Process Monitor Charts

The second of six uses of SPC

Donald J. Wheeler

The simple process behavior chart may be used in several ways. Many articles and some textbooks describe process behavior charts as tools for process monitoring. In Norway the words used for SPC translate as "statistical process steering." In this column we will look at the pros and cons of using a process behavior chart as a process monitor to steer a process.

Process behavior charts allow us to detect process upsets. Clearly, when we have an upset it is important to get things operating normally again, so it is natural to think of using process behavior charts to make adjustments to keep our processes operating at a desirable level. Hence the idea so succinctly summarized as "process steering." Or as one author put it, either consciously or unconsciously, "statistical process-control." The insertion of the hyphen changes the meaning. Instead of a nominative phrase referring to a wholistic approach to analyzing observational data, the hyphen changes SPC into a process-control algorithm that uses statistics. Thus, by virtue of background, training, and nomenclature, many people have come to think of a process behavior chart as simply a manual technique for maintaining the *status-quo*.

In what follows we will see how the process behavior chart performs as a process monitor; we will compare it with two automated process-control algorithms; and we will consider the important question of whether or not this use of the charts will deliver the best process performance. To obtain an example data set containing known upsets we shall begin with data from a predictable process and add some simple step changes. By using data with known upsets we can see how each technique deals with these upsets. This allows us to make direct and equitable comparisons between the techniques.

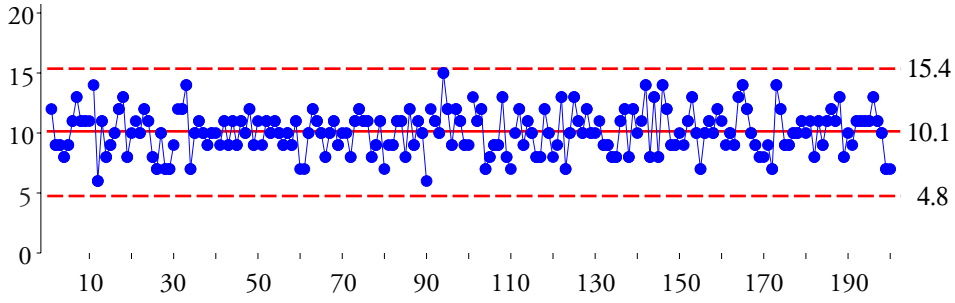

Figure 1: *X* Chart for the 200 data of Line Three

The predictable data set consists of 200 values from Line Three. These data were given in my December 2015 ***QDD*** column, "*The Secret Foundation of Statistical Inference*." The *X* chart for these data is shown in Figure 1. The average is 10.11 units and the average moving range is 1.98 units. Let us assume that the target value for this process is 10.0 units and that the watershed

specifications are 6.5 units to 13.5 units.

Next the data of the predictable process in Figure 1 are modified by adding the step changes of Figure 2. These step changes are 0, 2, 4, 6, 8, or 10 units above or below zero. The labels show what these shifts amounted to in sigma units (based on the estimated process standard deviation of 1.79 units for Figure 1).

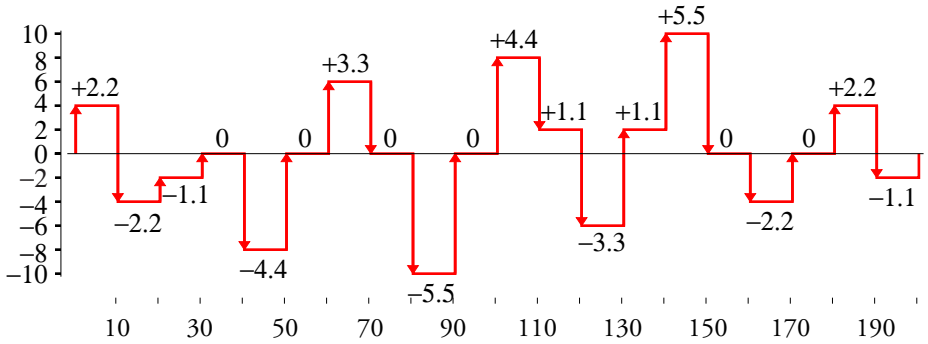

Figure 2: Shifts used to create signals within the data

Six "shifts" are of size zero, so that only fourteen groups were actually shifted from their original values. Four of these shifts amounted to only two units, which is slightly greater than one sigma, so that only ten groups were shifted by an amount that is greater than two sigma. However, a process change occurs on every tenth point, starting with point number 1 and continuing on to point number 191 (see the arrows). Thus, there are twenty process changes (upsets) that are introduced when we add the values of Figure 2 to those of Figure 1.

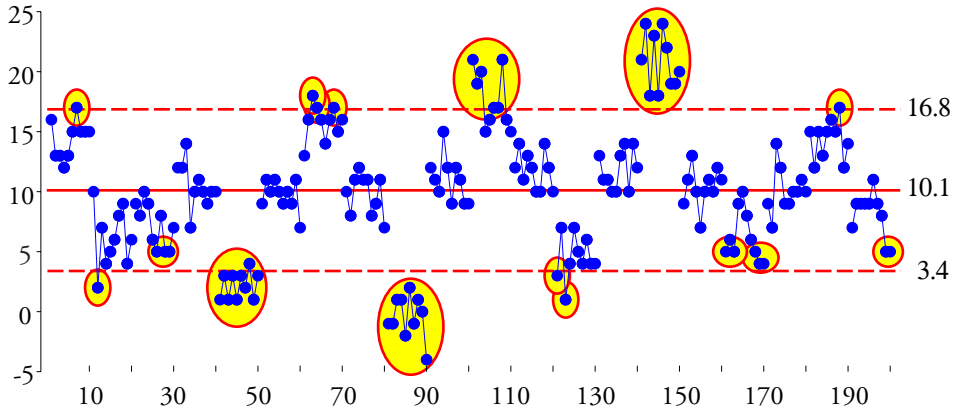

Figure 3: *X* Chart for the Example Data

Our resulting example data set is shown on the *X*-chart in Figure 3 and tabled in Figure 14. The average value remains 10.11 but the average moving range is now 2.51. This results in limits that are 26 percent wider than those in Figure 1. In spite of these wider limits, twelve of the fourteen step changes inserted into Figure 3 are detected by points outside the limits or by runs beyond two-sigma. The two missed shifts are the 1.1 sigma shifts for points 111-120 and 131-140. Thus, all ten of the shifts greater than 2.2 sigma, and two of the 1.1 sigma shifts are successfully

detected by the *X*-chart in Figure 3. This ability to get good limits from bad data is what makes the process behavior chart such a useful technique.

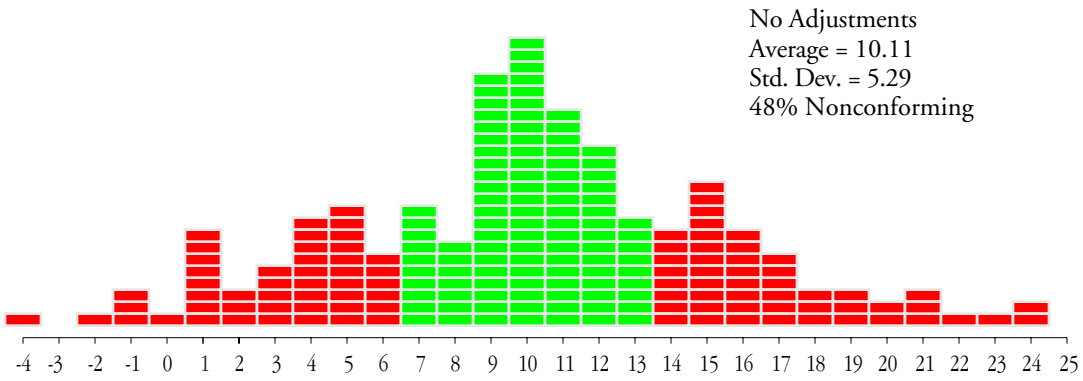

Figure 4: Histogram for the Example Data of Figure 3

With specifications of 6.5 to 13.5 the histogram in Figure 4 has 48% nonconforming. It has a global standard deviation statistic of 5.29 units. Since the data of Figures 3 and 4 will be used in what follows, the descriptive summaries above will be our baseline for comparisons as we consider different approaches to handling process upsets. Specifically we are interested in how much we can reduce both the fraction nonconforming and the standard deviation statistic while keeping the average close to the target value of ten.

The first technique we shall consider will be the use of a process monitor chart.

STATISTICAL PROCESS STEERING

If we use our process behavior chart as a process monitor and make an adjustment to the process aim every time a point goes outside the limits, that adjustment will affect all subsequent values.

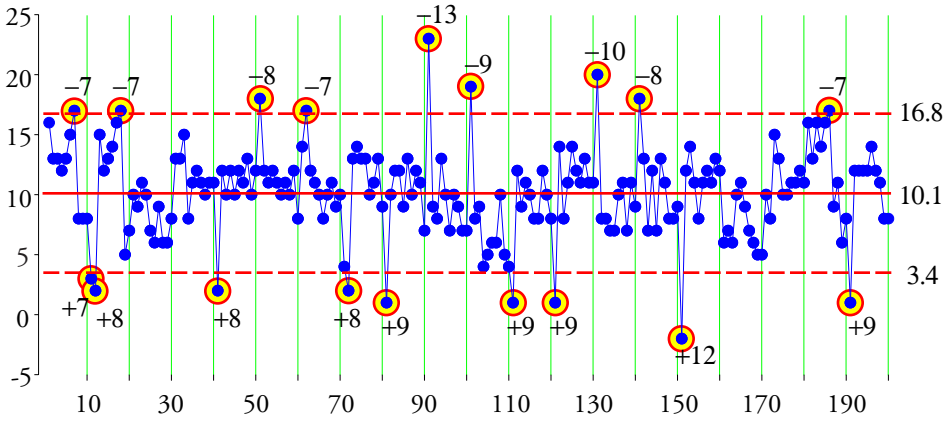

Figure 5: *X* Chart Used as a Process Monitor Chart (18 adjustments)

The cumulative nature of these adjustments means that we will have to process the values in

our example data set in sequence. Using the limits of Figure 3 we will interpret values of 17 or greater and values of 3 or smaller as upsets. Whenever an upset is detected an adjustment equal to the difference between the target value and the current value is made. In this way we end up making eighteen adjustments as shown in Figure 5.

Sixteen of the eighteen signals detected in Figure 5 correspond to process changes introduced in Figure 2. The signals at points 12 and 18 represent a false alarm and the correction for that false alarm. Thus, we compute a false alarm rate of 2/180 or one percent.

The sixteen upsets that were detected were all two-sigma or larger changes. Virtually all of your process changes that are large enough to be of economic consequence will be detected by a process monitor chart.

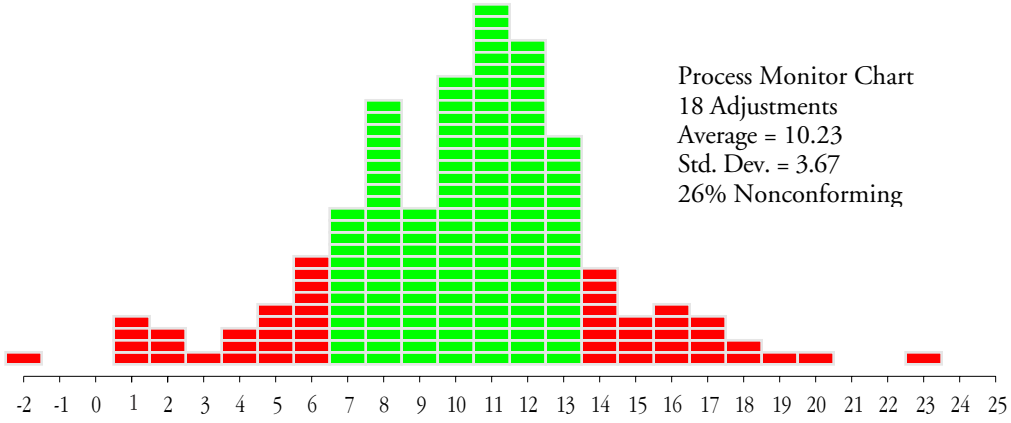

Figure 6: Histogram for Result of Using a Process Monitor Chart

The histogram in Figure 6 has 26% nonconforming and it has a global standard deviation statistic of 3.67 units. These two descriptive measures are markedly better than those of Figure 4. By correctly identifying and adjusting for sixteen of the twenty changes in level, the process monitor chart improved the process outcomes while keeping the process centered near the target value. We have cut the fraction nonconforming in half and reduced the variation in the product stream.

PID CONTROLLERS

Once we view the process behavior chart as a process steering wheel, we open the door to consider other process-control algorithms. These algorithms are intended to keep the process on target by continually adjusting the process aim. There are many types of these devices, and they greatly facilitate all kinds of operations today. The type of process-control algorithm that will be considered here is a simple Proportional-Integral-Differential (PID) controller of the type that has been around for over 100 years.

As the name suggests a PID controller makes adjustments based on the size of three elements. The first element (proportional) will depend upon the difference between the current value and the target value. This difference is known as the error for the current period. Letting the subscript t denote the current time period:

$$\textit{Error for time period } t \;=\; E_t \;=\; X_t \;-\; \textit{Target}$$

The second element (integral) will depend upon the sum of all the prior error terms, which I shall denote here as *SumE* where:

$$\textit{Sum of Error Terms up to time } t \;=\; SumE_t \;=\; E_1 + E_2 + E_3 + \ldots + E_{t-1} + E_t$$

And the third element (differential) will depend upon the difference between the last two error terms, which I shall denote here as *DeltaE* where:

$$\textit{Difference of Error Terms at time } t \;=\; DeltaE_t \;=\; E_t \;-\; E_{t-1}$$

Of course, *DeltaE* for the first time period will be zero. Given these three elements and following the observation at time period t, we can compute the adjustment to make before time period t+1 as:

$$ADJUSTMENT_{t+1} \;=\; -\; a\; E_t \;-\; b\; SumE_t \;-\; c\; DeltaE_t$$

where a, b, and c are the proportional, integral, and differential weights. The reasoning and constraints behind the selection of these weights is beyond the scope of this article. We will use two examples to illustrate the general properties of this class of process-control algorithms.

GIBB'S RULE

Using weights of $a = 0.5$, $b = 0.0$, and $c = 0.0$, the general PID algorithm becomes the simple P-controller sometimes known as Gibb's rule for process adjustment. This process-control algorithm makes an adjustment that is equal to half the difference between the current value and the target. When this rule is sequentially applied to the values tabled in Figure 14 we get the values shown in Figures 7 and 8.

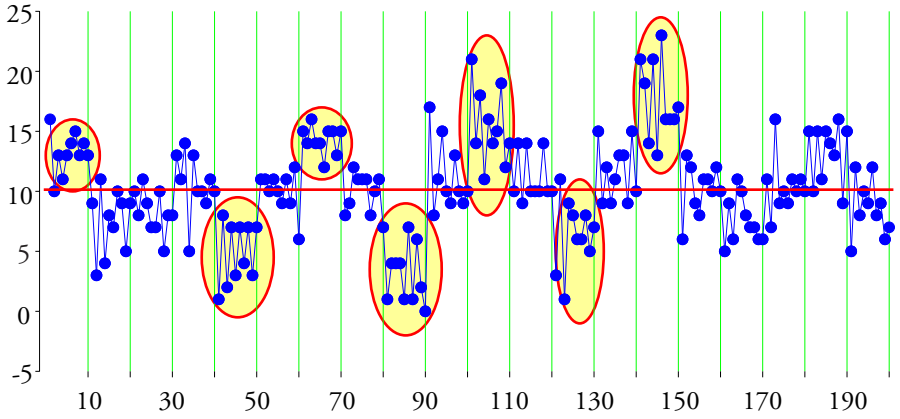

Figure 7: Gibb's Rule (171 adjustments)

Gibb's rule makes 171 adjustments to the process aim during the time covered by these 200 values. Since there were only 20 process changes present in these data, at least 151 of the adjustments made by Gibb's rule have to have been made in response to noise rather than real signals within the process. Hence Gibb's rule has a false alarm rate of 151/180 or 84 percent.

So what is the consequence of making all of these unnecessary adjustments? Figure 7 shows

an oscillating process with seven runs about the target consisting of eight or more successive values on the same side of the target. This oscillation problem is common with many PID-controllers. While this algorithm managed to keep the overall average near the target it did this by weaving back and forth while allowing the average to be far from the target for extended periods.

Time after time, when my clients put the data from their automatic process-control algorithms on a simple *XmR* chart, they are surprised by two things: the oscillations made by their controller, and the substantial increase in the process variation that results from those oscillations. In many cases their reaction is to turn off the process-controller and return to manual operation.

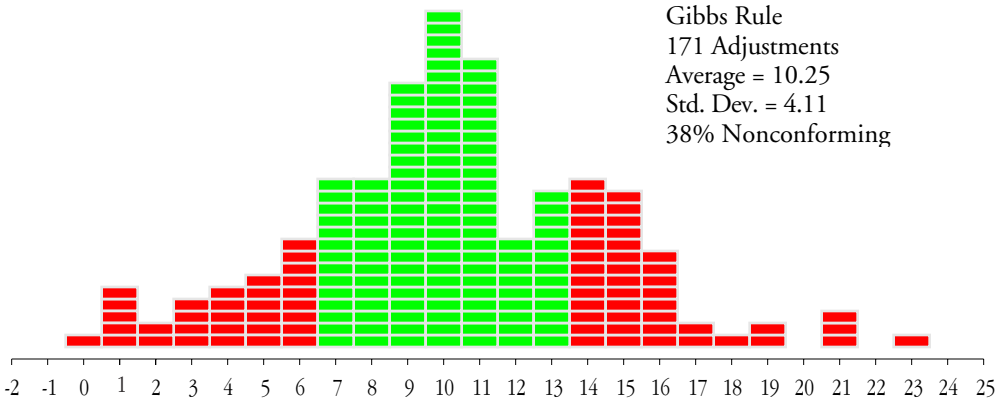

Figure 8: Histogram for the Outcome of Using Gibb's Rule

The histogram in Figure 8 has 38% nonconforming and a global standard deviation statistic of 4.11 units. While these descriptive measures are somewhat better than those of the example data set in Figure 4, they are not as good as those produced by the process monitor chart in Figure 6. Instead of merely reacting to those shifts that are large enough to be of economic consequence, Gibb's rule reacted to almost everything, and consequently ended up making 171 adjustments.

This process simply will not fit inside the specifications for the product. The process is a wide load, and the specifications are a narrow bridge. And when you have a wide load and a narrow bridge, it does no good to put a drunk at the wheel.

A PID CONTROLLER

In looking for a PID controller that would do better than Gibb's rule I considered a total of 129 sets of PID weights. Only 57 of these PID controllers did not explode. Of these 57, seventeen actually made things worse—they ended up with more than the 48 percent nonconforming found in the example data set. Fifteen PID controllers resulted in 38 to 48 percent nonconforming. So 32 of the 57 did worse than Gibb's rule. Twenty-three PID controllers had between 28 and 38 percent nonconforming, and only two PID controllers were found that were comparable with the process monitor chart's 26 percent nonconforming. The percentages nonconforming for these 57 different convergent PID controllers are shown in Figure 9.

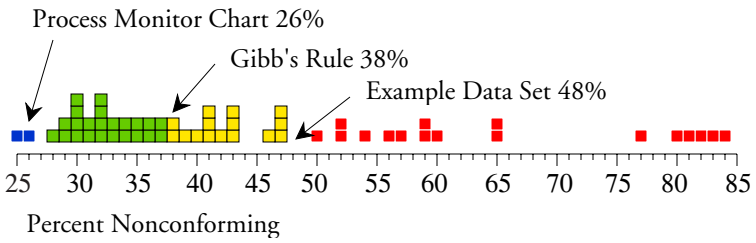

Figure 9: The Fraction Nonconforming Produced by Each of 57 PID Controllers

The PID controller used for comparison here is the best algorithm from Figure 9. It has weights of $a = 0.4$, $b = 0.4$, and $c = 0.0$. The results are shown in Figures 10 and 11.

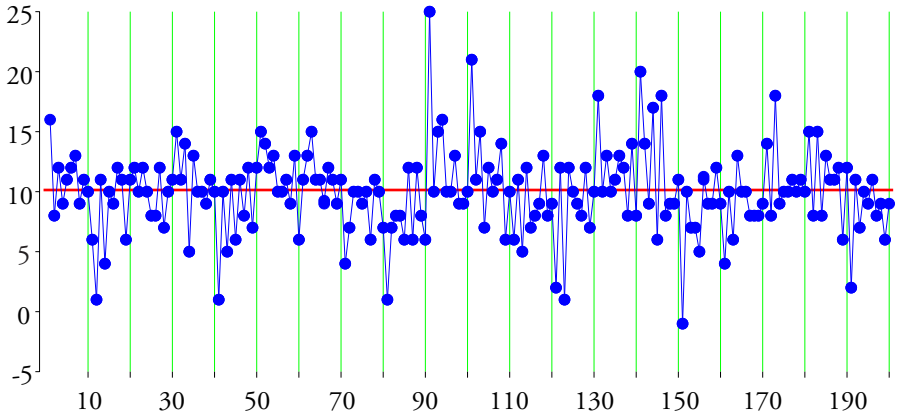

Figure 10: The Best PID-Controller (189 adjustments)

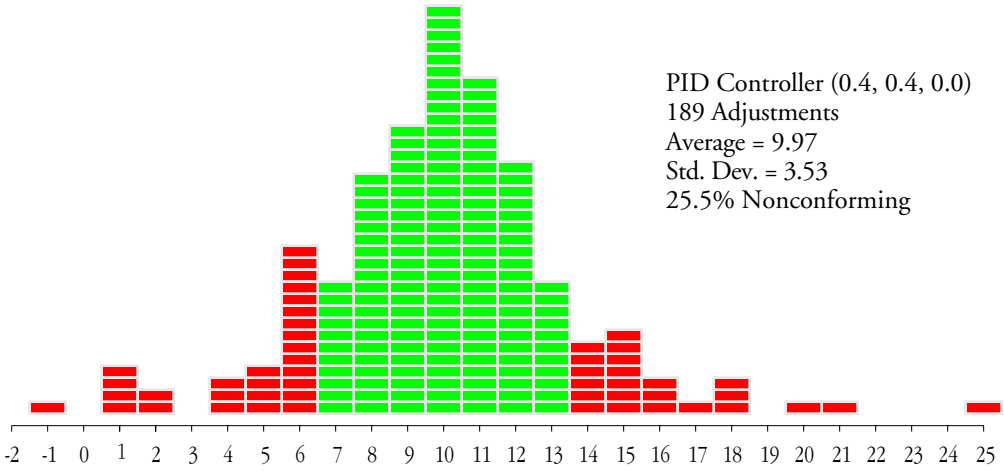

Figure 11: Histogram for the Outcome of Using the Best PID-Controller

This PI-controller made 189 adjustments to the process aim even though there were only 20 actual changes in level present in these data. Thus, this controller has a false alarm rate of 169/180 or 94 percent. So even though this controller is constantly adjusting the aim, it

dampened the oscillations inherent in P-controllers by including an integral term. The result is a running record in Figure 10 that looks much more like that of Figure 5.

The histogram in Figure 11 has 25.5 percent nonconforming and a global standard deviation statistic of 3.53. These descriptive statistics are on a par with those for the process monitor chart in Figure 6. Figure 11 effectively represents the best that can be done with these data using a PID controller. Other types of process-control algorithms might do slightly better, but as may be seen in Figure 9, automated process-controllers have trouble equaling the performance of a process monitor chart.

PROCESS MONITORING

What we have illustrated here is that as a *manual* technique for maintaining the *status quo,* a process monitor chart will do essentially as well as the best possible PID-controller, and it will do this using fewer process adjustments. The process monitor chart will also be much easier to use than a PID controller since it will not need to be "tuned" to match the characteristics of the process being adjusted. With this combination of simplicity and efficiency, the process behavior chart represents the best that you can do when considering a manual technique for process steering.

If you want an *automatic* technique for process steering there are many different algorithms that are more easily programmed than the process monitor chart. However these automated algorithms for process steering will tend to make many more adjustments than the process monitor chart. One reason why this happens is that in control theory the assumption is often made that it costs nothing to make an adjustment. However, in practice, as opposed to theory, every unnecessary adjustment increases the variation in the product stream, and increased variation always results in increased costs downstream. For this reason most process-control algorithms will not do as well as a process monitor chart. A few, when properly tuned and adjusted, will do about the same as a process monitor chart. But if you pick the wrong process-control algorithm you may actually make things worse than doing nothing.

Adjustments make sense when they are based on signals of actual upsets. Adjustments cannot help when they are based on background noise. The key to any effective process steering mechanism is to filter out the noise and react only to the signals. With most process-control algorithms it is up to you to devise a technique that will separate the potential signals from the probable noise and will then react appropriately.

When you get lost in the complexity of setting up a process-control algorithm, remember that Shewhart has already given us a proven and effective way of separating signals of economic consequence from the background noise. With a process behavior chart you use a generic filter that has been proven the world over to work in all kinds of situations and with all kinds of data. Simply add data and stir.

OPERATING AT FULL POTENTIAL

None of the process monitoring techniques given above allow you to operate this process up to its full potential. By adjusting the process aim you will always be reacting to the perceived upsets rather than preventing them. The process monitor approach completely ignores the use of a process behavior chart as a process *improvement* mechanism.

Instead of merely reacting to the process upsets by adjusting the process aim, it is much better to seek to learn about the assignable causes which are creating the upsets. As you identify these assignable causes you can begin to remove their effects from your process by making the assignable causes part of the set of controlled inputs for your process. In this way you can take chunks of variation out of the product stream and operate your process up to its full potential. When we do this for the data of Figure 3 we end up with the *X* chart of Figure 1 and the histogram of Figure 12.

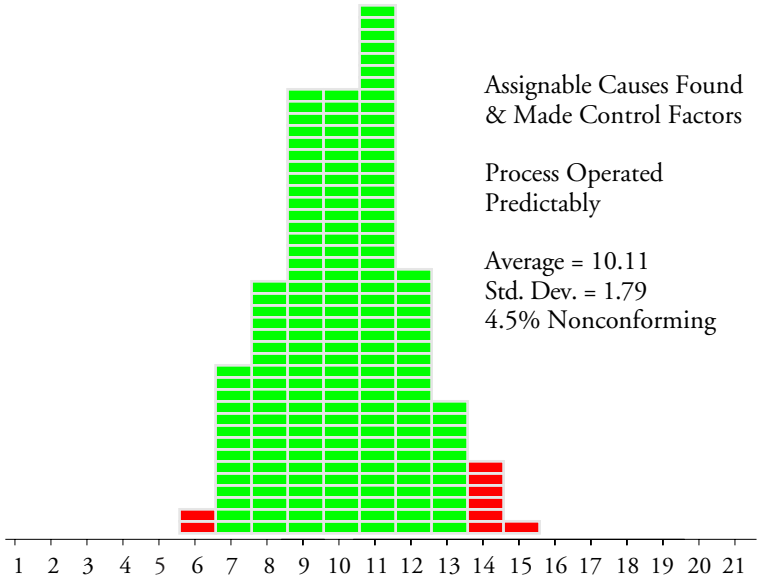

Figure 12: Histogram for This Process Operated Predictably

Figure 12 shows the example data set after the assignable causes have been found and made part of the set of control factors for this process. Now the process is operating up to its full potential. The global standard deviation statistic is 1.79, which is half of what the best process-control algorithms could accomplish. Figure 12 has 4.5 percent nonconforming, which is one-fifth of what the best process-control algorithms could accomplish, and one-tenth of what the example data set displayed in Figure 4. So while this process is not capable of meeting these specifications, operating it predictably will greatly improve its performance compared with the process-steering approaches.

Figures	Process Description	No. of Adjust.	Aver.	Std. Dev.	% Out of Spec.	False Alarms
3 & 4	Example Data Chart	0	10.11	5.29	48.0%	0%
7 & 8	Gibb's Rule	171	10.25	4.11	38.0%	84%
5 & 6	Process Monitor Chart	18	10.23	3.67	26.0%	1%
10 & 11	Best PID-Controller	189	9.97	3.53	25.5%	94%
1 & 12	Operated Predictably	0	10.11	1.79	4.5%	0%

Figure 13: Summary of Results of Different Approaches

Figure 13 compares the process-control algorithms considered. They are ordered by the size

of their standard deviation statistics and their fractions nonconforming. The last column shows the false alarm rates for each. The process monitor chart does the best job with the fewest false alarms. Always has, always will.

So what do you want for your process? Do you want to simply maintain the *status quo*, or do you want to operate your process up to its full potential? The process monitor view of SPC assumes that the *status quo* is good enough. But process behavior charts were created to help you get the most out of your process.

Time after time my clients tell me that as they learn how to operate their processes predictably they find that the process variation has also been reduced to one-half, one-third, one-fourth, or one-fifth of what it was before. The only way to operate your process up to its full potential is to operate your process predictably. So, while it may be appropriate to react to a process upset by making a process adjustment, no process-steering technique will allow you to operate your process up to its full potential unless you are already operating it predictably.

A process behavior chart may well be one of the best process monitoring techniques ever invented, but it was created to do so much more. Therefore, trust no one who tells you that SPC is merely a process monitoring technique.

APPENDIX

The data for our example data set shown in Figure 3 are listed in Figure 14 in time order by columns.

1-20	21-40	41-60	61-80	81-100	101-120	121-140	141-160	161-180	181-200
16	9	1	13	−1	21	3	21	5	15
13	8	3	16	−1	19	7	24	6	12
13	10	1	18	1	20	1	18	5	15
12	9	3	17	1	15	4	23	9	13
13	6	1	16	−2	16	7	18	10	15
15	5	3	14	2	17	5	24	8	16
17	8	2	16	−1	17	4	22	6	15
15	5	4	17	1	21	6	19	5	17
15	5	1	15	0	16	4	19	4	12
15	7	3	16	−4	15	4	20	4	14
10	12	9	10	12	12	13	9	9	7
2	12	11	8	11	14	11	11	7	9
7	14	10	11	10	11	11	13	14	9
4	7	11	12	15	13	10	10	12	9
5	10	10	11	12	12	10	7	9	9
6	11	9	11	9	10	13	10	9	11
8	10	10	8	12	10	14	11	10	9
9	9	9	9	11	14	10	10	10	8
4	10	11	11	9	12	14	12	11	5
6	10	7	7	9	10	12	11	10	5

Figure 14: Example Data Set Containing 20 Upsets

129 different sets of PID weights were considered where $a \geq b > c$, but only 57 sets resulted in a PID controller that did not blow up when applied to the example data set. Figure 15 lists the PID weights for these 57 different algorithms along with the resulting percentages nonconforming for the example data set.

a	b	c	%	a	b	c	%	a	b	c	%
.1	.0	.0	47.0	.4	.3	.2	56.5	.6	.3	.1	59.5
.1	.1	.0	46.0	.4	.4	.0	25.5	.6	.4	.0	40.0
.2	.0	.0	47.0	.4	.4	.1	36.0	.6	.4	.1	83.0
.2	.1	.0	41.5	.4	.4	.2	77.0	.6	.5	.0	50.5
.2	.2	.0	30.5	.5	.0	.0	38.0	.6	.6	.0	59.5
.2	.2	.1	32.0	.5	.1	.0	34.0	.7	.0	.0	37.5
.3	.0	.0	43.5	.5	.2	.0	30.5	.7	.1	.0	38.5
.3	.1	.0	37.5	.5	.2	.1	38.5	.7	.2	.0	39.5
.3	.2	.0	31.5	.5	.3	.0	30.5	.7	.2	.1	81.0
.3	.2	.1	32.5	.5	.3	.1	43.5	.7	.3	.0	47.5
.3	.3	.0	26.0	.5	.4	.0	30.0	.7	.4	.0	57.0
.3	.3	.1	29.5	.5	.4	.1	54.0	.7	.5	.0	65.0
.3	.3	.2	42.5	.5	.5	.0	33.5	.7	.6	.0	84.0
.4	.0	.0	41.5	.5	.5	.1	60.5	.8	.0	.0	41.5
.4	.1	.0	35.5	.6	.0	.0	36.5	.8	.1	.0	43.5
.4	.2	.0	29.0	.6	.1	.0	34.5	.8	.2	.0	52.0
.4	.2	.1	35.5	.6	.2	.0	32.5	.8	.3	.0	65.0
.4	.3	.0	28.5	.6	.2	.1	52.0	.8	.4	.0	82.5
.4	.3	.1	31.0	.6	.3	.0	33.5	1.0	.0	.0	80.0

Figure 15: 57 PID Controllers That Converged