with respect to average load in steady-state may be determined. This type of

simulation is particularly useful for examining the impact of different operation schedules on the power demand for the building.

The second type of simulation is more sophisticated. A detailed software routine describes the electrical behavior of each load. This software description may be a set of algebraic equations, a state space description of the type used in programs like SIMNON, or a pre-packaged model built into the program for certain loads such as induction motors. The program incorporates a recursive descent parser for interpreting the code and uses advanced numerical integration schemes to solve the differential equation systems. The phase currents for each load are computed for every electrical line cycle. The current waveforms for each load are aggregated to find the composite waveforms for any section of the network. This simulation is considerably more numerically complex than the piece-wise linear simulation. However, this type of simulation is capable of yielding extremely sophisticated and accurate results given accurate load models. The user may save data from a simulation for later examination in other analysis programs such as MATLAB. These results may be especially useful, for example, for analyzing the harmonic content of current waveforms in the network. To maximize simulation efficiency, common loads are represented by pre-packaged, optimized models built into the program.

B.1 Raising a Building with RAPID

The program models the electrical distribution system in a building as a radially-distributed network of *panels.* A *panel* may represent an actual circuit breaker panel in a building, or a busbar, or a transformer. The *root-panel* in a building model typically represents an actual circuit breaker panel fed by the main three-phase utility service entry into the building. Each *switch* on a panel represents a circuit breaker or bus connector which initiates a connection to a load or a sub-panel, busbar, or transformer.

To create a building model, the user first answers questions which define the utility service entry – details like the line-to-line voltage, entry fusing, entry transformer type, and so on. Once the program has the basic details of the service entry, a main panel file is created in a building database. This main panel is the root-panel for what will eventually be a network of panels, distributed radially from the main panel, which describe all or part of the building's electrical wiring.

B.1.1 Describing the Wiring

Initially, the main panel contains no switches. The user specifies the details of each switch or breaker on the panel, one switch at a time. A computer screen image showing the main panel for a building named "TEST" is shown in Fig. B.1. Several of the switches on this panel have already been specified. It may be helpful to refer to Fig. B.1 throughout the discussion in this section.

To add a new switch to the panel, the computer's mouse is clicked in the window labeled *switch* in Fig. B.1. A blank switch is created in the window. The program queries the user for some basic "boiler plate" information about the device to be added. The

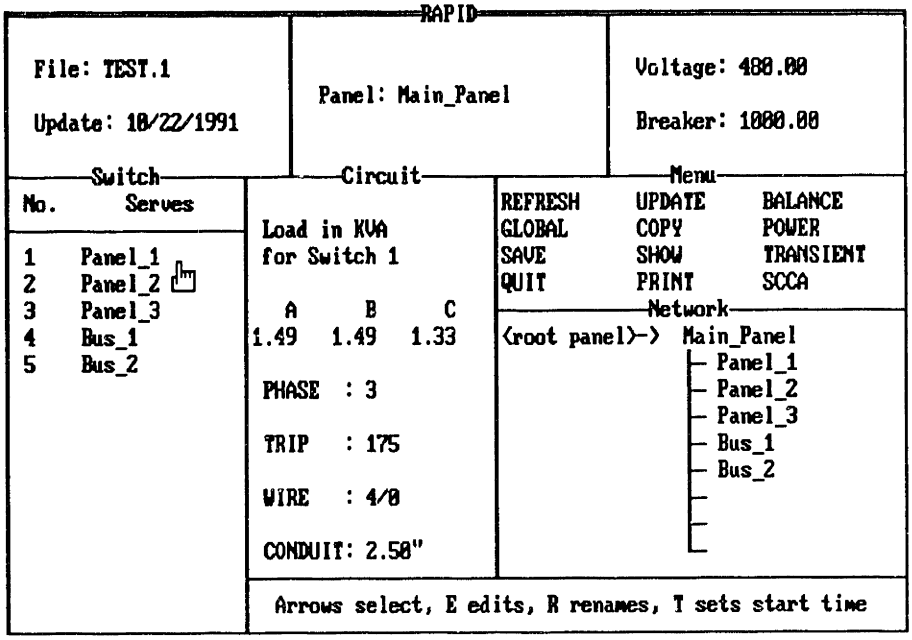


Figure B.1: RAPID's Main Interface Window

program recognizes eight different device classes:

- Load: This category represents most general electrical loads.
- Motor: This category includes induction, dc, and synchronous electric machines, or loads whose primary electrical component is a rotating electric machine.
- Receptacle: This represents an electrical outlet, recognized only on 120/208 volt panels in the current version of RAPID.
- Spare and Space: These represent unused portions of the panel.
- Panel, Transformer, and Bus: These three categories represent the classes which can be RAPID panels. The program understands the differences between the various transformer winding types, such as $Y - \Delta$, $Y - Y$, and $\Delta - \Delta$ transformers. Phase currents are transformed correctly across the transformer types selected by the user.

Based on the device name input by the user, the program attempts to interpret the device class. The user is offered a default class for the device, or he may select any other choice from the remaining seven classes. The program also asks for the nominal load power, power factor, and number of phases used by the load.

The remaining questions about the load are cross-referenced against the program's knowledge of the electrical code and the user-specified load characteristics. The user is asked, for example, to specify the breaker or fuse type, wire size for the hook-up, associated

conduit, turn-on and turn-off times, etc. Some questions may be device class specific. For each question, the user is allowed to enter his own choice or select a programmed default. If the user's choice conflicts with the program's in an unsafe manner, e.g., the wire size is too small, the user is warned.

During this device creation process, the user is also asked to specify the transient electrical characteristics of the load. Depending on the version of the program, this specification may involve the creation or selection of a previously saved piece-wise linear waveform, or it may involve the creation of a state space description that specifies the device's electrical characteristics. The first version of the program contains a convenient graphical entry system for describing a transient waveform profile. The second version contains a built-in editor that allows the user to enter a code fragment describing the load.

The *circuit* window in Fig. B.1 displays pertinent information about a specific device under consideration. For example, when a new device is fully specified, the circuit window will be updated to display the steady-state power and hook-up specification for the new load. The information for any load on the panel may be viewed in this window by first selecting the load in the switch window.

If the new load is a panel, busbar, or transformer, a new database file will be created for the load and the *network* window will be updated. This window shows the local structure of the building power network. The top of the "tree" in this window, shown in Fig. B.1, is the panel currently under examination. To the left of the current panel is the "parent" panel for the current panel. In Fig. B.1, the root panel in the network is under examination, so the parent panel is listed as <*root–panel*>. The new, "child" panel, busbar, or transformer which has just been added will appear as a "leaf" below the current panel. The user may move up and down the network by selecting a parent or child panel with the mouse. When a new panel is selected, the entire screen shown in Fig. B.1 will be updated. The program will update the top of the screen in Fig. B.1 to display the status information for the new panel, e.g., line-to-line voltage, feeder breaker size, etc. The switch and circuit information for the new panel will be displayed, and the network window will be updated to show the current location in the network.

Any previously specified switch may be deleted. The specifications for any load which is not a panel, busbar, or transformer may be altered or copied. The ease with which data may be entered and altered makes the program ideal for tracking and analyzing changing load configurations.

B.1.2 Tailoring the Program

The user may adjust some of the parameters governing the program's operation by selecting the *global* command in the *menu* window, shown in Fig. B.1. This command activates the tailoring window, shown in Fig. B.2. By selecting one of the options, the user may set the selected parameter. Examples include the smallest wire size allowed in the network, the number of switches permitted on a single panel, and the maximum over-current to be used in computing fuse or breaker sizes.

The user may engage in more sophisticated tailoring of the program by altering the ASCII files in which the program stores its knowledge of basic wiring requirements. These

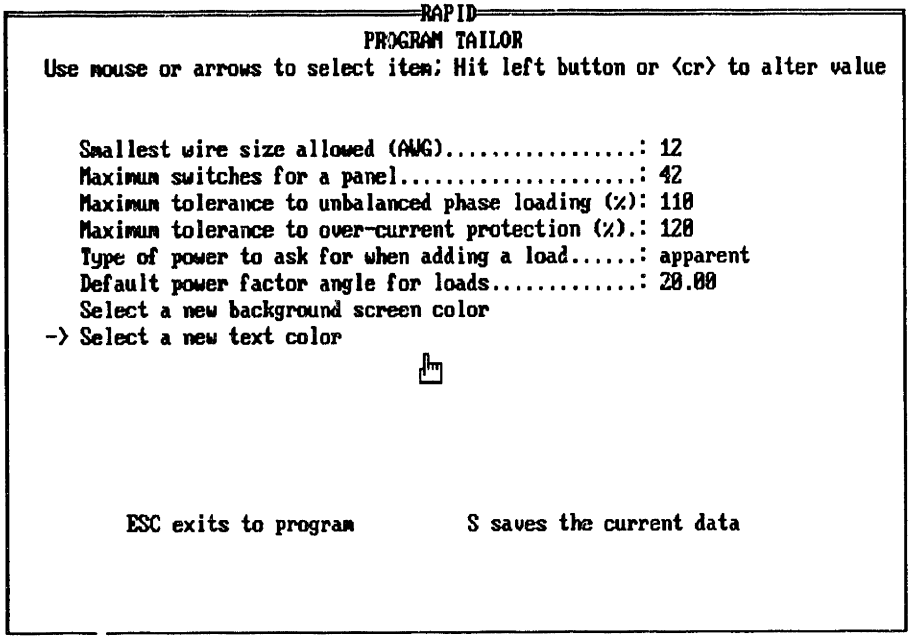


Figure B.2: Tailoring Window

tables cross-reference wire ampacity and gage with breaker and conduit size and type. Default power factor information for a variety of common loads is also saved in these tables. These tables are initially set to reflect requirements specified in the National Electrical Code. Altering the entries in these tables will tailor the program for a wide variety of local or proprietary standards.

B.1.3 Analyzing a Network

Many of the commands in the *menu* window provide information about the operating characteristics of the network. The *balance* and *power* commands may be used to find the relative steady-state phase loading and total steady-state power per phase, respectively, for the current panel under examination. The program automatically aggregates the total power used on all sub-panels. Any changes to the loading of sub-panels are reflected in the total steady-state power and balance calculations for the current panel.

The *print* command produces a summary sheet of the loads on the current panel by switch number; this summary is suitable for updating the legends frequently incorporated in circuit breaker panels.

The program is also capable of providing a rudimentary short circuit current analysis through the *scca* command. The user may place a hypothetical fault on a one-line diagram of the network, as shown in Fig. B.3. The program will compute the available short circuit current at the fault and also at the service entry based on the calculated impedances of the connecting wires and the known impedances of transformers in the network.

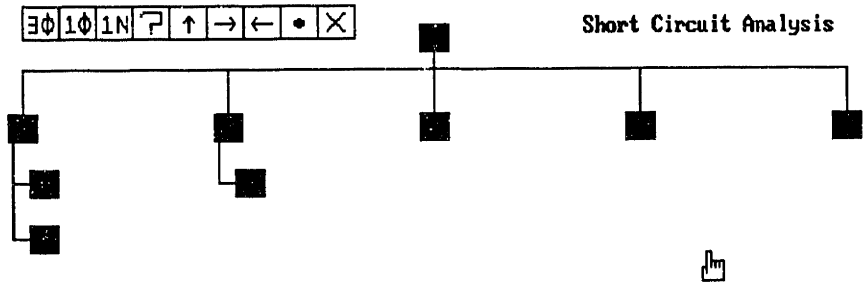


Figure B.3: One-Line Network Diagram

The *transient* command engages the load simulator, as described below.

B.2 Studying Electrical Transients

The piece-wise linear simulator is particularly useful for studying the impact of different operating schedules on the peak power demand over certain intervals. Part of the specification information for entering a new load into the program is an absolute start and stop time for the load, assuming that no load starts before time zero. The user may easily change the start and stop times for any load, permitting the study of a limitless range of operating schedules. For example, the program can give rough estimates of the cold load pick-up transient, i.e., the electrical demand when many loads are suddenly turned on. Also, the program can be used to assist in the timing of load operation to help avoid excessive electrical demand during any single short time billing period. The simulation may be conducted for the whole or any part of the network.

For an example of typical results from the piece-wise linear simulator, consider the plot of power versus time shown in Fig. 1.4. This plot shows the actual aggregated apparent power drawn by a bank of three thermostatically-controlled resistive heaters. Figure B.4 shows a RAPID simulation of this bank of heaters, in which the cycle times for the various heaters have been approximated and entered as part of the load model. Given a reliable load model, the program is capable of constructing accurate apparent power usage profiles. A one-time investment in collecting basic load information such as the cycle time for the heaters, either through on-line data collection or off-line calculation, can yield generally applicable models for a variety of load simulations. Because the program stores the entire power waveform profile for each load, computation during the simulation is minimized and many simulations may be conducted in a relatively short time.

Figure B.5 shows one of the three phase currents computed by a RAPID state space simulation of a panel with two loads, a three phase induction motor and a single phase resistive heater. This type of simulation is more sophisticated than the piece-wise linear simulation. The induction motor starts first at time zero. The resistive heater starts at time

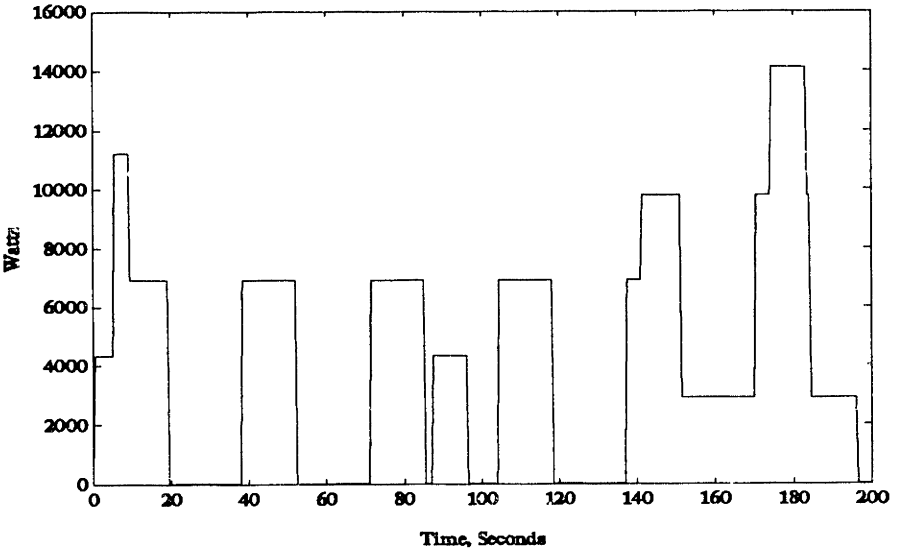


Figure B.4: Simulated Power Profile

one, after the induction motor has entered steady-state. The plot in Fig. B.5 shows one of the three phase currents on a phase that services both the induction motor and the heater for a simulation interval of 1.5 seconds. The induction motor model is a standard induction motor state space representation that has been built into the program. The resistive heater is modeled with a simple set of algebraic equations. While more time-consuming, this analysis is capable of giving detailed, cycle-by-cycle representations of the load dynamics. The level of detail presented by the state space simulation is particularly suited for harmonic content analysis and high resolution transient studies. Point-by-point data from the simulations may be down-loaded to an ASCII file for analysis in other programs like MATLAB.

B.3 Conclusions

The Radial Panel Installation Designer provides many important tools for the industrial or commercial facilities manager, not available in many standard CAD and database packages for building design and maintenance. The ability to swiftly prototype and alter the database which describes a building's electrical network facilitates good documentation habits. More significant, the program's analysis tools can provide speedy safety checks and answers to hypothetical load scheduling questions with a greater degree of certainty and flexibility than most other programs. The program's customized interface and analysis routines make the program a more natural tool for tracking building operation than a generic spreadsheet or database program.

The program has proven to be an invaluable tool for generating raw data for initial checks of the prototype event detector. It has been used in the Laboratory for Electromagnetic and Electronic Systems for providing test data for harmonic analysis hardware and

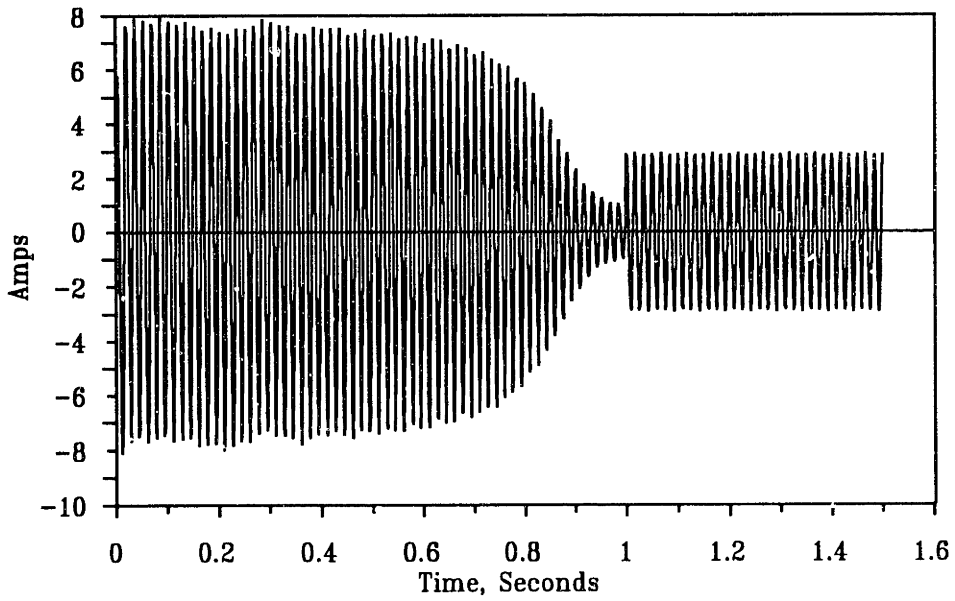


Figure B.5: State Space Simulation

electrical transient pattern recognition algorithms, and may eventually be used in forecasting studies.

Appendix C

EDWS

The Extended Duration Waveform Synthesizer (EDWS) is an electronic circuit designed to create one or more high bandwidth analog signals from stored digital samples. The EDWS operates from the internal bus of a personal computer. Digital data is stored on a mass storage device in the computer, typically a hard drive. The EDWS reads data from the storage device into a section of page-interleaved electronic memory and outputs the data through one or more digital-to-analog converters to create analog waveforms. Page-interleaving has been used to decrease the memory access time in personal computers and workstations. Until now, this scheme has not been widely employed for use in waveform synthesis applications. Page-interleaved memory buffers make the EDWS circuit architecture unique among waveform synthesizers. The page-interleave memory buffering scheme, described in more detail below, allows the EDWS to output analog data without interruption, despite the need to periodically read new data points from the mass storage device on the personal computer.

The EDWS will synthesize analog waveforms continuously as long as there is fresh data on the mass storage device. That is, the size of the mass storage device is the only limitation on the amount of data the EDWS can process without interruption to create analog output. Other waveform synthesizers create only analog waveforms with fixed shape, or, in the case of caching synthesizers, they create waveforms that are flexibly specifiable over a short interval and which must then either terminate or repeat. As a benchmark, a typical cache-type, four channel, PC-based commercial waveform synthesizer with 16K words of memory creating analog waveforms at a sample rate of 1920 Hertz per channel is capable of generating new data points for about 2 seconds before the analog waveforms must either terminate or repeat. The EDWS, with the same 16K words of memory arranged in the page-interleave architecture operating from a low-end personal computer with a 40 megabyte hard disk is capable of generating new analog data points on four different channels at the same sample rate of 1920 Hertz for over 40 minutes. If desired, the EDWS can also operate in the traditional mode, in which data is stored once in the memory buffer and repeated to create a periodic waveform. Hence, the EDWS design increases the overall functionality of a waveform synthesis system with no loss in flexibility.

C.1 Background

The original EDWS design described in this appendix creates four different analog data channels. Installed in a low-performance IBM PC, the EDWS is capable of synthesizing waveforms at a rate of 1920 Hertz per channel. The EDWS is currently being used in the Massachusetts Institute of Technology's Laboratory for Electromagnetic and Electronic Systems (LEES) to provide analog test waveforms for the development of the prototype multiscale event detector for use in the NILM. In short, the EDWS functions as a kind of disk player, creating analog waveforms from digital data derived from computer simulations of a building's operation. The waveforms created by the four analog data channels of the EDWS are used to represent the four currents in a three-phase, four wire power distribution system, as might be found in a large commercial or industrial building. The prototype event detector monitors the waveforms created by the EDWS. These "played-back" analog waveforms provide the raw data used to test the operation of the detector.

The potential utility of the EDWS is not limited to testing the NILM and its components. The EDWS is generally useful to create waveforms for testing power quality monitors. The 1920 Hertz output sample rate of the EDWS resolves up to 16 harmonics of the 60 Hertz utility fundamental frequency. Hence, the EDWS can represent waveforms with substantial frequency content with respect to a 60 Hertz fundamental. This makes the EDWS an ideal test platform for creating a variety of pathological waveforms, obviating the need to collect real data in the early stages of research and testing of different power monitoring devices.

The performance of the EDWS improves with the quality of the host computer on which it is installed. Tests conducted in LEES indicate that a modern, 486-based personal computer could easily permit the four channel EDWS to run at 3840 Hertz per channel, double the operation rate when installed in a low-end PC. Furthermore, the sample rate per channel can be increased by lowering the total number of channels. Halving the number of channels doubles the data rate per channel. So, a two channel EDWS installed on a 486-based PC could run at a sample rate of 7680 Hertz per channel. With a 40 megabyte hard disk, this sample rate is sufficient, for example, to synthesize 40 minutes of moderate quality speech on two different channels. By tailoring the operation of the EDWS and the host computer, the EDWS could be used in an enormous number of tasks for long duration, high bandwidth waveform synthesis.

C.2 System Overview

This section reviews the operation of the Extended Duration Waveform Synthesizer. At any instant during the steady state operation of the EDWS, one of two memory buffers (buffer A or buffer B) is being loaded with fresh data points, while data points from the other buffer (buffer B or buffer A) are read into the output latches. Assume for the moment that buffer A, the *inactive buffer*, is being loaded with data while buffer B, the *active buffer*

is writing data to the output latches. A software program running on the host PC controls the data flow from the mass storage device. Data points are loaded sequentially into one of the input buffers, buffer A in this example. The address counter A cycles through each of the memory locations in memory buffer A. As each location is addressed, a fresh data point from the PC interface is read into the memory location from the input buffer. While buffer A is being loaded with fresh data, the address counter B cycles through each of the memory locations in buffer B. As each buffer B location is addressed, the data stored in the memory location is written out to the output latch B. This data point is converted to an analog voltage level by one of a bank of digital-to-analog converters (DACs) on the output of the EDWS. By the time all of the data in memory buffer B has been written to the DACs, buffer A has been completely filled with fresh data. At this time, the control block switches the roles of the two memory buffers. Now, the data in buffer A is written out to the DACs, while buffer B is loaded with fresh data. New data is continuously written to the DACs until the data stream on the PC mass storage device is exhausted.

Multiple channels are handled by partitioning the memory buffers. In the current EDWS implementation, each buffer consists of 16K bytes of memory space. One DAC is required for each analog output channel. Each DAC in the current implementation requires two bytes or one word per output point. For four channels of analog output, 4K bytes or 2K words of memory are available per channel. On a simple PC, this amount of memory permits the stated peak output sample rate per channel, 1920 Hertz, with uninterrupted output. In other words, the 1920 Hertz output rate will insure that the inactive buffer in the system is filled with fresh data before the four 4K byte blocks in the active buffer are exhausted.

On a faster computer, the rate at which the inactive buffer can be filled will increase. Hence, the peak output sample rate in the active buffer may also be increased. Alternatively, on a faster computer, the total number of channels at the original 1920 Hertz peak output sample rate could be increased. If only two channels are desired the peak output sample rate could also be increased. In a two channel arrangement, the memory would be partitioned in 8K byte or 4K word blocks, one block for each of the two channels. Each of the two blocks in the active buffer could be emptied at a rate of 3840 Hertz and the inactive buffer would be guaranteed to be filled before the active buffer was exhausted.

The EDWS may be configured to operate as a "standard" waveform synthesizer by filling the buffers with data once and then repeating the data, *ad infinitum*. This results in an output waveform comparable to the type derived from a fixed or caching waveform synthesizer. In this operating mode, the peak output sample rate is no longer limited by the rate at which data may be loaded from the PC and very high data rates are possible.

C.3 Reduction to Practice

A four channel EDWS has been implemented in MIT's LEES laboratory. Operation of this hardware has provided experimental verification of the anticipated performance benefits of the EDWS. See [126] for a detailed review of real-time experiments with the EDWS.

In summary, the features of the Extended Duration Waveform Synthesizer include:

- The use of page-interleaved memory in a waveform synthesis environment.
- A synthesis duration limited only by the amount of available mass storage on the host PC.
- A flexible trade-off between the number of output channels and peak output sample rate per channel.
- The ability to be configured for operation as a standard waveform synthesizer if needed.

The potential applicability of the EDWS includes but is not limited to the original use of testing power monitoring hardware. Almost any electronic application which makes use of a waveform generator or synthesizer could benefit from the availability of flexible, long duration waveform synthesis.

Appendix D

Event Detector Schematics and Software

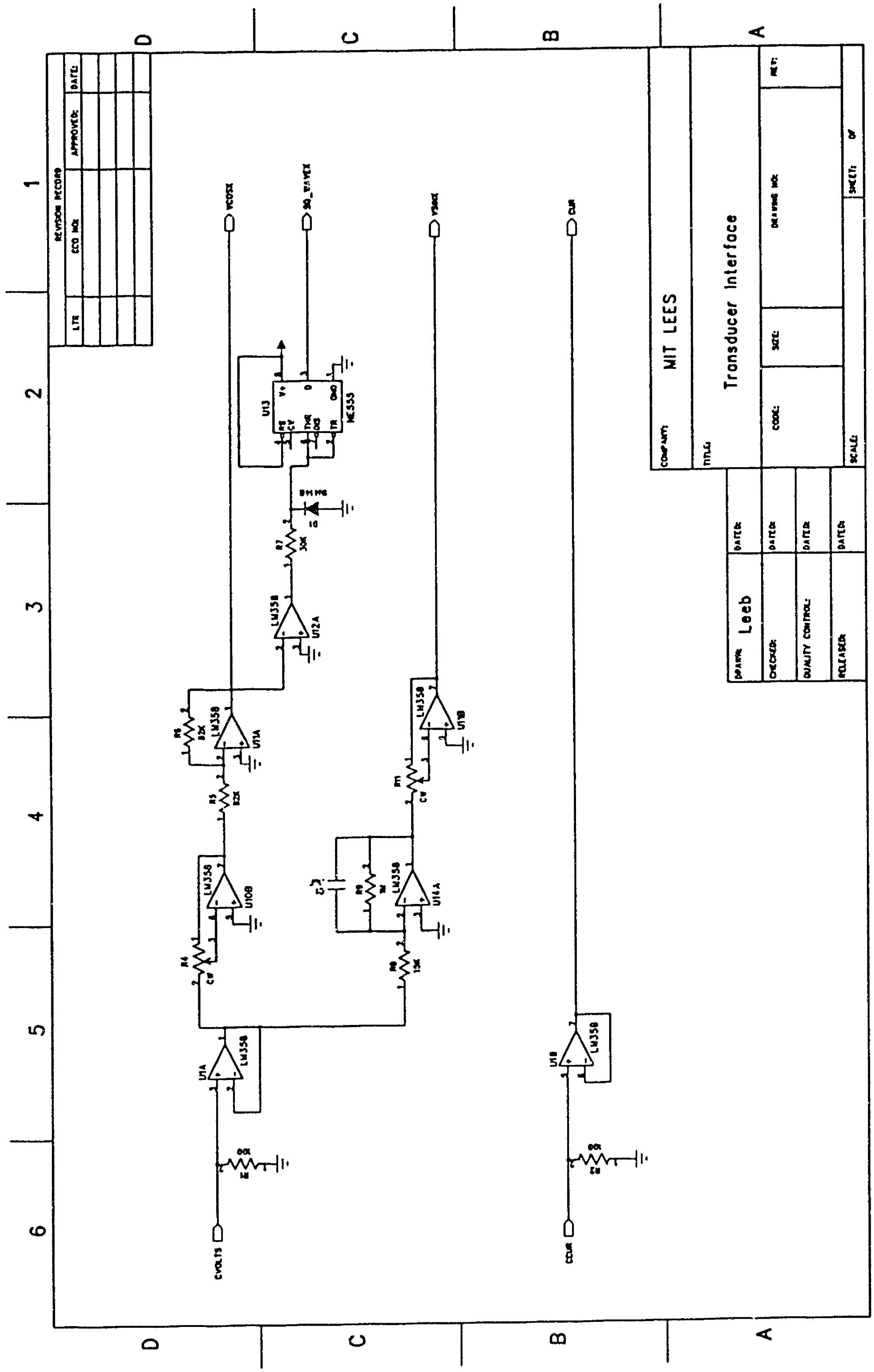
This appendix reviews the circuit schematics for the analog preprocessor described in Chapter 5, as well as the source code for the *NILMscope* DSP event detection algorithm. The user interface code, responsible for such operations as drawing the graphs in Figs. 5.4 through 5.13, has not been included, in order to save space while focusing attention on the event detection algorithm. To assist in further understanding the algorithm presented in Chapter 5, and to ensure the reproducibility of the results, the source code that specifically describes the transient event recognition process has been included and documented in detail.

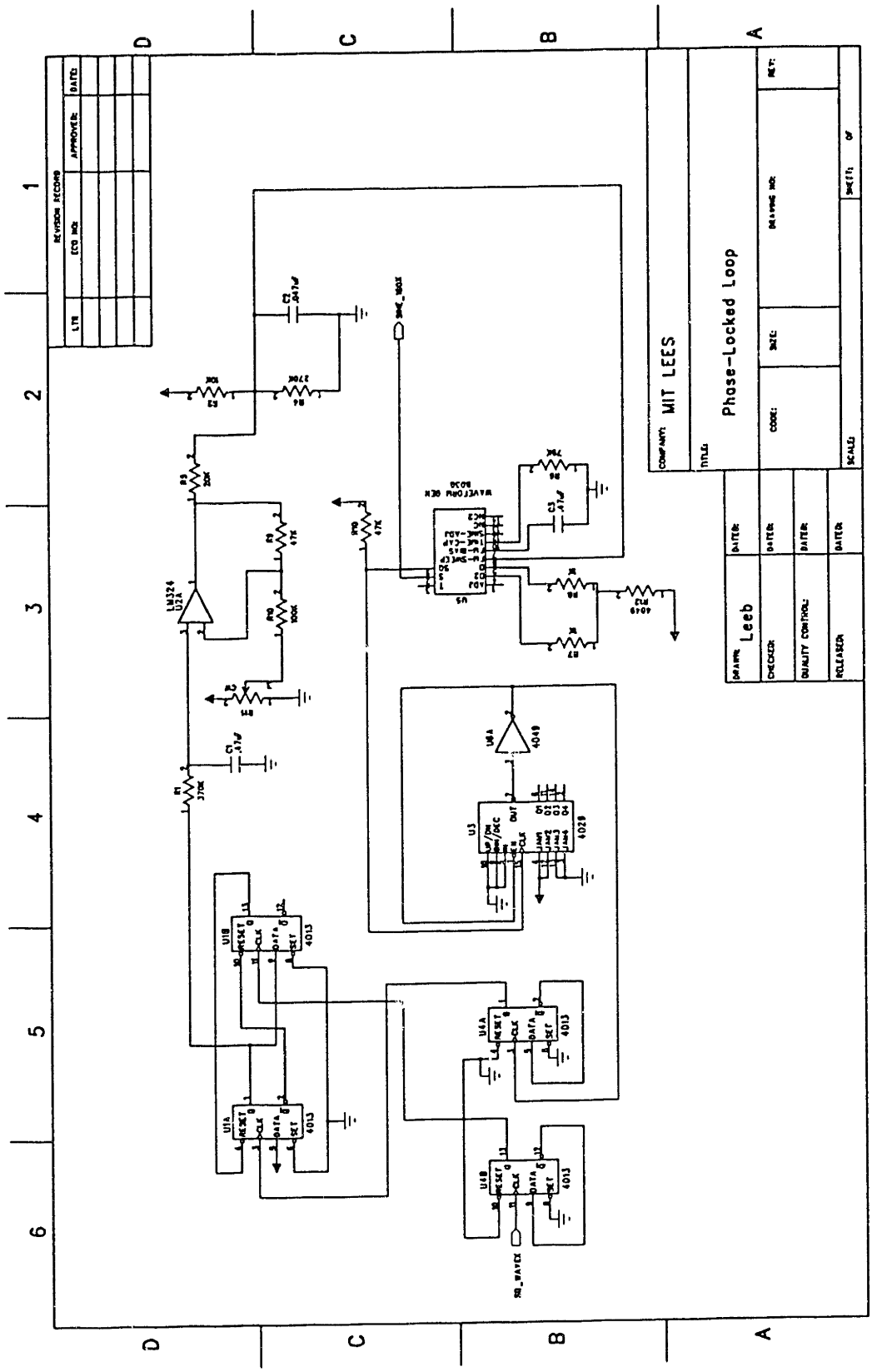
D.1 Analog Preprocessor Schematics

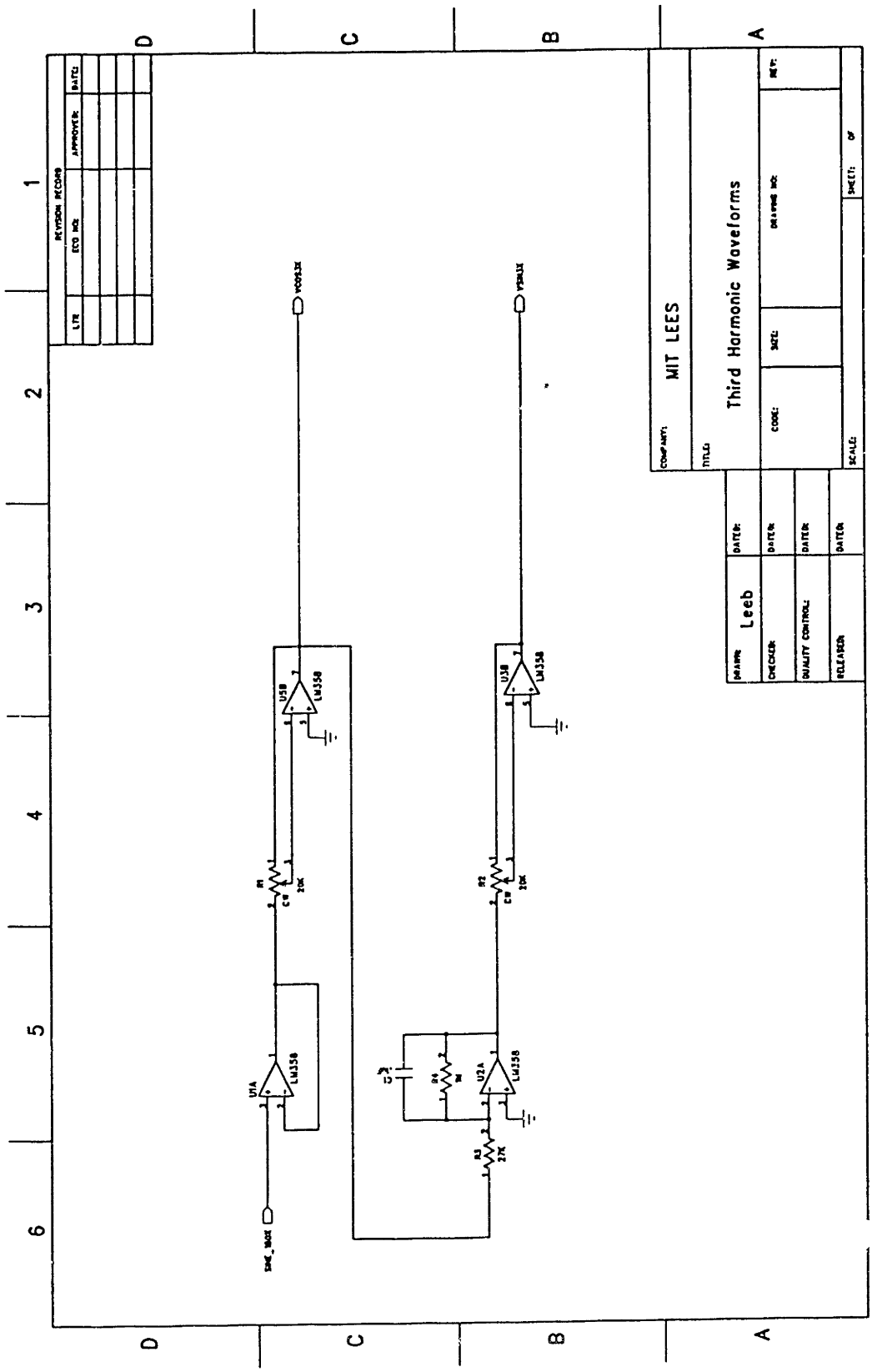
The following pages of schematics document the construction of the analog preprocessor described in Chapter 5. The preprocessor contains the voltage and current sensors that deliver isolated and scaled waveforms that correspond to the line voltage and current on each phase of a three phase electrical service. It computes estimates of the envelopes of real and reactive power and also the in-phase and quadrature third harmonic content of the current on each phase. The calibration of the conversion factors used in the *NILMscope* user interface software which relate the envelopes computed by this circuitry to the actual real and reactive power levels is approximate. So, the graphs in Figs. 5.4 through 5.13, for example, have the correct shape but potentially not the exactly correct amplitudes. Since the v-section templates were developed with data collected by the same analog preprocessor used for the experiments, any calibration errors have no effect on the performance of the event detection algorithm as implemented in the prototype.

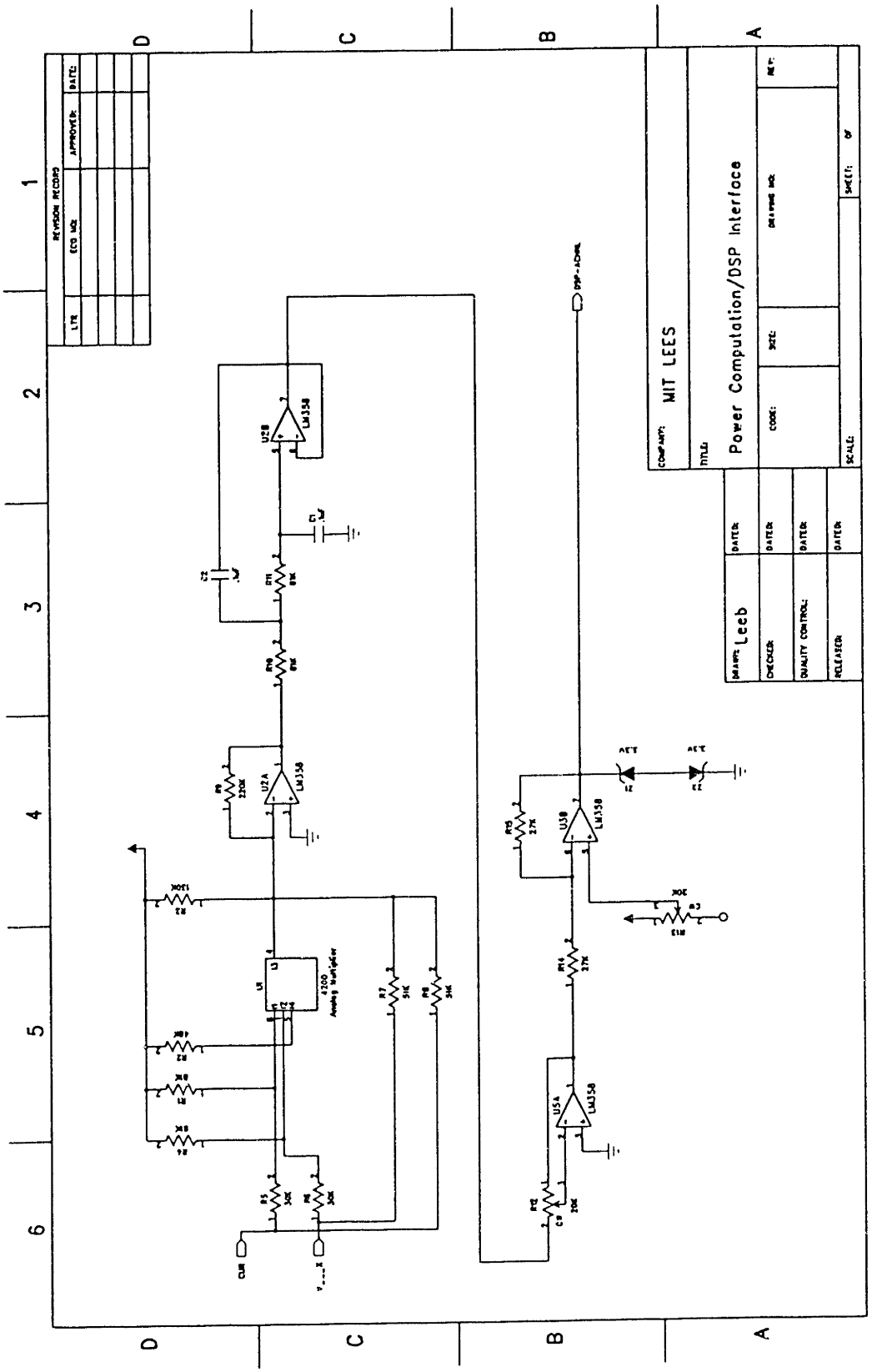
D.2 *NILMscope* DSP Software

The sections below contain the key source code components of the event detection algorithm as implemented in the *NILMscope* software that was used in the real-time experiments with


MIT LEES
Transducer Interface
Leeb
COMPANY:
TITLE:
DRAWN:
CHECKED:
QUALITY CONTROL:
RELEASED:
CODE:
SIZE:
DRAWING NO:
SCALE:
REVISION RECORD
LTR
ECO NO:
APPROVED:
DATE:
NE555
U13
LM358
CVOLTS
CCUR
D
C
B
A
1
2
3
4
5
6


MIT LEES
Phase-Locked Loop
Leeb
LM324
4049
4029
4013


MIT LEES
Third Harmonic Waveforms
Leeb
LM358
U1A
U2A
U3B
U4B
R1
R2
R3
R4
C1
C2
10K
2K
1M
27K


MIT LEES
Power Computation/DSP Interface
Leeb
LM358
4200
Analog Multiplier
DSP-ADIN
CLK

the prototype. These sections were specifically referenced during the review of the algorithm in Chapter 5. Figure 5.3 and the explanation of the algorithm steps in the associated text will be helpful in understanding the role of the routines that follow.

The first three routines, *nilm*, *nilm2*, and *nilm3*, implement the data acquisition, tree-structured decomposition, and v-section set recognition procedures, respectively. All three sets of code execute on the TMS320C30 DSP system. They can be compiled with the TMS320C30 *C* compiler and *SPOX* interface library, provided with the programmer's development kit when the DSP system is purchased. The fourth routine, *lock*, implements the inter-scale v-section lock out. It runs on the 80486 PC component of the prototype. It may be compiled with the version 5.1, Microsoft *C* compiler.

D.2.1 Data Acquisition Routines: *nilm*

This is the first component of the *NILMscope* software run on the DSP system. It executes after the event detector has been armed by the appropriate command from the PC, issued by the user of the prototype. This code monitors the two analog channels on the DSP system, waiting for a triggering event. When the system is triggered, 330 sample points are collected on each channel and written to the mass storage device on the PC.

```
#include <spox.h>
#include <dan.h>
#include <math.h>
#include <stdio.h>

#define SIZE 1
#define ASIZE 300
#define SAMPLECLK 46296  /* 180 Hz sample rate */

smain()
{

        SS_Stream instream;   /* Streams for the analog channels */
        SS_Stream instream2;
        SA_Array  array;
        SA_Array  array2;
        SV_Vector data;
        SV_Vector data2;
        SV_Vector oldda*a;
        SV_Vector olddata2;
        SV_Cursor cur;
        int       len;
        int       i;
        int       j;
        int       firsttime;
        int       flag;
        float     cnt;
        float     avg;
```

```
float     sum;
float     old;
float     *anach1;
float     *anach2;
FILE      *pf;

/* allocate space for the variables that store individual */
/* samples from the input streams.                        */
anach1 = (float *) malloc(ASIZE*sizeof(float));
anach2 = (float *) malloc(ASIZE*sizeof(float));

/* Initialize the analog input streams on both DSP channels */
instream  = SS_create(SS_NULL, 0, SIZE, NULL);
instream2 = SS_create(SS_NULL, 0, SIZE, NULL);
olddata   = SV_create(FLOAT, SA_create(0, SIZE, NULL), NULL);
olddata2  = SV_create(FLOAT, SA_create(0, SIZE, NULL), NULL);

SS_open(instream, "/ain0", SS_READ);
SS_open(instream2,"/ain1", SS_READ);

array  = SA_create(SS_memseg(instream), SIZE, NULL);
array2 = SA_create(SS_memseg(instream2), SIZE, NULL);

SS_ctrl(instream, DAN_CSETSAMPLERATE, SAMPLECLK);
SS_ctrl(instream2, DAN_CSETSAMPLERATE, SAMPLECLK);

flag = 1;
firsttime = 1;
while(flag) {  /* keep sampling until a window of data is collected */

        SS_get(instream, array);
        data = SV_create(FLOAT, array, NULL);

        /* Wait for a triggering event */
        if (firsttime == 0) {
                for(len=SV_scan(data, &cur),cnt=0.0;len>0;len--){
                        cnt = *(Float *)SV_next(&cur);
                        if (fabs(cnt-old) > 200.0) flag = 0;
                }
        }
        else firsttime = 0;

        if (flag == 1) {
                for(len=SV_scan(data, &cur),cnt=0.0;len>0;len--){
                        old = *(Float *)SV_next(&cur);
                        len = 0;
                }
        }
```

```
                /* Once a triggering event is received, collect two */
                /* vectors of data, one on each channel.            */
                if (flag == 0) {
                        for (i = 0; i < ASIZE; i++) {
                            SS_get(instream, array);
                            SS_get(instream2, array2);
                            data = SV_create(FLOAT, array, NULL);
                            data2 = SV_create(FLOAT, array2, NULL);
                            for(len=SV_scan(data, &cur),cnt=0.0;len>0;len--){
                                anach1[i] = *(Float*)SV_next(&cur);
                            }
                            for(len=SV_scan(data2, &cur),cnt=0.0;len>0;len--){
                                anach2[i] = *(Float *)SV_next(&cur);
                            }
                        }
                }

        }


        /* Write the two collected windows of data to the mass     */
        /* storage device on the PC.  Free all allocated variable  */
        /* space, and quit.                                        */
        pf = fopen("data.dat","w");
        for (i = 0; i < 30; i++) fprintf(pf,"%f %f \n",0.0,0.0);
        for (i = 0; i < ASIZE; i++)
                fprintf(pf,"%f %f \n",(anach1[i]-old)*3.0/32768.0,
                        (anach2[i]-anach2[0])*3.0/32768.0);
        fclose(pf);
        free(anach1);
        free(anach2);
}
```

D.2.2 Tree-Structured Decomposition: *nilm2*

Once the *nilm* routines have collected windows of data on every input channel, the following code performs a tree-structured decomposition. This code runs on the DSP system. It produces the input data for the two coder stages in the decomposition following the first stage, which operates with the data collected at the highest sample rate by the *nilm* routines. Once this code has executed, the mass storage device on the PC will contain three files. The first is the original data collected by *nilm*. The second and third files contain the inputs to the coarser coder stages produced by *nilm2*. The decomposition employs an FIR filter in the resolving path of each coder stage. This filter is designed by the *remz* algorithm, presented later in this appendix. The filter taps used for the experiments in Chapter 5 are also included later in this appendix.

```
#include <spox.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

#define ASIZE   330
#define THRESH  0.04
#define SWEEP   5

smain()
{
        int       i;
        int       j;
        int       count;
        int       count2;
        int       flag3;
        float     num1;
        float     num2;
        float     *anach1;
        float     *anach2;
        float     *res;
        float     *res2;
        float     *res3;
        float     *res4;
        float     ifilt[20];
        FILE      *pf;

        /* allocate space to read in data and produce the     */
        /* filtered and downsampled inputs for the subsequent */
        /* coder stages.                                      */
        anach1 = (float *) malloc(ASIZE*sizeof(float));
        anach2 = (float *) malloc(ASIZE*sizeof(float));
        res    = (float *) malloc(ASIZE*sizeof(float));
        res2   = (float *) malloc(ASIZE*sizeof(float));
        res3   = (float *) malloc(ASIZE*sizeof(float));
        res4   = (float *) malloc(ASIZE*sizeof(float));

        /* Read in the data collected by nilm */
        pf = fopen("data.dat","r");
        for (i = 0; i < ASIZE; i++) {
                fscanf(pf,"%f",&num1);
                fscanf(pf,"%f",&num2);
                anach1[i] = num1;
                anach2[i] = num2;
        }
        fclose(pf);

        /* Load the filter taps stored in the file filt.dat */
```

```
if ((pf = fopen("filt.dat","r")) == 0) {
                printf("ERROR ==> Cannot read filt.dat\n");
                exit(0);
}
for (i = 0; i < 11; i++)  { fscanf(pf,"%f",&num1); ifilt[i] = num1;}
fclose(pf);

/* Open a file to store the input to the second coder stage */
pf = fopen("data1.dat","w");

/* Convolve the raw input data with the filter taps */
/* On Channel 1: */
conv(anach1,ifilt,ASIZE,11,res);
/* This next function call performs an adaptive, 2 to 1 */
/* sample rate alteration, as described in Chapter 4.   */
adapt_decimate(res,res3,anach1,&count,ASIZE);

/* On Channel 2: */
conv(anach2,ifilt,ASIZE,11,res);
/* This next function call performs an adaptive, 2 to 1 */
/* sample rate alteration, as described in Chapter 4.   */
adapt_decimate(res,res4,anach2,&count2,ASIZE);

/* Check length to avoid fence post problems. */
if (count2 < count) count--;
else if (count < count2) count2--;
if (count > (ASIZE-14)/2) {count--; count2--;}
i = count;
while(count < ASIZE/2) {
        res3[count] = res3[i-1];
        res4[count2] = res4[i-1];
        count++; count2++;
}

/* Reduce the amplitude of the data */
for (i = 0; i < count; i++)
        fprintf (pf,"%f %f\n",res3[i]*.9,res4[i]*.9);

fclose(pf);


/* Open a file to store the input to the third coder stage */
pf = fopen("data2.dat","w");

/* Convolve the raw input data with the filter taps */
/* On Channel 1: */
conv(res3,ifilt,ASIZE/2,11,res);
/* This next function call performs an adaptive, 2 to 1 */
/* sample rate alteration, as described in Chapter 4.   */
```

```
        adapt_decimate(res,res2,res3,&count,ASIZE/2);

        /* On Channel 2: */
        conv(res4,ifilt,ASIZE/2,11,res);
        /* This next function call performs an adaptive, 2 to 1 */
        /* sample rate alteration, as described in Chapter 4.   */
        adapt_decimate(res,res3,res4,&count2,ASIZE/2);

        /* Check length to avoid fence post problems. */
        if (count2 < count) count--;
        else if (count < count2) count2--;
        if (count > ((ASIZE/2)-1)/2) {count--; count2--;}
        i = count;
        while(count < (ASIZE/2-1)/2) {
                res2[count] = res2[i-1];
                res3[count2] = res3[i-1];
                count++; count2++;
        }

        /* Reduce the amplitude of the data */
        for (i = 0; i < count; i++)
                fprintf (pf,"%f %f\n",res2[i]*.9,res3[i]*.9);

        fclose(pf);


        /* free space and quit */
        free(anach1);
        free(anach2);
        free(res);
        free(res2);
        free(res3);
        free(res4);

}


/* Performs the adaptive decimation described in Chapter 4 to */
/* ensure that the best set of points is passed to each coder */
/* stage.                                                     */
adapt_decimate(res,res3,anach1,rcount,asize)
float *res;
float *res3;
float *anach1;
int *rcount;
int asize;
{

        int       loc[ASIZE];
```

```
        int       nsegs;
        int       i;
        int       j;
        int       count;
        float     suma;
        float     sumb;
        float     num1;
        float     num2;


        nsegs = 1;  loc[nsegs-1] = 0;   num1 = anach1[0]; j = 0;
        while (j < asize-14) {
                while(fabs(num1 - res[j]) < THRESH) j += 2;
                while(fabs(num1 - res[j]) > THRESH) j += 2;
                if (j < asize-13) {
                        if (j > loc[nsegs-1]) {
                                loc[nsegs] = j+15;
                                nsegs++;
                        }
                }
        }
        loc[nsegs] = asize-14; nsegs++;

        count = 0;
        for (j = 0; j < nsegs-1; j++) {
                suma = 0.0; sumb = 0.0;
                for (i = loc[j]; i < loc[j+1]; i += 2) {
                        suma += res[i]*res[i];
                        sumb += res[i+1]*res[i+1];
                }
                if (suma > sumb) {
                        for (i = loc[j]; i < loc[j+1]; i+= 2) {
                                res3[count] = res[i];
                                count++;
                        }
                }
                else {
                        for (i = loc[j]; i < loc[j+1]; i+= 2) {
                                res3[count] = res[i+1];
                                count++;
                        }
                }
        }
        *rcount = count;
}


/* Convolve the vector src with the vector temp */
conv(src,temp,ls,lt,res)
```

```
float *src;
float *temp;
int ls;
int lt;
float *res;
{
   int i;
   int j;
   float sum;

   for (i = 0; i < ls-lt-1; i++) {
        sum = 0.0;
        for (j = 0; j < lt; j++) sum += src[i+j] * temp[j];
        res[i] = sum;
   }
}
```

D.2.3 V-Section Set Recognition: *nilm3*

The routines in this section implement the hierarchical v-section set search with intra-scale v-section lock out. This code is the central core of the transient detection algorithm. In the prototype, this code is executed three times. First, it implements a pattern search on the finest time scale, and is run exactly as presented here. The second and third runs implement the pattern search, or discriminating branch, on the next two coder stages in the tree-structured decomposition. For these two runs, slight alterations in the code, listed below, tailor the search for the appropriate coder stage in the tree. For example, the variable ASIZE, the width of the input data vector, is halfed for each successive run of the code. This alteration is necessary because each coder stage in the prototype works with half the number of input points of the previous stage.

```
#include <spox.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

#define ASIZE   330  /* 165 for M = 2, 82 for M = 1 */
#define THRESH  0.25  /* 0.3 for M = 1 */
#define SWEEP   5

smain()
{
        char      work[50];
        int       loc;
        int       *lockout;
        int       locks;
        int       i;
```

```
int       j;
int       TRAN;
int       firsttime;
int       flag;
int       flag2;
int       flag3;
float     cnt;
float     num1;
float     num2;
float     *anach1;
float     *anach2;
float     *res;
float     *res2;
float     *res3;
float     *res4;
float     *sres;
float     *sres2;
float     *sres3;
float     *sres4;
float     irp1[10];
float     irp2[40];
float     irq1[15];
float     itp1[10];
float     imp1[10];
float     imp2[25];
float     imq1[10];
float     imq2[25];
FILE      *pf;

/* allocate space for the input data */
anach1 = (float *) malloc(ASIZE*sizeof(float));
anach2 = (float *) malloc(ASIZE*sizeof(float));
res    = (float *) malloc(ASIZE*sizeof(float));
res2   = (float *) malloc(ASIZE*sizeof(float));
res3   = (float *) malloc(ASIZE*sizeof(float));
res4   = (float *) malloc(ASIZE*sizeof(float));
sres    = (float *) malloc(ASIZE*sizeof(float));
sres2   = (float *) malloc(ASIZE*sizeof(float));
sres3   = (float *) malloc(ASIZE*sizeof(float));
sres4   = (float *) malloc(ASIZE*sizeof(float));
lockout= (int *)   malloc(ASIZE*sizeof(int));


/* Read the input data file for this coder stage in the tree */
/* file = data1.dat for M = 2, data2.dat for M = 1           */
pf = fopen("data.dat","r");
for (i = 0; i < ASIZE; i++) {
        fscanf(pf,"%f",&num1);
        fscanf(pf,"%f",&num2);
```

```
        anach1[i] = num1;
        anach2[i] = num2;
}
fclose(pf);

/* Open the file that contains the v-section templates */
if ((pf = fopen("temp.dat","r")) == 0) {
                printf("ERROR ==> Cannot read temp.dat\n");
                exit(0);
}

/* Read each v-section template into an array */
/* FIRST: Read the rapid start v-sections, two in real  */
/* power, one in reactive power.                        */
for (i = 0; i < 6; i++)  { fscanf(pf,"%f",&num1); irp1[i] = num1;}
for (i = 0; i < 31; i++) { fscanf(pf,"%f",&num1); irp2[i] = num1;}
for (i = 0; i < 10; i++) { fscanf(pf,"%f",&num1); irq1[i] = num1;}
/* SECOND: Read the single instant start v-section in real power */
for (i = 0; i < 9; i++)  { fscanf(pf,"%f",&num1); itp1[i] = num1;}
/* THIRD:  Read the four induction motor v-sections, two in */
/* real power, two in reactive power.                       */
for (i = 0; i < 9; i++)  { fscanf(pf,"%f",&num1); imp1[i] = num1;}
for (i = 0; i < 21; i++) { fscanf(pf,"%f",&num1); imp2[i] = num1;}
for (i = 0; i < 7; i++)  { fscanf(pf,"%f",&num1); imq1[i] = num1;}
for (i = 0; i < 21; i++) { fscanf(pf,"%f",&num1); imq2[i] = num1;}
fclose(pf);

/* Open a file to record any contacts */
/* File = report1.dat M = 2, report2.dat, M = 1 */
pf = fopen("report.dat","w");


/* Conduct the hierarchical v-section pattern search with */
/* lock out on this scale.  Start with the most complicated */
/* pattern, finish with the least complicated.              */
/* SEARCH FOR INDUCTION MOTOR PATTERN */
locks = 0;
/* Perform shape and amplitude transversal filter checks for */
/* first induction motor v-section.                          */
ampconv(anach1,imp1,ASIZE,9,res);  threshold(res,ASIZE-10);
shapeconv(anach1,imp1,ASIZE,9,sres);  threshold(sres,ASIZE-10);
firsttime = 1;
for (i = 0; i < ASIZE-10; i++) {
    if (fabs(res[i] - 1.0) < THRESH & fabs(sres[i]-1.0) < THRESH) {
         num1 = fabs(res[i] - 1.0); loc = i;
         for (j = 1; j < SWEEP; j++) { /* check for strongest return*/
                 if (fabs(res[i+j]-1.0) < num1) {
                         num1 = fabs(res[i+j]-1.0);
                         loc = i+j;
```

```
                }
            }
            i = loc;
            if (firsttime == 1) {
               /* if the first matches, check for other v-sections */
               ampconv(anach1,imp2,ASIZE,21,res2);
               threshold(res2,ASIZE-22);
               ampconv(anach2,imq1,ASIZE,7,res3);
               threshold(res3,ASIZE-8);
               ampconv(anach2,imq2,ASIZE,21,res4);
               threshold(res4,ASIZE-22);

               shapeconv(anach1,imp2,ASIZE,21,sres2);
               threshold(sres2,ASIZE-22);
               shapeconv(anach2,imq1,ASIZE,7,sres3);
               threshold(sres3,ASIZE-8);
               shapeconv(anach2,imq2,ASIZE,21,sres4);
               threshold(sres4,ASIZE-22);
               firsttime = 0;
            }
            flag = 0; flag2 = 0; flag3 = 0;
            for (j = -SWEEP; j < SWEEP; j++) {
                 if (fabs(res2[i+j+30]-1.0) < THRESH)
                   if (fabs(sres2[i+j+30]-1.0) < THRESH)  flag = 1;
                 if (fabs(res3[i+j] -1.0) < THRESH)
                   if (fabs(sres3[i+j] -1.0) < THRESH)    flag2 = 1;
                 if (fabs(res4[i+j+30] -1.0) < THRESH)
                   if (fabs(sres4[i+j+30] -1.0) < THRESH) flag3 = 1;
            }
            /* if all v-sections match, record an event */
            if (flag == 1)
                if(flag2 == 1)
                    if (flag3 == 1) {
                            fprintf (pf,"Contact MOTOR time: %d\n",i+4);
                            lockout[locks] = i; locks++;
                            i += SWEEP;
                    }
    }
}


/* SEARCH FOR RAPID START PATTERN */
ampconv(anach1,irp1,ASIZE,6,res);  threshold(res,ASIZE-7);
shapeconv(anach1,irp1,ASIZE,6,sres);  threshold(sres,ASIZE-7);
firsttime = 1;
for (i = 0; i < ASIZE-7; i++) {
    flag = 0;
    if (locks > 0) {
            flag = 0;
```

```
            for (j = 0; j < locks; j++)
                    if (abs(lockout[j]-i) <= SWEEP) flag = 1;
    }
    if (flag == 0) {
    if (fabs(res[i] - 1.0) < THRESH & fabs(sres[i] - 1.0) < THRESH) {
         num1 = fabs(res[i] - 1.0); loc = i;
         for (j = 1; j < SWEEP; j++) { /* check for strongest return*/
                 if (fabs(res[i+j]-1.0) < num1) {
                         num1 = fabs(res[i+j]-1.0);
                         loc = i+j;
                 }
         }
         i = loc;
         if (firsttime == 1) {
            ampconv(anach1,irp2,ASIZE,31,res2);
            threshold(res2,ASIZE-32);
            ampconv(anach2,irq1,ASIZE,10,res3);
            threshold(res3,ASIZE-11);
            shapeconv(anach1,irp2,ASIZE,31,sres2);
            threshold(sres2,ASIZE-32);
            shapeconv(anach2,irq1,ASIZE,10,sres3);
            threshold(sres3,ASIZE-11);
            firsttime = 0;
         }
         flag = 0; flag2 = 0;
         for (j = -SWEEP; j < SWEEP; j++) {
                if (fabs(res2[i+j+200]-1.0) < THRESH)
                   if (fabs(sres2[i+j+200]-1.0) < THRESH) flag = 1;
                if (fabs(res3[i+j+80] -1.0) < THRESH)
                   if (fabs(sres3[i+j+80] -1.0) < THRESH) flag2 = 1;
         }
         if (flag == 1)
                 if (flag2 == 1) {
                         fprintf (pf,"Contact RAPID time: %d\n",i+3);
                         lockout[locks] = i; locks++;
                         i += SWEEP;
                 }
    }
    }
}


/* SEARCH FOR INSTANT START PATTERN */
ampconv(anach1,itp1,ASIZE,9,res);  threshold(res,ASIZE-10);
shapeconv(anach1,itp1,ASIZE,9,sres);  threshold(sres,ASIZE-10);
for (i = 0; i < ASIZE-10; i++) {
    flag = 0;
    if (locks > 0) {
            flag = 0;
```

```
                for (j = 0; j < locks; j++)
                        if (abs(lockout[j]-i) <= SWEEP) flag = 1;
            }
            if (flag == 0) {
            if (fabs(res[i] - 1.0) < THRESH & fabs(sres[i] - 1.0) < THRESH) {
                        fprintf (pf,"Contact INSTANT time: %d\n",i+3);
                        i += SWEEP;
            }
            }
        }


        fprintf (pf,"none\n");


        fclose(pf);
        free(anach1);
        free(anach2);
        free(res);
        free(res2);
        free(res3);
        free(res4);
        free(sres);
        free(sres2);
        free(sres3);
        free(sres4);
        free(lockout);



}


/* This routine performs an amplitude transversal filtering operacion */
/* on the vector src with the tap vector temp.                        */
ampconv(src,temp,ls,lt,res)
float *src;
float *temp;
int ls;
int lt;
float *res;
{
   int i;
   int j;
   float sum;

   for (i = 0; i < ls-lt-1; i++) {
        sum = 0.0;
        for (j = 0; j < lt; j++) sum += src[i+j] * temp[j];
```

```
        res[i] = sum;
    }
}

/* This routine performs a shape transversal filtering operation     */
/* on the vector src with the tap vector temp.                       */
shapeconv(src,temp,ls,lt,res)
float *src;
float *temp;
int ls;
int lt;
float *res;
{
    int i;
    int j;
    float sum;
    float dc;
    float ttemp[40];
    float tsrc[40];

    /* First, normalize the tap vector to unit length */
    sum = 0.0;
    for (i = 0; i < lt; i++) sum += temp[i]*temp[i];
    sum = sqrt(sum);
    for (i = 0; i < lt; i++)  {
            ttemp[i] = temp[i]/sum;
    }


    /* Second, remove the dc component from a window of */
    /* input data, and then normalize the ac component  */
    /* to unit length.                                  */
    /* Finally, perform convolution and store result in */
    /* vector res.                                      */
    for (i = 0; i < ls-lt-1; i++) {
         dc = 0.0;
         for (j = 0; j < lt; j++) dc += src[i+j];
         dc = dc/((float) lt);
         for (j = 0; j < lt; j++) tsrc[j] = src[i+j] - dc;
         sum = 0.0;
         for (j = 0; j < lt; j++) sum += tsrc[j]*tsrc[j];
         sum = sqrt(sum);
         for (j = 0; j < lt; j++) tsrc[j] = tsrc[j]/sum;
         sum = 0.0;
         for (j = 0; j < lt; j++) sum += tsrc[j] * ttemp[j];
         res[i] = sum;
    }

}
```

```
/* Discretize the values in vector src to 0 or 1 to determine */
/* a match.                                                   */
threshold(src,ls)
float *src;
int ls;
{
   int i;
   int j;

   for (i = 0; i < ls; i++) {
           if (src[i] < 0.0) src[i] = 0.0;
           if (fabs(src[i] - 1.0) < THRESH) src[i] = 1.0;
           else src[i] = 0.0;
   }
}
```

D.2.4 Inter-Scale V-Section Lock Out: *lock*

The following routines execute on the PC after all of the DSP components are finished. This code performs an inter-scale v-section lock out. It collates any events detected at every coder stage, and writes a final report of any events or contacts to the mass storage device on the PC. This final report is displayed by the *NILMscope* user interface in the contact window on the PC monitor.

```
#include "sim2.h"

#define SWEEP 25

main()
{
   char work[100];
   int i,j, cons1, cons2, cons3, flag, type;
   int motor, rapid, instant, L, T;
   int scale1[2][100], scale2[2][100], scale3[2][100];
   float num1;
   FILE *d1,*d2,*d3,*fopen();

   /* Open and read the contact reports from every coder stage. */
   if ((d1 = fopen("report.dat","r")) == 0) printf ("Error1\n");
   if ((d2 = fopen("report1.dat","r")) == 0) printf ("Error2\n");
   if ((d3 = fopen("report2.dat","r")) == 0) printf ("Error3\n");
   flag = 1; cons1 = 0; cons2 = 0; cons3 = 0; L = 0; T = 1;

   while(flag == 1) {
           fscanf(d1,"%s",work);
```

```
        if (stricmp(work,"none") == 0) flag = 0;
        else {
                fscanf(d1,"%s",work);
                scale1[L][cons1] = trantype(work);
                fscanf(d1,"%s",work); /* read time: */
                fscanf(d1,"%d",&i);
                scale1[T][cons1] = i;
                cons1++;
        }
}
flag = 1;
while(flag == 1) {
        fscanf(d2,"%s",work);
        if (stricmp(work,"none") == 0) flag = 0;
        else {
                fscanf(d2,"%s",work);
                scale2[L][cons2] = trantype(work);
                fscanf(d2,"%s",work); /* read time: */
                fscanf(d2,"%d",&i);
                scale2[T][cons2] = i*2;
                cons2++;
        }
}
flag = 1;
while(flag == 1) {
        fscanf(d3,"%s",work);
        if (stricmp(work,"none") == 0) flag = 0;
        else {
                fscanf(d3,"%s",work);
                scale3[L][cons3] = trantype(work);
                fscanf(d3,"%s",work); /* read time: */
                fscanf(d3,"%d",&i);
                scale3[T][cons3] = i*4;
                cons3++;
        }
}

/* Perfrom the inter-scale v-section lock out */
for (i = 0; i < cons1; i++) {
        for (j = 0; j < cons2; j++) {
                if (abs(scale1[T][i] - scale2[T][j]) < SWEEP) {
                   if (scale1[L][i] > scale2[L][j]) scale2[T][j]=-1;
                   else if (scale1[L][i] < scale2[L][j]) scale1[T][i]=-1;
                   else if (scale1[L][i] == scale2[L][j]) scale2[T][j]=-1;
                }
        }
        for (j = 0; j < cons3; j++) {
                if (abs(scale1[T][i] - scale3[T][j]) < SWEEP) {
                   if (scale1[L][i] > scale3[L][j]) scale3[T][j]=-1;
```

```
                    else if (scale1[L][i] < scale3[L][j]) scale1[T][i]=-1;
                    else if (scale1[L][i] == scale3[L][j]) scale3[T][j]=-1;
                }
        }
}


for (i = 0; i < cons2; i++) {
        for (j = 0; j < cons3; j++) {
                if (abs(scale2[T][i] - scale3[T][j]) < SWEEP) {
                    if (scale2[L][i] > scale3[L][j]) scale3[T][j]=-1;
                    else if (scale2[L][i] < scale3[L][j]) scale2[T][i]=-1;
                    else if (scale2[L][i] == scale3[L][j]) scale3[T][j]=-1;
                }
        }
}

fclose(d1); fclose(d2); fclose(d3);

/* Write the final report to the mass storage device. */

d1 = fopen("freport.dat","w");
fprintf (d1,"Fine-Scale:\n");
for (i = 0; i < cons1; i++) {
    if (scale1[T][i] >= 0) {
        strantype(scale1[L][i],work); num1 = (float) scale1[T][i];
        sprintf (work,"%s--Time:-%.2f",work,num1/180.0);
        fprintf(d1,"%s\n",work);
    }
}
fprintf(d1,"end\n");

fprintf (d1,"Mid-Scale:\n");
for (i = 0; i < cons2; i++) {
    if (scale2[T][i] >= 0) {
        strantype(scale2[L][i],work); num1 = (float) scale2[T][i];
        sprintf (work,"%s--Time:-%.2f",work,num1/180.0);
        fprintf(d1,"%s\n",work);
    }
}
fprintf(d1,"end\n");

fprintf (d1,"Coarse-Scale:\n");
for (i = 0; i < cons3; i++) {
    if (scale3[T][i] >= 0) {
        strantype(scale3[L][i],work); num1 = (float) scale3[T][i];
        sprintf (work,"%s--T:-%.3f",work,num1/180.0);
        fprintf(d1,"%s\n",work);
    }
```

```
   }
   fprintf(d1,"end\n");
   fprintf(d1,"finis\n");

   fclose(d1);


}

/* Converts the contact report to numerical values for easy checking */
trantype(work)
char *work;
{
   if (stricmp(work,"motor") == 0) return(3);
   else if (stricmp(work,"rapid") == 0) return(2);
   else if (stricmp(work,"instant") == 0) return(1);
   else {
          printf ("ERROR ==> Unknown tran type\n");
          return(0);
   }
}

/* Converts numerical codes to strings for presentation to the user */
strantype(num,string)
char *string;
int num;
{
  if (num == 1) strcpy(string,"Instant");
  else if (num == 2) strcpy(string,"Rapid");
  else if (num == 3) strcpy(string,"Motor");
}
```

D.3 Filter Design

The MATLAB script, *remz*, presented below, designs equiripple Type I and Type II FIR filters using the Remez multiple exchange algorithm. Also included are four short scripts called by *remz* to generate appropriate weighting and template functions, and a fifth script that contains a user entered template scaling. The template scaling included here created the ramped error characteristic presented in Fig. 4.9.

```
function [y,f] = remz(N,wp,ws)
% function [y,f] = remz(N,wp,ws)
%
% Implements the Parks-McClellan adaptation of the Remez algorithm for
% filter design. Designs either Type I or Type II, linear phase, FIR filters.
%
```

```
% INPUTS:
% N : Number of taps in desired filter response.
% wp: Cutoff Frequency in radians, e.g., .4*pi.
% ws: Stopband Frequency in radians, e.g., .6*pi.
%
% OUTPUTS:
% y : Chebyshev polynomial coefficients.
% f : Filter tap weights (impulse response).
%
% User modifiable m-file TEMP.M contains the template for the desired
% ripple weighting.

K = 1;
N = N-1;
if abs(floor(N/2)-ceil(N/2)) < .1
     M = N/2;
     ftype = 1;
else
     M = (N-1)/2;
     ftype = 2;
end
xp = cos(wp);
xs = cos(ws);
numfine = 60*M;                  % size of fine grid
tol = .1/numfine;
xfine = linspace(-.99,.99,numfine); % Make a fine grid
if (any(xfine == xp) == 0),
   xfine = [xfine xp];
end
if (any(xfine == xs) == 0),
   xfine = [xfine xs];
end
xfine = sort(xfine);
for i = 1:length(xfine),
    if xfine(i) == xp,
        xploc = i-1;
    elseif xfine(i) == xs,
        xsloc = i+1;
    end
end
% Generate appropriate desired function and weighting function
if ftype == 2,
        Wfine = temp2(xfine,xp,xs,K)';
        Hdfine = des2(xfine,xp,xs)';
else
        Wfine = temp1(xfine,xp,xs,K)';
        Hdfine = des1(xfine,xp,xs)';
end
plot(acos(xfine),Wfine);
```

```
title('Weighting function');
pause;
% Make a coarse grid and include pass and stop band end points
xcoarse = [linspace(-.98,xs,floor(M/2)+1) linspace(xp,.98,ceil(M/2)+1)];
d = 0; iter = 0; flag = 1;

while(flag)
    iter = iter+1;
    if ftype == 2,
        W = temp2(xcoarse,xp,xs,K);
        Hd = des2(xcoarse,xp,xs)';
    else
        W = temp1(xcoarse,xp,xs,K);
        Hd = des1(xcoarse,xp,xs)';
    end
    xnew = []; xold = xcoarse;

    % Solve the matrix equations for the Remez algorithm
    nsum = 0; dsum = 0;
    for i = 1:M+1,
            A(:,i) = (xcoarse').^(i-1);
    end
    for i = 1:M+2,
            A(i,M+2) = ((-1)^(i-1))/W(i);
    end
    params = inv(A)*Hd;
    p = params(1:M+1); p = flipud(p);
    dnew = params(M+2);
    string = sprintf ('Iter %g delta %g',iter,abs(dnew));
    disp(string);
    d = dnew;

    % Search for new extrema
    Pfine = polyval(p,xfine');
    E     = Wfine.*(Hdfine-Pfine);
    aE    = abs(E);  alts = 0;

    for i = 2:length(aE)-1,
        if (aE(i)>aE(i-1)) & (aE(i) > aE(i+1)) & (xfine(i)>xp | xfine(i)<xs),
             alts = alts+1;
             xnew(alts) = xfine(i);
             Eloc(alts) = i;
        end
        if alts == M+2,
             i = length(aE);
        end
    end
```

```
[val,loc] = min(abs(xnew-xp));
if abs(val) < tol
    xnew(loc) = xp;
else
        alts = alts+1;
        xnew(alts) = xp;
        for i = 1:length(aE),
                if abs(xp-xfine(i)) < tol
                        Eloc(alts) = i;
                end
        end
end

[val,loc] = min(abs(xnew-xs));
if abs(val) < tol
    xnew(loc) = xs;
else
        alts = alts+1;
        xnew(alts) = xs;
        for i = 1:length(aE),
                if abs(xs-xfine(i)) < tol
                        Eloc(alts) = i;
                end
        end
end

if ((aE(1)>=aE(length(aE))-tol) | alts==M) & (aE(1)>aE(2)) & (alts < M+2)
    alts = alts+1;
    xnew(alts) = xfine(1);
    Eloc(alts) = 1;
end
if alts < M+2
    alts = alts+1;
    xnew(alts) = xfine(length(aE));
    Eloc(alts) = length(aE);
end


if alts < M+2,
     xfine(1),E(1)
     xfine(length(E)),E(length(E))
end
if alts < M+2
    disp('ERROR => insufficient alternations found!');
    return;
end

xcoarse = sort(xnew);
if norm(xcoarse-xold) < tol
```

```
        flag = 0;
        disp('FINAL SOLUTION:')
        pause;
    end

    % Plot the perfect filter
    angxfine = acos(xfine)/pi; axp = acos(xp)/pi; axs = acos(xs)/pi;
    semilogy(angxfine,abs(Pfine),axp,1,'o',axs,1,'*');
    title('Approximated Freq. Response');
    pause;
    plot(angxfine,abs(Pfine.*Wfine),axp,1,'o',axs,1,'*');
    title('Approximated Freq. Response');
    pause;

    % Plot any remaining error
    PE = E;  z = xsloc:xploc; z = z-z;
    PE(xsloc:xploc) = z;
    plot(angxfine,PE,axp,.1,'o',axs,.1,'*');
    hold on;
    for i = 1:alts,
         plot(acos(xfine(Eloc(i)))/pi,E(Eloc(i)),'b+');
    end;
    title('Error');
    hold off
    pause;

end

% Convert back to time-domain filter coefficients
y = p;
arg = linspace(0,2*pi,N+2);
arg = arg(1:N+1);
x = cos(arg);
if ftype == 2,
        f = real(fftshift(ifft(polyval(y,x).*cos(arg/2).*exp(j*arg/2))));
else
        f = real(fftshift(ifft(polyval(y,x))));
end
```

The next script sets up the desired filter response for Type I FIR filters.

```
function [out] = des1(x,xp,xs)

for i = 1:length(x),
  if x(i) >= xp,
     out(i) = 1;
  else
```

```
    out(i) = 0;
  end
end
```

The next script sets up the desired filter response for Type II FIR filters.

```
function [out] = des2(x,xp,xs)

for i = 1:length(x),
  if x(i) >= xp,
    out(i) = 1/cos(acos(x(i))/2);
  else
    out(i) = 0;
  end
end
```

The next script sets up the error weighting for Type I FIR filters.

```
function [W] = temp1(x,xp,xs,K)

W = x./x;

Wtemp = temp(x,xp,xs);
W = W.*Wtemp;
```

The next script sets up the error weighting for Type II FIR filters.

```
function [W] = temp2(x,xp,xs,K)


W = cos(acos(x)/2);
Wtemp = temp(x,xp,xs);
W = W.*Wtemp;
```

The next script sets up a user-determined scaling for the error weighting.

```
function [Wtemp] = temp(x,xp,xs)

Wtemp = x./x;

for i = 1:length(x),
        if  acos(x(i)) <= acos(xp),
                scalefact = 300*((acos(xp)-acos(x(i)))/acos(xp)) + 50;
                Wtemp(i)  = scalefact*Wtemp(i);
        end
end
```

D.4 Resolving Filter and V-Section Templates

This section contains the numerical data for the resolving filter impulse response and v-section templates used for the experiments in Chapter 5. The FIR filter was constructed with the *remz* MATLAB script. The *NILMscope* routines in *nilm2*, the source code that implements the tree-structured decomposition, look for the filter tap data in a file named `filt.dat` on the mass storage device of the PC. The routines in *nilm3* that implement the hierarchical v-section search look for the v-section templates listed below in a file named `temp.dat`. The names of the v-section vectors listed below correspond to the associated array names in *nilm3*.

$$\text{Impulse Response} = \begin{bmatrix} 3.0226862e-02 \\ 1.1822732e-16 \\ -8.0614295e-02 \\ 2.2882580e-16 \\ 3.0907493e-01 \\ 5.0000000e-01 \\ 3.0907493e-01 \\ -2.0504727e-16 \\ -8.0614295e-02 \\ 8.2601421e-17 \\ 3.0226862e-02 \end{bmatrix}$$

The following three vectors contain the v-section templates for the rapid start fluorescent lamp bank. The first two vectors are v-sections in real power, and the third is a v-section in reactive power.

$$\text{IRP1} = \begin{bmatrix} -5.4072579e+00 \\ 3.1352998e+00 \\ 2.9683779e+00 \\ 1.3323450e+00 \\ -5.7558715e-01 \\ -1.4531776e+00 \end{bmatrix} \quad \text{IRP2} = \begin{bmatrix} -8.5187219e-01 \\ -8.0984447e-01 \\ -8.6237338e-01 \\ -7.7727357e-01 \\ -7.1422052e-01 \\ -7.6256043e-01 \\ -6.5223480e-01 \\ -5.9339371e-01 \\ -6.3542142e-01 \\ -5.0932680e-01 \\ -4.3788428e-01 \\ -4.6204275e-01 \\ -2.9077580e-01 \\ -1.4262294e-01 \\ -1.1005203e-01 \\ 1.0639874e-01 \\ 1.6313959e-01 \\ 1.1269945e-01 \\ 2.7240936e-01 \\ 3.8169060e-01 \\ 3.7012207e-01 \\ 6.1074279e-01 \\ 6.8639727e-01 \\ 5.7501579e-01 \\ 6.8639727e-01 \\ 7.0215479e-01 \\ 6.4751417e-01 \\ 8.3139404e-01 \\ 8.4926327e-01 \\ 7.4523837e-01 \\ 8.7132151e-01 \end{bmatrix} \quad \text{IRQ1} = \begin{bmatrix} -1.9066482e+01 \\ -1.9730708e+01 \\ -9.1088902e+00 \\ -2.6029544e+00 \\ -6.1027592e-01 \\ 3.3721805e+00 \\ 1.4260849e+01 \\ 7.6214876e+00 \\ 1.2932396e+01 \\ 1.2932396e+01 \end{bmatrix}$$

The next vector contains the v-section in real power for the instant start fluorescent lamp bank.

$$\text{ITP1} = \begin{bmatrix} -1.6392493e+00 \\ -8.3907236e-02 \\ 7.7494364e-01 \\ 6.5023240e-01 \\ 2.7019248e-01 \\ 5.1517186e-02 \\ 6.9932071e-02 \\ 1.5786960e-02 \\ -1.0944819e-01 \end{bmatrix}$$

The four vectors listed below contain the v-section templates for the induction motors. The first two v-sections are in real power and the second two are in reactive power.

$$\text{IMP1} = \begin{bmatrix} -1.2262559e+00 \\ -5.7848081e-01 \\ -2.0320606e-01 \\ 5.0881323e-03 \\ 2.7484854e-01 \\ 3.6393963e-01 \\ 3.7264868e-01 \\ 5.0578789e-01 \\ 4.8562987e-01 \end{bmatrix} \quad \text{IMP2} = \begin{bmatrix} 5.3348171e-01 \\ 5.2699082e-01 \\ 4.7263575e-01 \\ 3.9617178e-01 \\ 3.6068015e-01 \\ 2.9557413e-01 \\ 2.1525773e-01 \\ 1.5380477e-01 \\ 8.3831693e-02 \\ 4.5299061e-03 \\ -5.3070612e-02 \\ -1.1959444e-01 \\ -1.8145501e-01 \\ -2.3236305e-01 \\ -2.7860563e-01 \\ -3.0963782e-01 \\ -3.3600234e-01 \\ -3.6236907e-01 \\ -3.7839247e-01 \\ -3.9177965e-01 \\ -3.9968834e-01 \end{bmatrix}$$

$$\text{IMQ1} = \begin{bmatrix} -1.9587082e+00 \\ -7.3147979e-01 \\ 7.5305942e-03 \\ 4.5463624e-01 \\ 5.1023503e-01 \\ 7.8186509e-01 \\ 9.3592100e-01 \end{bmatrix} \quad \text{IMQ2} = \begin{bmatrix} 1.3465042e+00 \\ 1.3225720e+00 \\ 1.2143144e+00 \\ 8.0941118e-01 \\ 7.6361598e-01 \\ 6.6056542e-01 \\ 2.9729629e-01 \\ 2.5150110e-01 \\ 1.1618471e-01 \\ -1.4924097e-01 \\ -1.8671390e-01 \\ -3.0745495e-01 \\ -5.0001534e-01 \\ -4.8232484e-01 \\ -5.8537540e-01 \\ -6.9258709e-01 \\ -6.5927529e-01 \\ -7.2589888e-01 \\ -8.5184704e-01 \\ -7.8938458e-01 \\ -8.5184704e-01 \end{bmatrix}$$

Appendix E

Hardware Median Filter

The development in this section is excerpted from [79]. This section describes a fast hardware implementation of a median filter. The implementation described here is discussed in further detail in [104]. The performance of the filter has been demonstrated with results from a prototype. Geometrical and other mathematical interpretations of the median filter's behavior may be found in the references [5], [7], [67], and [116].

E.1 Hardware Implementation

A block diagram of a $2N+1$-point hardware filter window is shown in Fig. E.1. The actual values from the input stream X are stored in the registers labeled **data**. The "age" of each sampled value, modulo $2N+1$ counting from zero, is stored in the auxiliary register labeled **index** associated with each data register. A **window element** is a combination of the data and index registers and supporting circuitry. The top element in this design contains the largest data value in the window and the bottom element contains the smallest, with all other data in rank order in the middle elements. After an initialization in which each window element is loaded with a data value of zero and an index value ranging from zero to $2N$, the steady state operation of this filter window is controlled by a finite state machine (FSM) which has two states, *shift* and *load*.

The *shift* operation prepares the window elements to receive a new input value from the analog-to-digital converter. During the *shift* state two processes occur: the value from the *data* register of the middle element is read into an output register as the median value, and each element compares the value of its *index* register with the constant $2N$. The element which matches contains the oldest point; an index of zero indicates the newest point. The result of the index age comparison in each element is or-gated with the result of the index comparison from the element above. This composite result propagates to the element below. All elements below the oldest point, determined by the propagated index age comparison, shift the values of their sampled data and index registers up one element, eliminating the oldest point by writing over it, and freeing the bottom point.

The *load* operation performs an insertion sort to place a fresh input value from the analog-to-digital converter into its correct position in the window. The fresh input value is

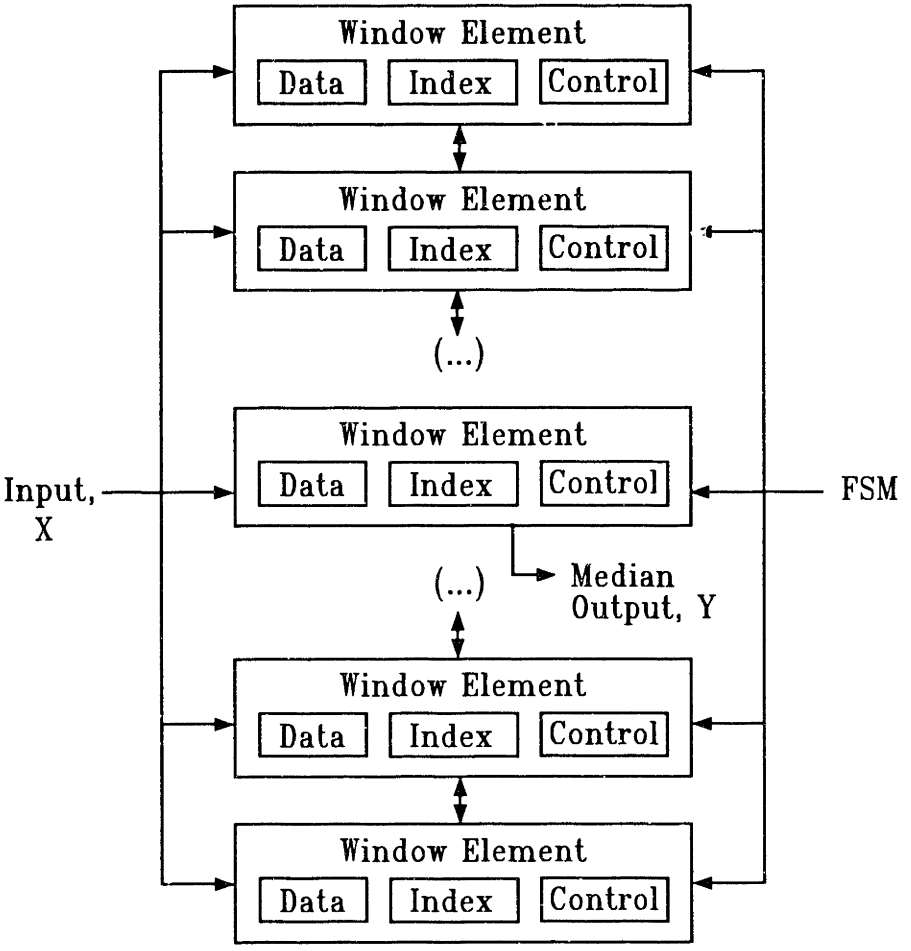


Figure E.1: Median Filter Block Diagram

broadcast to all of the window elements. Each window element compares the input value with the value in its *data* register. At some "break point" in the window, all of the elements below the break point will be less than the input value and all of the elements above the break point will be greater than the input value. The break point is unique in that the element immediately below it compares low to the input value and the element above it compares high; this condition is easily tested in parallel for all pairs of window elements. All elements below the break point shift down, opening an element in which the new data point from the input bus is inserted. The *index* register for this new point is set to zero. All other index registers are incremented by one. When the *load* operation is complete, the machine returns to the *shift* state and the process begins again. A new output point, the median data value in the current window, is made available after every two finite state machine cycles (i.e., clock cycles) regardless of the size of the window or the type of waveform being filtered.

Figure E.2 shows a schematic of a typical window element which operates under the

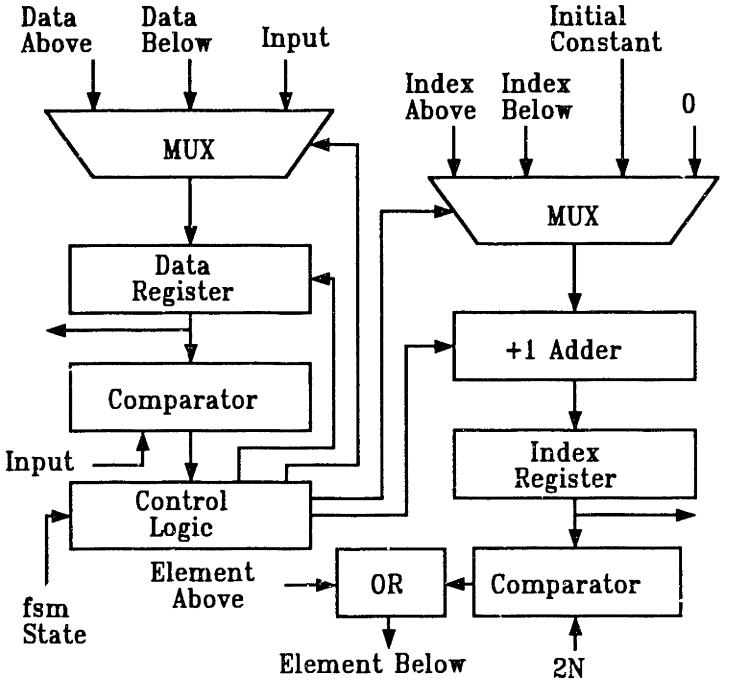


Figure E.2: Typical Window Element

two state scheme described above. Slight modifications must be made for the top and bottom elements since these elements do not have elements above or below, respectively. An action table summarizing the behavior of the top, bottom, and middle elements during the *shift* and *load* states is shown in Fig. E.3. The *propagated* signal, checked during the *shift* state for each middle element refers to the value of the propagation chain that indicates the presence of an index match with the constant $2N$ above an element.

Only the top and bottom elements, therefore, require special control coding. A filter of arbitrary size may be constructed by stacking sufficient "generic" middle elements between a top and bottom pair. For simplicity, analog signals are level shifted prior to analog-to-digital conversion in our prototype, so the lowest-valued digital sample stored in the window is zero. Hence, our prototype does not handle negative sample values digitally; all register values are positive binary numbers. The median output of the filter can be inverse shifted to restore the original bias.

E.2 Experimental Results

To study the validity and potential performance of the proposed design, a hardware prototype of a median filter of size one (a window size of three points) was constructed. First, to test the hardware sorter, the prototype was used to filter digital data stored in an EPROM.

Element	State	
	Shift	*Load*

Top	If (index = 2N+1)
	data = data below
	index = index below
		If (data <= input)
		data = input
		index = 0
		else
		index += 1

Middle	If (index = 2N+1
	or
	propagated =TRUE)
	data = data below
	index = index below
		If (data <= input
		and
		data above>input)
		data = input
		index = 0
		else if (data>input
		and
		data above>input)
		index += 1
		else
		data=data above
		index =
		index above+1

Bottom	If (index = 2N+1)
	do nothing
		If (data above > input)
		data = input
		index = 0
		else
		data=data above
		index =
		index above+1

Figure E.3: Element Action Table

A section of the digitized input waveform is shown in the top trace in Fig. E.4. The precise sample locations are indicated with + symbols. The edges and constant neighborhoods in the signal are corrupted with spike noise. The bottom trace in Fig. E.4, the output of the filter, shows that the noise is removed by the prototype filter with its three-point window.

The filter was also tested in the configuration shown in Fig. E.5. The signal Y_u was used to monitor the waveform of digitized samples fed as input to the median filter. The signal Y_f was a real-time, median filtered version of the digitized input waveform, Y_u. The filter performed successfully with a variety of input waveforms at sample rates as high as 1 megaHertz. Details of the real-time experiments may be found in [79] and [104].

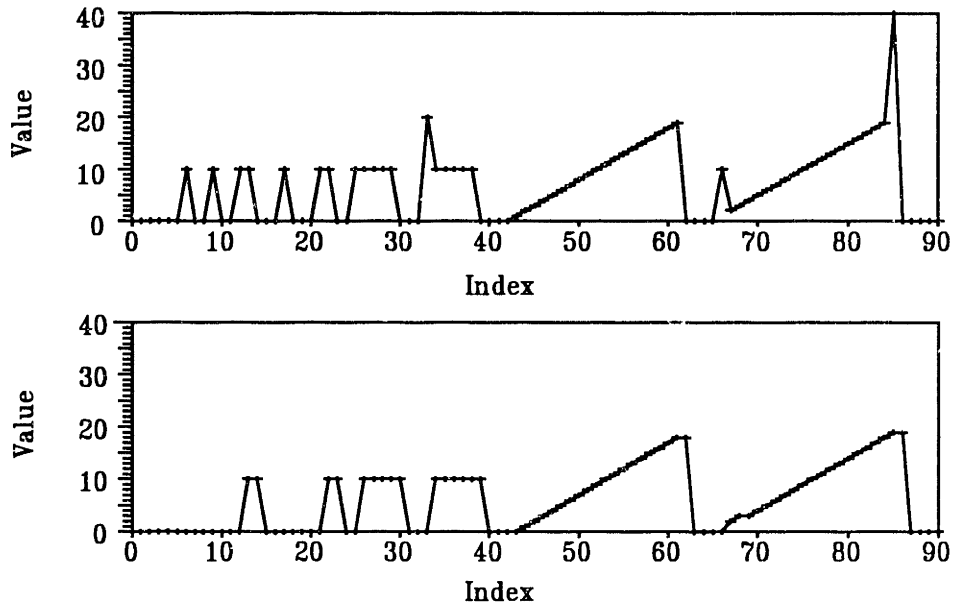


Figure E.4: Test Input and Output

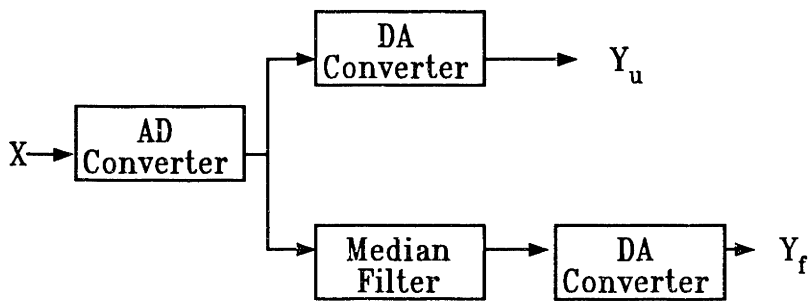


Figure E.5: Median Filter Test Configuration