

Chapter 1

Introduction to Alarm System Engineering

Alarms on industrial plants have existed since the first automatic control systems were implemented. Initially, alarms were hard-wired to some form of annunciator, and because of the considerable work involved in implementing any one alarm, there generally were only a few alarms on one piece of equipment, and at most one or two hundred alarms in any control house. And the ones that were implemented were likely thought out fairly well, both in determining what hazardous condition the alarm was detecting, and what the response should be to the alarm.

But, as anyone who has graduated from an analog system to a modern computerized system will tell you, the great disadvantage of an analog system is that it lacks flexibility – even a modest change required a significant capital expenditure and time commitment. In contrast, adding an alarm on a computerized control system is as simple as flipping a bit, and a modern distributed control system (DCS) has an almost infinite capacity for alarms, and for displaying these alarms in one format or another.

And, as could be expected, when something is cheap it results in a host of other problems that were unanticipated by the designer (such as greenhouse gases or insufficient exercise caused by cheap gasoline). Alarms are no exception – the current challenge for an engineer responsible for the alarm system is not adding the alarm, but that there are now thousands (or tens of thousands) of alarms that need to be configured in an alarm system.

And each, possibly interdependent, alarm has a list of configurable parameters to choose.

Of course it's worse for the operational staff, as they are the ones that are faced with the real-time onslaught of alarms, even when the process is in relatively stable condition.

As also could be expected, when there is an industrial problem (and particularly when industrial financial resources are available), there will be a host of solutions, companies, and technologies devoted to solving the problem, or at least appear to solve the problem. Indeed, in the last decade or so, several references and consortiums have appeared that address alarm system configuration and use. Probably the most widely used reference worldwide is the EEMUA manual *Alarm Systems, a Guide to Design, Procurement, and Management* [1]. In addition, the *Abnormal Situation Management*TM *Consortium* [2] has undertaken considerable research into alarm system design and management. Finally, ISA is drafting (and may have released at the time you are reading this) a specification entitled *Management of Alarm Systems in the Process Industries* [3].

All the above are excellent references, but they are (as indicated in their titles) about the *management* of alarm systems. What is sometimes more useful to an engineer toiling away in the metal bowels of an industrial plant is information on how to *engineer* an alarm system; that is, what are the algorithms, design guidelines, and heuristics necessary to design, analyze, and maintain an industrial alarm system?

Shown in this book are a set of technologies that answer (or at least attempt to answer) the above question. Of course, as an alarm system has a multitude of aspects, there are

also a multitude of solutions, with each one preferably constructed to best fit the problem at hand. Most of the solutions proposed here are at least somewhat mathematical – this may be a just be reflection of the authors background, but hopefully it reflects the problem as well. Alarm systems are, in some ways, just signal processing, and there is a rich heritage of sophisticated and powerful algorithms developed in this area that should be part of the control engineers’ tool chest.

Note the term control engineer above. There are many people responsible for setting an alarm (process engineers, design engineers, operators), but in some ways the best person for overseeing the alarm system is the control engineer. These people are usually most conversant with configuring the DCS, and they (unlike process and design engineers) deal with the dynamic behaviour of the plant. And they are often the first line of engineering that the console operators turn to when voicing their concerns (something I have found they are not reticent of doing). Finally, as control engineering often requires at least some mathematical finesse, the applicable mathematical and statistical techniques should hopefully be not too far outside the control engineers skill set.

The techniques shown in this book may be divided into two areas: techniques for analyzing the alarm system as a whole, and techniques for configuring the parameters on a single alarm system. All the methods require a complete history of the plant alarms, operator moves, plant measurements, and controller outputs. Fortunately, such data is readily available today – computers may make it easy to overload the operator with

alarms, but they are also splendidly able to track every change made to, and by, them into a database.

What the methods do not consider is whether an alarm should be present in the first place; this of course requires knowledge of the underlying process, and this can only be obtained by examining the design and operation of the actual process (and perhaps the sometimes bitter experience of past operational mishaps).

This book will cover some seven areas of alarm system engineering, with particular attention to the technology and algorithms that mine plant databases in order to better determine alarm settings and configurations. In brief, these areas are:

1. Database requirements for alarm system engineering.
2. Statistical quality control of alarm systems.
3. Redundant alarm detection.
4. Alarm trip point determination and evaluation.
5. Operator move / alarm correlations.
6. Alarm deadband determination.
7. Controller performance assessment using alarms.

Chapter 2

Database Requirements

Alarm System Engineering requires a long term (months or years) of two kinds of data – discrete events (alarms and operator moves) and continuous plant data (plant measurements and controller outputs). Plant distributed control systems have native storage of these items, but they are generally unsuitable for alarm analysis as they might not store enough data and/or have limited means of access from an external computer.

This is hardly a problem, as there are a multitude of databases on the market that can obtain continuous and discrete data from the DCS and store it to a readily accessible database. And the data requirements, in the scheme of things, are relatively modest – the storage requirements won't tax modern hardware, and there are rarely more than 10 or 20 users attached to the database at any one time. Discussed in this section are some aspects of database configurations that the control engineer should be aware of when mining databases for alarm information.

Measurement Sampling Rates

The field of informational theory has its theoretical beginning in 1948 when Claude Shannon at Bell labs published his seminal paper *A Mathematical Theory of Communication*. For the problem at hand, the pertinent part of this paper is what is known as Shannon's theorem, which states (briefly) that to reconstruct a signal, you need to sample at a least twice the frequency as the highest frequency in the signal. What

happens if you don't sample this fast? If your signal was a combination of sine waves, you would get something called *aliasing*, which just means that the perceived frequency would be less than the actual one (see Figure 2.1).

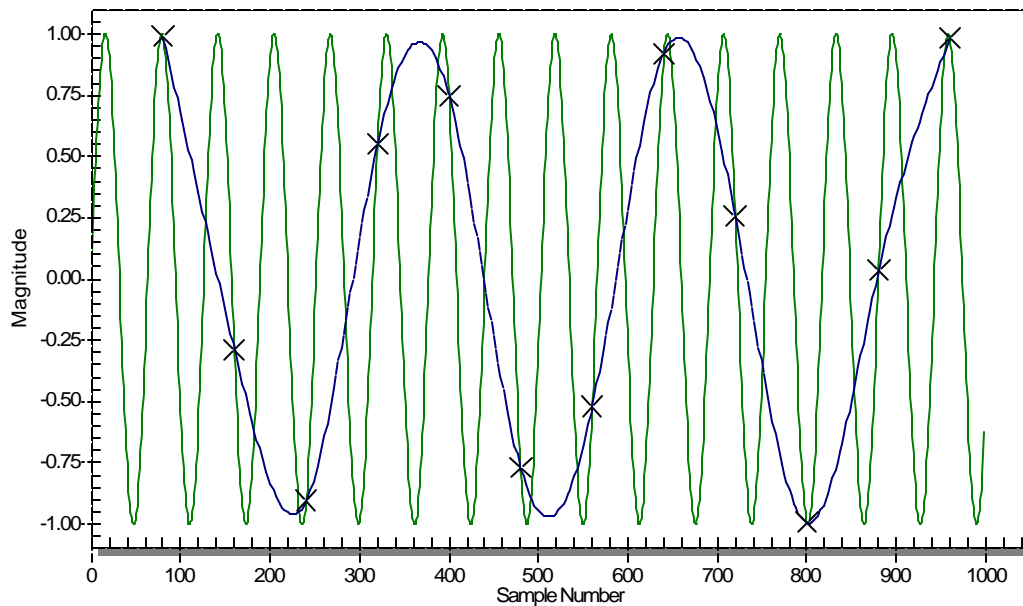


Figure 2.1. Actual sine wave (solid line) and apparent sine wave (dashed line) when sampling too slow.

A normal process signal may not (hopefully) look like a sine wave, but as it turns out, any signal may be constructed from a combination of sine waves at different frequencies and magnitudes. If you break apart a signal into its different sine waves components (using the ubiquitous Fast Fourier Transform or FFT) and then plot the magnitude of each sine wave as a function of its' frequency, you get the spectrum of the data – a typical one is shown as Figure 2.2. Now if you're sampling too slow, then the high frequency tail end of the distribution folds back the body of the main distribution, warping the shape on the lower end, and generally mucking up the analysis.

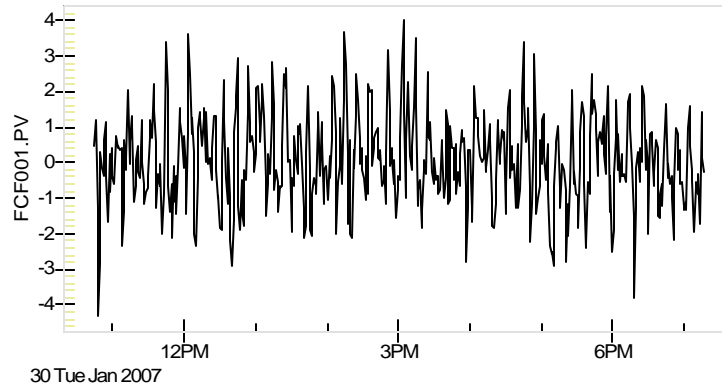


Figure 2.2a. A typical noisy data set. Hidden in the data is a 0.1 cycles/min sinusoid.

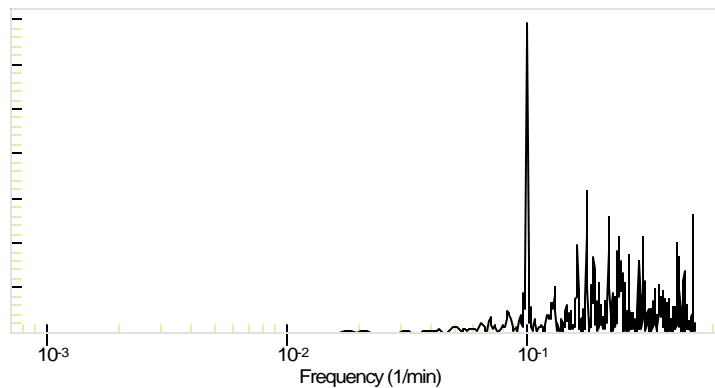


Figure 2.2b: Spectrum for the above data set. The sinusoid at 0.1 cycles/min is clearly evident.

So according to Shannon, you need to consider the highest frequencies that you can expect during a process upset, multiply by 2, and there's your required sampling frequency. And how do you determine the highest frequencies? If the tail end of your spectrum at high frequencies doesn't taper off, then you need to sample faster.

The complication is that you usually want to consider the entire spectrum of the process movement, but this is often confounded with higher frequency random measurement noise. Of course, if you have the complete spectrum (assuming you're sampling fast enough) you can design a filter that removes the measurement noise and leaves the process signal more or less intact. There are all sorts of filter design algorithms that,

while elegant, require such unknowns as cut-off frequencies and roll-off rate, none of which are obvious. They may be worthwhile if you're designing a radio, but a control engineer deals with a less-specified system, at least in terms of frequency characteristics. But, for the topic at hand, we occasionally need to design measurement filters specifically to improve the alarm system, and as shown later in this paper, there are design techniques utilizing the spectrum that help in accomplishing this task.

Note that the DCS itself is usually sampling fast enough (generally about 12 samples per second), so DCS filters generally don't get caught up in aliasing effects (unless there's a significant amount of higher frequency noise coming from, say, rotating equipment). But often the analysis is done using historized data, in which case the sampling will be slower and aliasing effects will come into play

A time domain interpretation of the above is somewhat more intuitive. All that is required is that the historized value be the same (or close to the same) as the alarm trip point at the time of the alarm. If you're sampling too slow, then chances are this won't be the case – an alarm can come and go in the space of 30 or 60 seconds. Database vendors occasionally claim that their interpolation algorithms solve this problem – they can reconstruct any signal at any time using some mathematical technique. Don't believe this – Shannon's theorem states that if you're sampling too slow, then some information is simply irretrievably lost.

Site system engineers are notorious for claiming that their systems are loaded and cannot handle faster sampling rates. This may be true in many instances, but older control engineers may recall that the common 30 or 60 second sampling rate has been in place for at least 20 years. Shown below is a indication of the increases in computer horsepower over this time period.

Attribute	1985	2007	Increase (%)
CPU Clock Speeds	486kHz	2.3GHz	373000
Disk drive capacity	10MB	1000GB	9999900
LAN bandwidth	10Mbit/s	10Gbit/s	9999
RAM	640K	4GB	624900

Table 2.1: Changes in computer and network capabilities.

Of course, this does not take into account the capacity of the DCS, which may in fact be 20 years or older. But all capital equipment gets replaced, and it's not hard to conceive of a 20 year old computer being on the replacement or upgrade list. Indeed, if the DCS is a limiting factor, alarm engineering requirements can be an important (but perhaps not the only) justification for a DCS upgrade.

If rules of thumb are required, the absolute slowest sampling time usable for alarm engineering is once every 15 seconds. Once every 5 seconds is preferable, and if you can get away with once every one or two seconds, go for it. Note that the data doesn't have to be *stored* to the database every sample interval (many historians won't store the data if it hasn't moved), but the process has to be at least *sampled* at the above time intervals.

Time Formats

Given the problems of Y2K, it is somewhat surprising to see that different manufacturers use different time formats. The ISO 8601 standard is to have the date/time formatted as YYYY-MM-DD hh:mm:ss (note the descending ordinal values), and it would make things incomparably easier if this was adopted throughout all systems. The following should be noted:

1. There is no AM or PM (which are Roman letters of course and would be different in different parts of the world). The hh value is 24-hour time (i.e., it ranges from 0 to 24).
2. The standard format for dates stored in SQL databases is the United States date format, i.e., MM-DD-YYYY. Always a source of confusion.
3. There is no standard for second fractions, and in fact the date format of SQL databases does not support second fractions either. Some DCS report the time accurate to the millisecond (which is helpful for analysis of equipment trip events), but the fractional second will be stripped off when stored to the database. Preferably, the second fraction may be stored in another field in the database.

Time Synchronization

If alarms and operator moves were considered alone, there might not be a need to maintain an accurate DCS time. But if the information is combined with the measurement and output data, plus perhaps information from other computer systems (emergency shut-down systems etc.), then having an accurate DCS time is paramount for

any analysis. The best way to do this is to have a program or programs that automatically synchronize all computers in the plant to the local time.

Daylight Savings Time

Daylight savings time represents a much more difficult software problem than Y2K as there is an hour gained and lost on different days in the year. And the day that this happens on changes from year to year and location to location. And some computers automatically update their clocks at the right time, others when the system engineer gets around to it, and some never get updated for daylight saving. Again, to be consistent, the DCS clock should be automatically updated by a program of some sort so that it *always* displays the right time.

But of course the real plant doesn't gain or lose an hour, so the database must account for the data during these duplicate or missing hours. Many databases get around this confusion by storing the data in Greenwich Mean Time or GMT (in some ways, England is still the center of the universe), making the conversion from/to local time when the data is stored or retrieved. But alarms and operator moves are based on DCS time, and the DCS follows local time (this is somewhat of a standard – local computers use local time). As most event capture databases just record the DCS time, expect to see odd effects on the days that daylight savings shift things around.

Batch Time

For any site that uses batch processes, the actual time of the alarm is less important than the elapsed time since the batch or phase in the batch started. For this reason, it's handy to have the event capture program automatically log the time when the batch or phase started. This would of course be in addition to the actual time, as the rest of the plant databases are likely time stamped with the actual time only.

This is less important in continuous plants, but even these have batch-like systems, i.e., catalyst regeneration, filter cleaning, exchanger fouling, etc. For such systems, having an alarm time based on the elapsed time from some event can be helpful in diagnosing problem alarms.

Chapter 3

Statistical Quality Control of Alarm Systems

The first step in auditing an alarm system is measuring the number of alarms that are presented to the operator. Several numerical indices, such as alarm per hour or day (which represents overall operator loading) or percentage of time in where the number of alarms exceeds some value (which represents alarm floods) have been proposed; shown below are some industry recommendations on acceptable limits for these values.

Metric	Acceptable	Manageable
Alarms per day	150	300
Alarms per hour	6	12
Alarms per 10 minutes	1	2
Fraction of hours with > 30 alarms	1%	
Fraction of 10 minute intervals with > 5 alarms	1%	
Maximum number of alarms received in any 10 minute interval	10	20

Table 3.1. Industry Recommendations on alarm counts.

When plotted as a function of time, the number of alarms can be used in essentially the same way as a statistical quality control run chart, with the above recommendations taking the place of the more familiar 2 and 3 sigma limits of statistical quality control run charts.

But statistical quality control has had something of a checkered past in the process industries – a great many statistical quality control initiatives (sometimes named other things like *Total Quality Control* or *Continuous Improvement*) fell under the wayside for one reason or another. Perhaps these initiatives were abandoned due to the usual corporate inertia, but another possible reason is that the statistical quality control solution didn't quite fit the problem at hand.

The techniques of statistical quality control were developed for the discrete part manufacturing industries, where there turns out to be a good mesh between statistical theory and the practical aspects of manufacturing a part. But continuous processes are different for the following reasons:

1. There is generally a cost involved in modifying a discrete process, so much of SQC is concerned with determining if the plant actually needs to be adjusted. In comparison, continuous processes are adjusted by changing a valve output, a procedure that has very little, if any, cost.
2. Many of the measurements for discrete processes are the actual product specifications (dimensions, strength, etc.). In contrast, there might be thousands of measurements in a chemical plant, but only 2 or 3 of them are product specs (and these are at the tail end of the process).
3. There are few dynamics in a discrete process, so that changes immediately show up and are persistent to the next change. Continuous processes have a time element associated with the changes, so that a disturbance or input has a gradual effect on the process.

4. The underlying statistical distribution of errors in a discrete process is more-or-less normally distributed. This is occasionally, or perhaps even usually, true in continuous processes. Moreover, SQC techniques are robust to mild non-normality in the underlying distributions. If, however, the data displays an extreme non-normal distribution, SQC can result in serious misinterpretation.

Now consider the above reasons when applied to alarms. It requires an effort to change an alarm system, the number of alarms is a relevant quantity to operations, and there really aren't any "dynamics" in an alarm system (i.e., you wouldn't model it with a time constant and deadtime). So alarm SQC techniques satisfy the first 3 objections.

The fourth one is considerably more problematic. Figure 3.1 is an alarm SQC chart for a typical industrial process. Most of the time there are few alarms, but there are occasional outbursts of high alarm activity. These outbursts make interpretation difficult, as it is difficult to see a trend in the data.

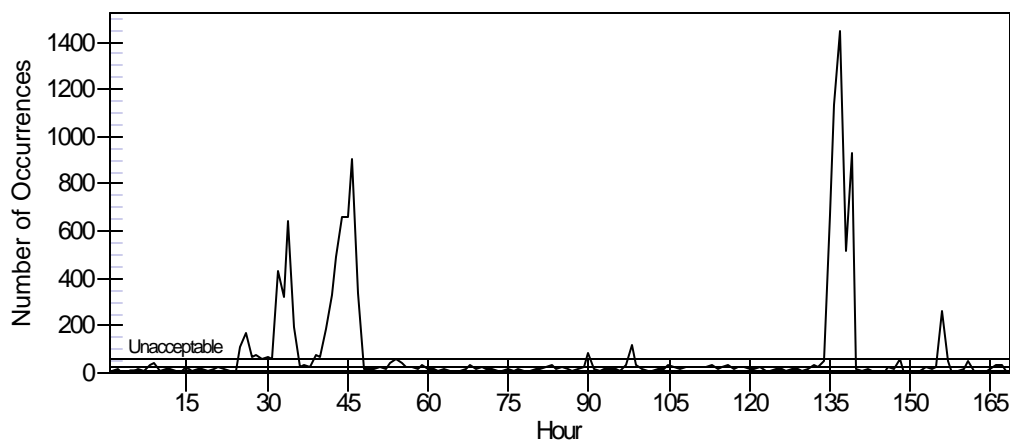


Figure 3.1: Alarm count as a function of time. The horizontal lines on the chart are industry recommendations of manageable, acceptable, etc. amounts.

From a statistical viewpoint, the alarm system is unsuitable for an SQC analysis as the underlying distribution of the data is not Gaussian, but rather the alarm counts follow more closely a Pareto distribution. In fact, if you do a standard goodness of fit test (check your undergraduate statistics test on what this is) on the above data, the index turns out to be 1440. For this data set, the critical value is 40, and as a number greater than the critical value indicates non-normality, then clearly this is a problem.

But there is a nice solution to this problem, in both an operational and statistical sense. Simply replacing each sequence of chattering alarms in the data set by a single alarm occurring at the beginning of the chattering sequence results in the run chart as shown in Figure 3.2. For this case, the goodness of fit test index is 84, which is much closer to normal, and indeed it is much easier to discern trends in this chart (and the metric in general meets industry recommendations).

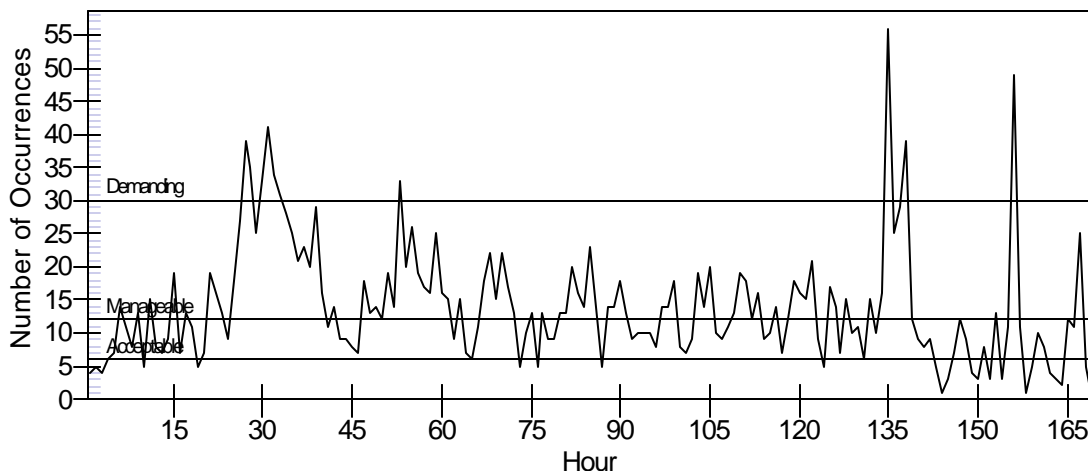


Figure 3.2. A simple transformation of the data results in an SQC chart that is much easier to use.

There are two operational justifications for replacing a sequence of chattering alarms by a single alarm:

1. On most alarm annunciators, a chattering alarm will appear largely the same as a single alarm. The only difference is that a chattering alarm will cause the annunciator to flash on and off (as the measurement goes into and out of alarm), but this can be similar behaviour to an unacknowledged alarm, which flashes on and off until it is acknowledged.
2. There is only one operator response required for a chattering alarm (as opposed to multiple responses required for an equivalent number of non-chattering alarms).

So replacing a sequence of chattering alarms by a single alarm also makes operational sense – certainly an operator wouldn't find a sequence of 50 alarms of the same type that happened in rapid succession as onerous as 50 different alarms.

Note there is a nice duality between statistical quality control theory and the operational considerations – the transformation required to satisfy the requirements of SQC also have a strong operational justification.

It is also necessary to track the chattering alarm events as these do have some effect on operations and data analysis. Since a chattering alarm can be considered to be an equipment failure, it is expected that these will follow a Poisson distribution. As indicated in Figure 3.3b, this is largely the case for the example data set. There are some techniques available for statistical quality control of Poisson or Pareto processes, but these are not as widely used (perhaps because they are not as developed) as statistical quality control of normally-distributed processes.

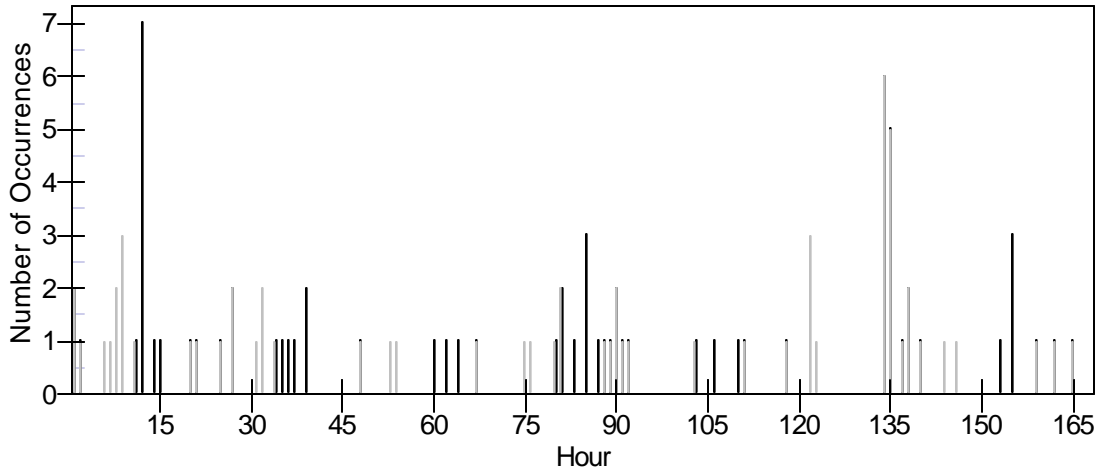


Figure 3.3a: Histogram of chattering alarm events.

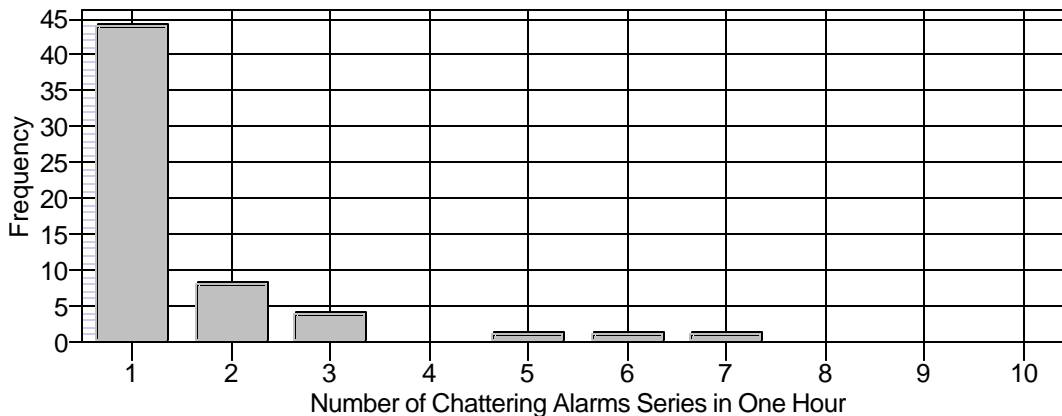


Figure 3.3b: The distribution of the chattering alarm events shown in Figure 3.3a. The distribution follows a typical Poisson shape.

Probably most alarm management software doesn't remove the chattering alarms from the analysis, with the expected result that improvements are mostly, or entirely, projects to remove chattering alarms. This is to be expected, as that is what the metric is measuring (and it's common to "manage to the metric"), but reducing the number of alarms by say 80% doesn't really improve the alarm system by this amount. And it might take engineering effort away from tasks that would make a difference to the operator – such things as determining better alarm settings or removing unnecessary alarms entirely.

Shown in the next sections are some techniques for removing the number of alarms *and* improving the quality of the alarm system.

Chapter 4

Detection of Redundant Alarm Groups

A consequence of designers adding alarms in isolation is that there are often redundant sets of alarms – i.e., alarms that occur close in time to one another most or all of the time, and essentially indicate the same process fault. Since redundant alarms add to operator loading and violate one of the tenets of a well designed alarm system (that each fault should only generate a single alarm), there is considerable incentive to find and eliminate any redundant alarms.

Given a database of alarm occurrences, the problem becomes one of determining which of the hundreds of alarms, and thousands of alarm occurrences, are redundant.

Compounding this problem is that the definition of redundancy is somewhat fuzzy – operators would consider an alarm redundant if it happened *close* in time to another alarm, *most* of the time.

To formulate this problem, the data may be represented by a matrix, where each row of the matrix represents a unique alarm (i.e., FCF001.PVHI), and each column of the matrix represents a time unit (i.e., a minute). An alarm occurring at time j for tag i would be represented by a one in the appropriate matrix element, otherwise the value would be zero. An example is shown as equation #1.

$$\begin{bmatrix} Alarm1 \\ Alarm2 \\ \vdots \\ AlarmN \end{bmatrix} = \begin{bmatrix} 0 & \dots & 0 & 1 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & \dots & 1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} 01/01/06 & 12:00:00AM \\ 01/01/06 & 12:01:00AM \\ \vdots & \vdots \\ 12/31/06 & 11:59:00PM \end{bmatrix} \quad (1)$$

The matrix of equation #1 would be inadequate for analysis, as two rows could be completely independent if one tag always alarmed one minute after another – in contrast, an operator may feel that the two alarms are dependent. To overcome this limitation, each unity element of the matrix, along with its row neighbors, can be replaced by a normal distribution as shown in equation #2. Here the standard deviation of the distribution represents the amount of “fuzziness” between the alarm events.

$$\begin{bmatrix} Alarm1 \\ Alarm2 \\ \vdots \\ AlarmN \end{bmatrix} = \begin{bmatrix} 0 & \dots & 0.88 & 1 & 0.88 & 0.61 & 0.33 & 0.14 & \dots & 0 \\ 0 & \dots & 0.61 & 0.88 & 1 & 0.88 & 0.61 & 0.33 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & \dots & 1 & 0.88 & 0.61 & 0.33 & 0.14 & 0.04 & \dots & 0 \end{bmatrix} \begin{bmatrix} 01/01/06 & 12:00:00AM \\ 01/01/06 & 12:01:00AM \\ \vdots & \vdots \\ 12/31/06 & 11:59:00PM \end{bmatrix} \quad (2)$$

Once the matrix has been represented in this form, some sort of algorithm must be applied to determine which rows are (at least mostly) redundant. As it turns out, this is a standard problem in linear algebra, and the solution (called Singular Value Decomposition or SVD) is mostly well-developed.

It's a standard problem because matrices with semi-redundant rows and/or columns are known to cause problems when the matrix is inverted. Certainly if any rows or columns of an n by n matrix are redundant, you can't invert the matrix (effectively, this would be asking to solve n variables use $n-m$ equations). But even mathematicians have long recognized that the situation is never quite black and white, in that there are often rows or columns of a matrix that, while not exactly the same, are close to the same. And these near-redundant rows or columns can play havoc with any results from the matrix inversion – parameter estimates will have large error bounds, LP's will bounce from constraint to constraint, process controllers will be very sensitive to model mismatch, etc.

Almost magically, the SVD techniques will solve these difficulties. Well, not quite magically, as what SVD does is solves the part of the problem that can be solved. So if your matrix has n rows, an SVD analysis will find a linear combination of $n-m$ rows that are independent, and solve for these. Except for two difficulties: 1) as mentioned, there's never a clean split between the independent and dependent matrix, so there's some user judgment as to a good value for m 2) you might specify that m rows should be removed, but SVD won't explicitly remove m of the rows – in fact the remaining $n-m$ rows just contain a linear combination of the original n rows of the matrix.

For our alarm analysis case, the rows of the matrix represent the DCS alarm points, so an SVD analysis will tell you which combinations of alarms are redundant. But often the results are ambiguous – it's always nice when the analysis says that one alarm is redundant with another alarm, but SVD will also indicate that a group of (say) five alarms may be just as easily represented by some linear combination of (say) four alarms. Perhaps this is to be expected, as the problem was set up with a bit of “fuzziness”, so its not surprising that the results might be ambiguous as well.

Some other small points of interest. The first is a comment on the scale of the problem. It's generally best to get a fairly long time period for analysis as you want the data set to cover a wide variety of plant conditions. If a year of plant data is taken, this represents a fairly large matrix (525,600 columns at a one-minute sampling) and as each row of the matrix represents a configured alarm, there might be thousands of rows. Standard analysis techniques might choke on the size of the matrix, but by and large the elements of the matrix are mostly zero, and there are analysis techniques (called sparse-matrix analysis) that can chew through these types of matrices without overloading a computer.

The second point is that chattering alarms can be very effective at “polluting” the data set. An alarm going off repeatedly will show up as being correlated with a host of other alarms, but in fact the correlation is not due to any actual process correlation. For this reason, it’s best to remove the chattering alarms, either by using the technique shown in the previous section, or by just fixing the actual alarm.

The last point is to note that this analysis is just a variation of the widely used principal component analysis (PCA) or partial least squares (PLS) techniques. The major difference is that these techniques are used to determine the *independent* dimensions of a system, here of course the analysis is used to determine the *dependent* dimensions. And PCA or PLS also has the same problem that, while computationally elegant, it’s sometimes hard to figure out the physical meaning of the resultant matrices.

Once a set of redundant alarms is identified, the question remains of what to do with them. The easy answer is just to eliminate one or more of the alarms, and this can be done in one of several ways. The alarm can be disabled of course, or the trip point can be changed so that the redundant alarm measures a truly unique event. Alternatively, logic is available on most DCS systems to suppress one alarm when another occurs (this is known as contact cut-out). Note this logic may not work if the alarms occur too close in time to each other – the DCS may not be able to cut out an alarm if it occurs within the same second as the primary alarm.

Finally, care must be taken to ensure that the DCS still gives consistent alarm information. For instance, an analysis might indicate that a low flow alarm always occurs when a pump trip alarm has occurred, but removing the low flow alarm might not be in the

operators best interest. After all, a low flow alarm is a pretty good confirmation that the pump did actually trip. And if the alarm was automatically cut-out on a pump trip alarm, then the operator would have to be aware of this logic or else would be left wondering why the low flow alarm didn't ring when in fact the flow went low.

Cutting out redundant alarms makes more sense when they are in fact measuring the same phenomenon, such as a set of reactor temperature alarms or a high/high-high alarm pair. For the reactor temperatures, the justification for the same alarm on different measurements as this can be more robust in the face of measurement failure. In this case, however, it's simpler and better to use some sort of voting logic so that only one alarm is set to the operator when some or the entire reactor gets too hot.

Chapter 5

Evaluation of Alarm Trip Point Settings

The high and low trip points (i.e., the value of the process measurement at which the alarm occurs) are generally set either by process designers or plant operators. These people may have limited information on the dynamic responses of the plant during upset. The process designer generally only has access to a steady-state simulation and knowledge of equipment hard constraints. An operator may have only experienced some upsets a few times, but is probably too busy during an alarm event to question whether the alarm settings are optimal.

Graphically, the problem may be represented as shown in Figure 5.1. The designer generally knows the normal operating point of the plant, and what value of the measurement (PV) which would result in a hazardous condition or equipment constraint. The alarm trip point value must be chosen somewhere between these two values, and it is often the case that the offset from the normal operating point is a somewhat arbitrary fraction of the total range between the normal operating point and the true hazardous point.

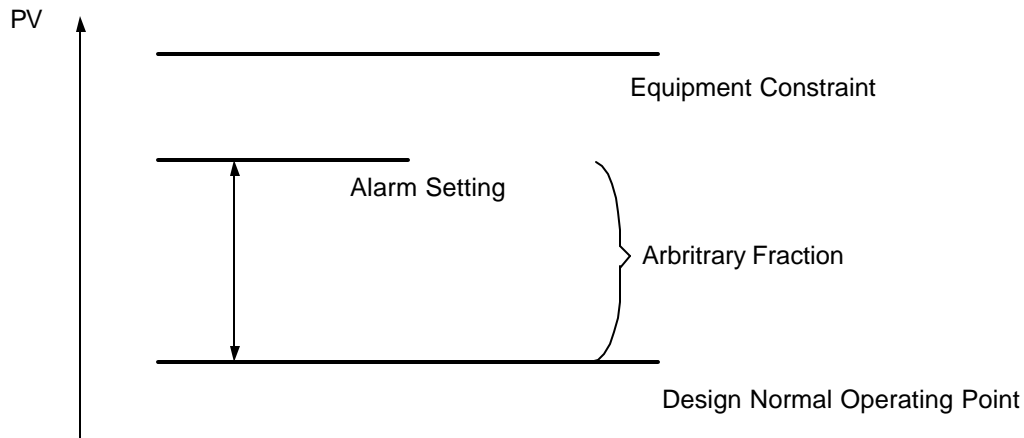


Figure 5.1: The alarm setting is usually placed somewhat arbitrarily between the normal operating point and the actual process constraint.

To determine the best value for an alarm trip point, the following two additional pieces of information must be obtained:

1. What is the normal variation of the process? Clearly, an alarm trip point should not be set so that the alarm occurs when the process will naturally return to the normal operating point.
2. What is the response time of the process, both for an operator to respond to an alarm, and for the process to respond to the operator actions? More precisely, how much further past the alarm does the measurement go when an alarm occurs, and how long does it take to get to the maximal point?

Graphically, this may be represented by the set of diagrams shown in Figure 5.2. The green region indicates the normal operating range, while the red region indicates the buffer required to manually control the process back to a safe condition. Note that the color in both these regions fades at the limits – this indicates the statistical nature of the process, as there is usually not a hard boundary between the different regions.



Figure 5.2a: The process has a suitable gap between the normal operating region and the buffer required to correct the process before an actual process constraint is hit.

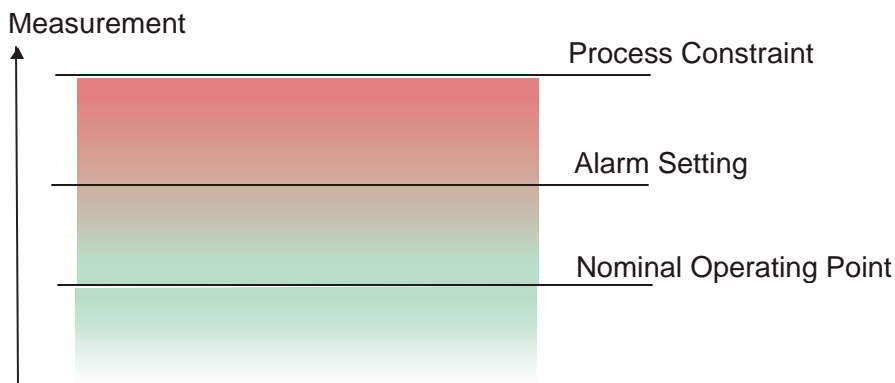


Figure 5.2b: The nominal operating point is so close to the process constraint so that either alarm trip points must be placed in the normal operating region or there is insufficient time for the plant to respond to an alarm.

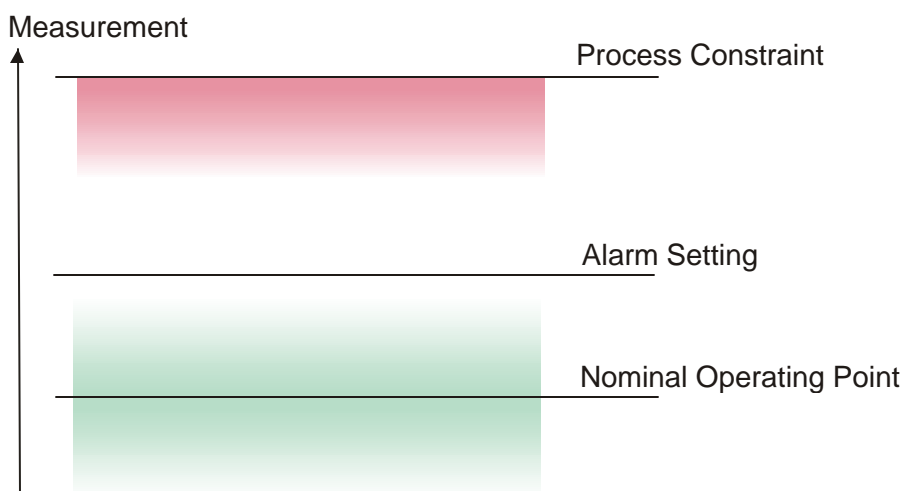


Figure 5.2c: Here the nominal operating point is far enough away from the process constraint so that there is a large area to set the alarm trip point.

Figure 5.2a represents the ideal case: the alarm is outside the normal operating region, but still provides adequate warning for the operator to manually control the process.

There is no suitable alarm trip point value for the system shown in Figure 5.2b; either the alarm must be set so that it occurs when the process is in normal operation (i.e., the automatic control system can return the process to its nominal operating point without operator assistance), or inadequate warning will be given.

The system shown in Figure 5.2c is adequate for alarm purposes, but it may also indicate that the plant is not operating at its most profitable point. There is often a large financial incentive to operate the plant as close as possible to process constraints, but operational conservativeness might result in excessive safety margins. It would be useful for an alarm analysis to indicate whether the nominal operating point may be moved closer to a constraint while still giving adequate operator warning.

The statistical or stochastic nature of the normal operating and response regions indicates that any analysis tools and results should also be presented in terms of statistical probabilities. While the analysis could be carried out using computer simulators or standard design methods, a more efficient and accurate method is just to mine the current operating data itself – after all, this contains the real plant variation and operator responses, both of which are hard to emulate in a simulator or design guideline.

First, consider a typical process response for a vessel level immediately before and after an alarm, as shown in Figure 5.3. For times before the alarm, it can be seen that the

measurement averaged the setpoint, with the usual variation around the setpoint. At some point however, the process does not return to setpoint, but rather continues increasing until the alarm occurs. Similarly, it can be seen that after alarming, the measurement continues increasing, achieving some maximal value before again returning to the normal operating point.

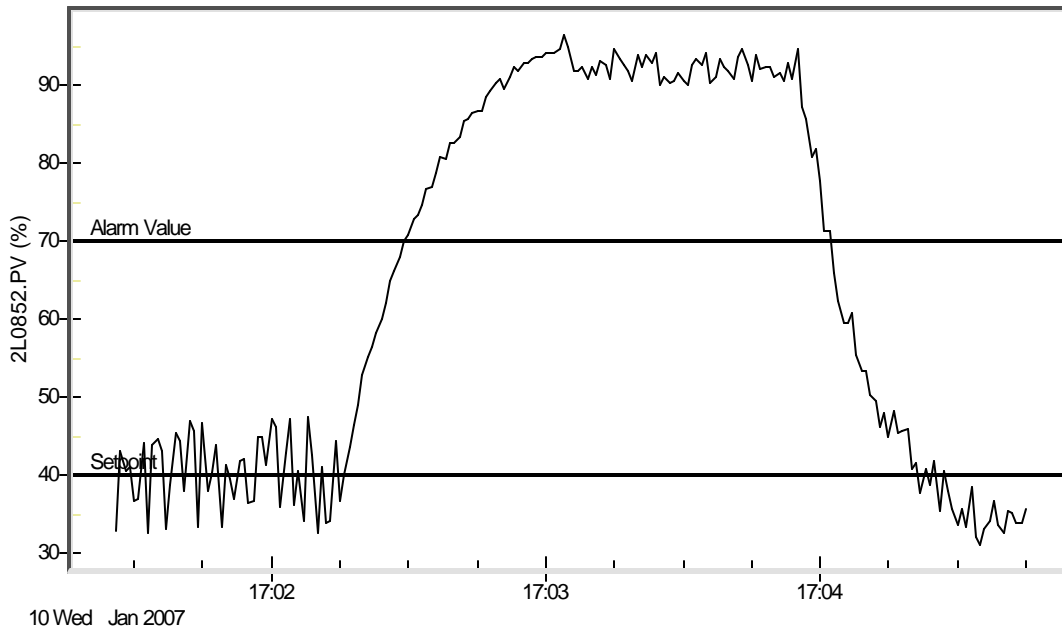


Figure 5.3: A typical response for a process going into and coming out of alarm.

From this portion of the data, one can determine at what point the process started heading into alarm, how long it took to get there, how far past the alarm did the process go, and how long did it take to get to the maximal value. Of course, one event is not enough to draw firm conclusions on the process behaviour (particularly during an upset), but it is common to find multiple alarm events in a typical data set. Combining the responses from all the alarm events will result in some general conclusions about the response of the process during an upset.

All this information may be displayed concisely in one chart, which indicates the probable response of the process as a function of the process measurement. An example of such a plot is shown as Figure 5.4. Note that Figure 5.3 shows the measurement (vertical axis) versus the time (horizontal axis). The probability plot of Figure 5.4 has the measurement as the horizontal axis, with the probability as the left vertical axis and the time to/from an alarm as the right vertical axis.

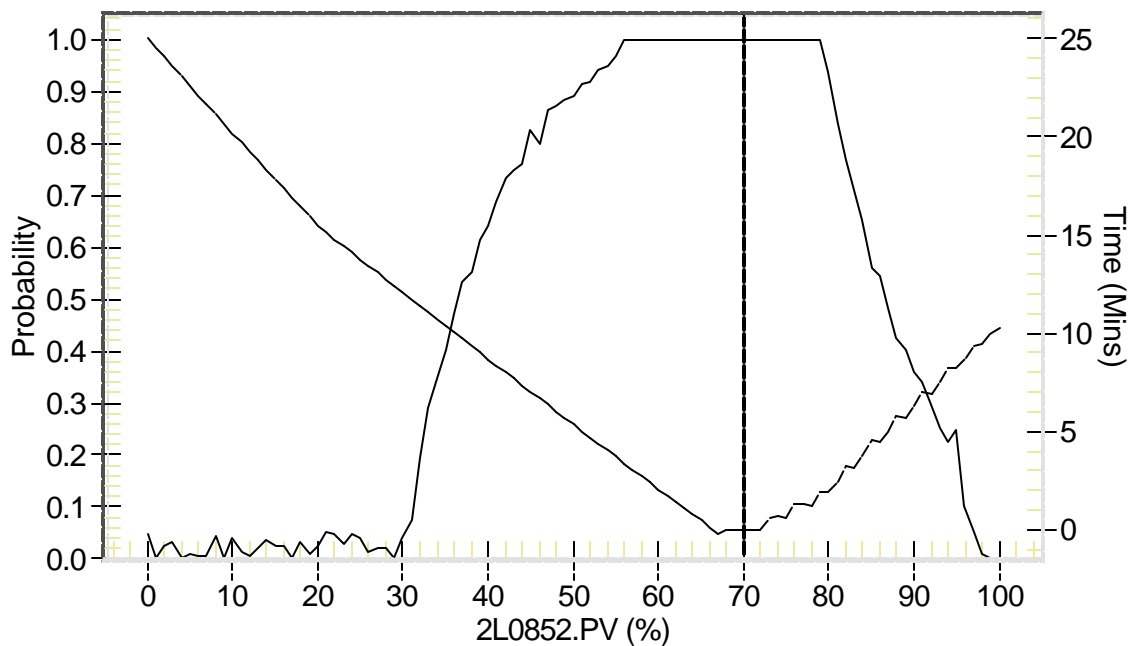


Figure 5.4: Alarm trip point probability plot.

The probability plot is a “distribution-free” method as no statistical distribution was assumed. Assumptions of normality, while perhaps justified during normal operation, may be suspect during the very non-linear operation of a process upset.

Figure 5.4 contains 4 different curves. These are:

1. The probability of the point going into alarm given that a certain PV has been attained (solid line to the left of the alarm value).
2. The probability that the process will achieve a given PV given that an alarm has occurred (solid line to the right of the alarm setting).
3. The amount of time the process will take to go into alarm (if it is in fact heading to an alarm) after achieving a specified PV (dashed line to the left of the alarm setting).
4. The amount of time taken to attain a specified PV beyond the alarm setting once an alarm has occurred (dashed line to the right of the alarm setting)

From this plot, the following can be ascertained:

1. Can the alarm be moved closer to the normal operating point without increasing the amount of "false" alarms?
2. Can the alarm be moved closer to an equipment limit without increasing the probability of hitting the limit?
3. How much additional time would the operator have to correct the process if the trip point was moved?
4. How much time does the operator take to correct the process after an alarm occurs?

For the example shown above, it is apparent that once the PV attained a value of approximately 55%, the level would *always* continue increasing until an alarm occurred. Therefore, lowering the alarm limit to 55 or 60% will not increase the number of

nuisance alarms, since once the process reached this value, it (over the given data set anyway) has always alarmed within a short time.

Of course, lowering the alarm trip point has the advantage of giving operators more warning time. Figure 5.4 indicates that lowering the alarm limit to 55% would give approximately five more minutes to respond to the abnormal situation.

The plot also shows that the operators in general were able to prevent the vessel from overflowing, as the measurement never achieved full range (105%). However, it often came close to filling – going up to 90% about 30% of the time, and taking only about five minutes to get to this value.

If it is not desired to change the alarm setting, then the probability chart of Figure 5.4 may be used in a predictive mode, with the prediction being the probability that, for a given measurement, the process is moving to an alarm state. For instance, if the level measurement is at 40%, an indication could be given to the operator that there is a 50% chance of the point alarming soon.

This is in some ways a more reasonable way to consider alarms – a normal alarm is a binary interpretation of a continuous process (either the process is in alarm or out of alarm), but in reality many processes have a continuous transition between the normal and alarm state. And because the probability chart can be easily generated from just the

process and alarm data, it is easy to automate this feature and apply it to all continuous alarms.

Chapter 6

Estimation of Alarm Deadbands

An alarm deadband is one means of reducing a chattering alarm, and logic for implementing deadbands is standard on all control systems. An alarm deadband reduces chattering as the measurement that is in alarm must pass through the deadband before it clears (and can ring again). Effectively, an alarm deadband is a high-pass *amplitude* filter; the alarm will only ring on significant changes in the measurement.

While easy to implement, there appear to be few guidelines as to how to set the value of the alarm deadband. It has been suggested that the deadband be set as a percent of the normal operating range; Table 6.1 provides values for different types of measurements.

Measurement Type	Deadband
Flow	5%
Temperature	1%
Level	2%
Pressure	2%
Composition	1%

Table 6.1: Industry recommendations for alarm measurement deadbands

Another type of deadband is based on time – an alarm is prevented from ringing for a specified time after it has already rang. This alarm is generally available on digital points, but may appear on analog points as well. Again, industry recommendations for this are:

Measurement Type	Delay Time
Flow	15 seconds
Temperature	60 seconds
Level	60 seconds
Pressure	15 seconds
Composition	120 seconds

Table 6.2: Industry recommendations for alarm time deadbands

Given the wide variation in spectrums from even one type of signal, the above should perhaps not be considered more than an initial estimate. A more complete approach would be to base the deadband on the actual measurement noise. The problem then becomes of dividing the signal into two portions: a low frequency process trend, which indicates the actual movement in the process, and a higher frequency measurement noise, which is responsible for alarm chattering. And obviously the discussion on sampling rates is pertinent here – sample slow enough and both these portions of signal will be corrupted.

Dividing a process signal into the process measurement and noise components can often be done visually to some extent, but here the field of *time series analysis* (Box and Jenkins [4]) can be profitably employed to this problem to obtain an elegant solution to this problem.

Basically, time series analysis is dynamic modeling of systems from a statistical viewpoint, recognizing that many systems have, at least to some extent, a random component. It has found wide use in a wide variety of areas, from economics (the familiar seasonally adjusted economic indicators are an example of time series analysis) to biological systems (many biological processes take time to complete, and are affected

by, say, diurnal cycles) to chemical processing systems (anyone looking at plant data can see that there appears to be a large random component that plays out through the plant over time). As control engineers are concerned with dynamic processes, time series analysis should be an integral part of the control engineers' toolkit.

The part of time series analysis of concern to deadband determination is in modeling the process signal. Traditionally, control engineers consider models to consist of the relationship between deterministic inputs and output, basically of the form:

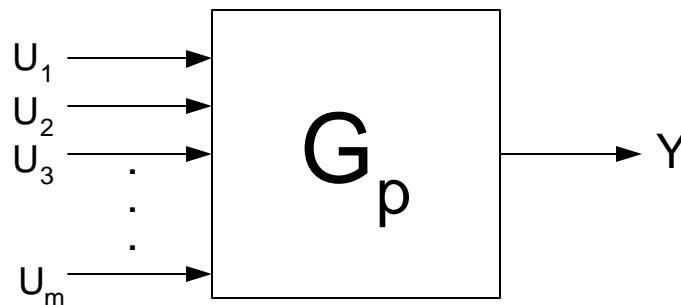


Figure 7.1: A traditional model relates an output Y_t to one or more inputs U_t through the dynamic model G_p .

The model G_p might be used for different purposes, such as designing a controller or inferring the properties of some stream from known measurements. But for our purposes, there are no defined inputs and we need to explicitly account for the noise characteristics of the measurement (after all, if there's no noise, there's no need for a deadband).

Hence time series analysis is well suited for the task at hand. The input to the time series model isn't a known, deterministic input U_t , but rather an unknown random signal input series a_t . The advantage of using a random signal is that you don't have to have a measurement of it – it's determined (along with its properties like the variance or distribution) from the measurement Y_t and the model fitting routine. And the process

model is basically the same as a traditional dynamic model in that it has poles and zeros and model order and all the other stuff.

Of course, the terminology is different, but for our purposes a signal can be well represented by something called an autoregressive-integrated-moving average (ARIMA) model of the form:

$$y_t = \frac{\mathbf{q}(B)}{\nabla^d \mathbf{j}(B)} a_t$$

Here y_t is the value of the process measurement at time t , a_t is a white noise process with variance σ_a^2 . The numerator (i.e. zeros) of the transfer function, termed $\mathbf{q}(B)$, is polynomial of order p and are the moving average part. The autoregressive part is the denominator (i.e., poles) and is written as $\mathbf{j}(B)$ and is of order q . B is the backwards shift operator (i.e., $By_t = y_{t-1}$). The term ∇ (which is the integrating part) equals $1-B$, and if the exponent d on this term is greater than zero, then it means that the signal doesn't have a constant mean. This is important for alarms – if the signal had a constant mean, then it wouldn't go into alarm as the process would drift around some stable value (generally the setpoint).

Perhaps confusing, but a time series model can represent a wide range of signals with just a few parameters. Figure 7.2 shows three variations on the above equation, which are all generated from the same series a_t .

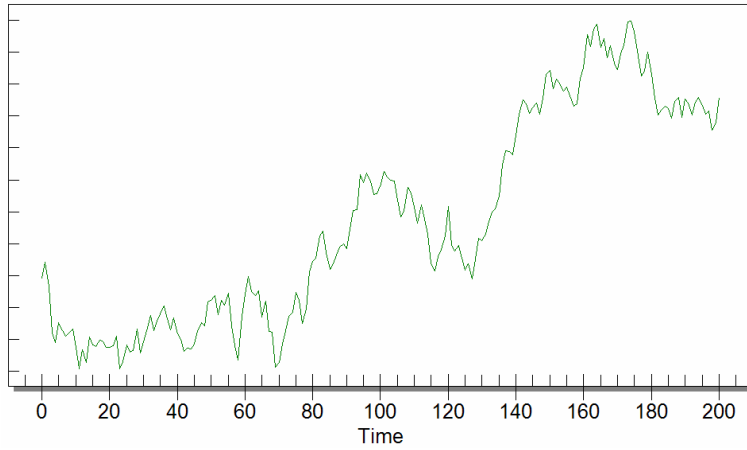


Figure 7.2a: Random Walk simulated disturbance: $y_t = \frac{a_t}{\nabla}$

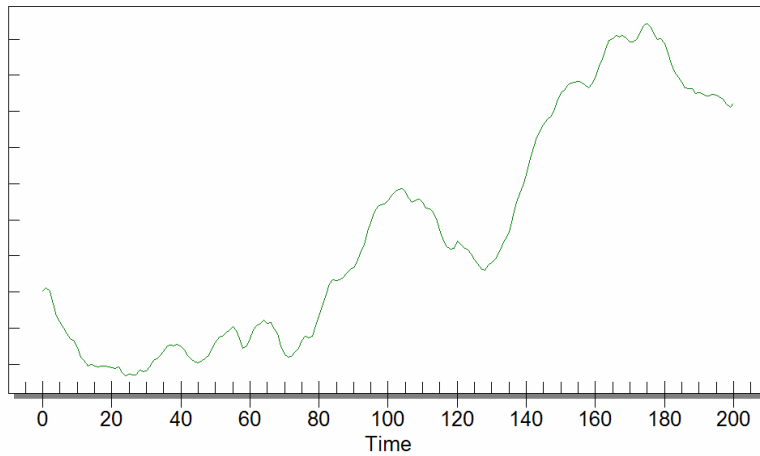


Figure 7.2b: Integrated Auto Regressive trend: $y_t = \frac{a_t}{(1 - 0.8z^{-1})\nabla}$

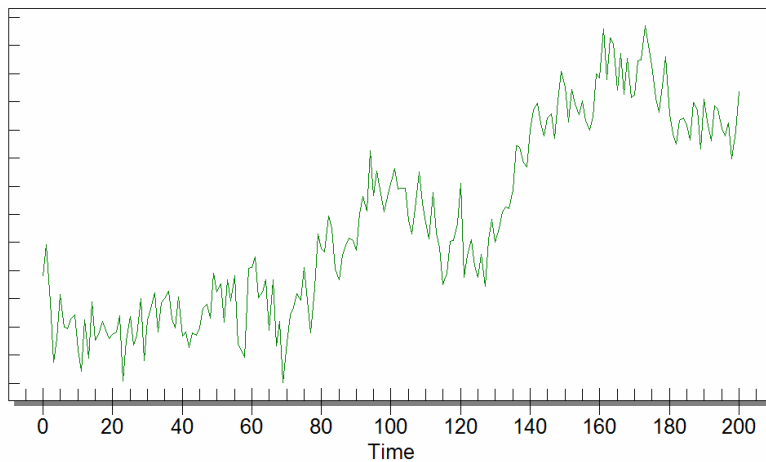


Figure 7.2c: Integrated Moving Average trend: $y_t = \frac{(1 - 0.5z^{-1})a_t}{\nabla}$

As shown in Figure 7.2, for many processes the actual process movement can be represented by the denominator of the ARIMA model, while the numerator models the higher frequency noise (either measurement noise or high frequency process noise). If this is the case, then the series consisting only of the numerator (i.e. high frequency components) given by:

$$z_t = \mathbf{q}(B)a_t$$

will represent the portion of the signal which is causing the chattering. Now for a little bit of time series math. The prediction of the numerator portion l time periods in the future if represented as $\tilde{z}_t(l)$ and is defined, along with $1 - \mathbf{e}$ probability limits by (Box and Jenkins, [4]):

$$z_{t+l} = \tilde{z}_t(l) \pm u_{\mathbf{e}/2} \left\{ 1 + \sum_{j=1}^{l-1} \mathbf{q}_j^2 \right\}^{1/2} s_a$$

where $u_{\mathbf{e}/2}$ is the deviate exceeded by a proportion $\mathbf{e}/2$ of the unit Normal distribution, and s_a is an estimate of the actual process variance \mathbf{s}_a^2 . For an invertible moving average process, the maximum bounds will be given by:

$$\pm u_{\mathbf{e}/2} \left\{ 1 + \sum_{j=1}^p \mathbf{q}_j^2 \right\}^{1/2} s_a$$

If the moving average is not invertible, the maximum bounds may be higher, but they can be easily calculated once the MA parameters are known.

The deadband should be set so that the prediction of the high frequency noise lies within the deadband. For 3-sigma limits, and assuming the moving average term is of order 2, the above reduces to:

$$deadband = 2.95(1 + q_1^2 + q_2^2)^{1/2} s_a$$

The above equation is all you really need to know. It says that the deadband will be a function of the noise variance (as expected – the larger the noise, the larger the deadband) modified by some terms to account for the fact that the signal isn't pure white noise, but is probably "colored".

Fortunately, there's lots of code for building an ARIMA model lying around (MATLAB for instance has a time series module), so it's not necessary to delve into non-linear least squares optimization routines to figure out the model parameters. Just enter the measurement data, and the program will calculate the terms in the above equation.

It's even possible to automate the above, as it's a relatively simple calculation and no plant tests are required. In fact, this may be preferable as the noise characteristics of a process may change when it goes into alarm, so that the best data set for determining the deadband may be when the process the alarms are occurring.

Chapter 7

Controller Performance Evaluation Using Alarms

There is a wide body of theoretical literature, and several commercial software packages, for assessing the performance of industrial controllers. At first glance, controller performance might appear as simple as determining how close the measurement is to setpoint, but this has long been recognized to be a flawed metric as it does not account for the level of disturbances that the controller was fighting during the time the data was gathered. If one were just to consider the setpoint tracking ability, a poor controller on a process with few disturbances might look better than a good controller on a noisy process.

For this reason, controller performance theory, and controller performance software, generally employs algorithms that compare the error variance of the actual system to that which would have been obtained if some optimal controller had been applied to the system. In that way, the effect of the disturbances cancel out, and you're left with an index that is scale independent. Of course, the complicated part is determining what the error variance of the optimal controller would be, without actually designing and applying one, and indeed without any plant tests at all. But it's (theoretically at least) possible to do this with a few statistical insights and a bit of time series analysis, and the commercial software packages wrap all this to hide the messy details from the user.

But comparing your controller to an "optimal" controller may be over-kill for many processes. In a lot of cases, tracking a setpoint isn't critical (which may require excessive

movement on the part of the manipulated variable), but rather it's sufficient to just keep the measurement in some range. And the acceptable range is often the alarm limits.

In this case, a simple and more direct indication of controller performance is to consider the controller output at the time of the alarm (Figure 7.1). The assumption here is that an alarm should occur only when operator intervention is required – i.e., that the control system cannot bring the system back to its nominal point automatically. If the output is not saturated when the alarm occurred, this would indicate that the controller is not acting fast enough, and should be retuned or replaced with a faster controller (such as a deadtime-compensated one).

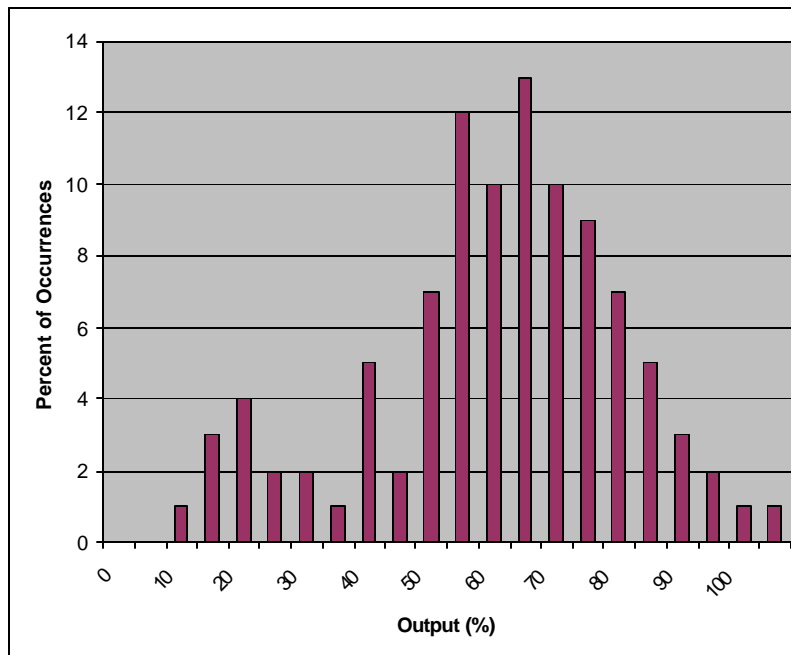


Figure 7.1: Histogram of the controller output at the time of an alarm.

The converse of this logic may also be true, in that non-saturated controllers at the alarm time may indicate that the alarm setting is poor, as the alarm occurred when the process

was still within its operating range. In this case, the alarm trip point should be re-evaluated, as discussed previously.

Chapter 8

Conclusions

Management of alarm systems can be accomplished with knowledge of the process flow sheet, plus some standard management tools. To properly engineer an alarm system however, historized values of the alarm events and process measurements are required, in addition to some moderately advanced statistical and matrix techniques.

Fortunately, most plants already have the necessary informational infrastructure to supply the necessary data. In addition, the mathematical techniques are moderate extensions to a control engineers existing toolset, so their use and interpretation can be easily accomplished. These techniques give greater insight into the performance of an alarm system and give an indication of the proper setting for the parameters associated with an alarm.

Nomenclature

a_t = white noise driving force

B = backwards shift operator

s_a = estimate of \mathbf{s}_a^2

y_t = process measurement at time t

z_t = noise part of measurement at time t

$\tilde{z}_t(l)$ = prediction of noise l periods in the future

∇ = $1-B$

j = autoregressive term

q = moving average term

\mathbf{s}_a^2 = variance of a_t

$u_{e/2}$ = deviate exceeded by a proportion $e/2$ of the unit Normal distribution

References

- [1] EEMUA, “Alarm Systems – A Guide to Design, Management, and Procurement”, 191:99, 1999.
- [2] *Abnormal Situation Management™ Consortium*, www.ASMConsortium.org
- [3] ISA, “Management of Alarm Systems in the Process Industries”, Draft 18.02, Instrument Society of America, 2007
- [4] Box, G.E., and Jenkins, G.M., “Time Series Analysis forecasting and control”, Revised Edition, Holden Day, 1976