Test of fire detection algorithms using artificially generated events

Abstract
The paper proposes a test method which is particularly suitable for the test of the false alarm rejection features of detection algorithms. It describes how to generate signal events artificially, which are similar to those occurring in practice and thus may or may not cause false or unwanted alarms. The generated events are of random nature, i.e. the signal shape, amplitude and duration of the events is randomly varied and the points in time, where these events occur, are randomly selected according to a well defined signal model. A set of parameters controls the event density, the probability for the duration of events, the fluctuation and the full dynamic range. The whole signal model uses only one pseudo random generator and generates a modified “random walk process”. Thus, it is either possible to reproduce with the same set of parameters for the signal model the same random sequence of events or, alternatively, to generate different random signal sequences using a different seed for the pseudo random generator. The test procedure is carried out in form of a computer simulation in time lapse mode and thus requires the detection algorithm to be available as a C-program.

1. Introduction
Modern fire detectors show good detection capabilities in the case of test fires as well as in genuine fires, but the rate of false or unwanted alarms is quite high (about 10..20 false or unwanted alarms per true alarm). The more automatic fire detectors are installed, the more serious becomes the false alarm problem.

There are various possible means available for reducing the false alarm rate:

- A professional design of the hardware for fire sensors and detectors.
- The use of multi-sensor fire detectors.
• The implementation of advanced detection algorithms with pre-selectable operating modes for different installation environments.

• The automatic adaptation of detection algorithms to varying environmental conditions.

However, the test of the false alarm resistance of fire detectors is in any case a difficult problem and, probably for this reason, test laboratories do not test the false alarm behaviour at all. It may be possible to define and standardize a limited number of false alarm tests, but the application of such tests is time-consuming and expensive.

A cheap alternative exists at least for the test of detection algorithms. Modern fire detectors use more sophisticated methods than a simple comparison of the fire-sensor signal with an alarm threshold. Such detection algorithms are usually implemented on microprocessors; hence there exist detection programs usually written in C, C++ or assembler. If we could generate artificial but somehow typical fire sensor signals for false alarm-relevant events and define for different environments the density of such events in time, this might help to optimise the overall performance of detection algorithms and/or to compare the quality of different detection algorithms using time lapse computer simulation methods.

The shortcoming of this method is that we do not test the features of the sensors and their hardware with respect to false alarm-relevant environmental events. But we definitely know, that the corresponding sensor signals of such events are of a random nature in genuine fire situations as well as in No-fire environments. Even if the statistical features of generated artificial events do not perfectly match what occurs in practice, the application of such a method is better than no false alarm test at all, and for adaptive detection algorithms it is the only reasonable method.

2. Design targets for the test generator.

Almost everywhere in our industrialized world controlled combustion processes occur, for example, in the combustion motors of cars and machines, in the heating of buildings, combustion processes in production plants, lighted cigarettes, cooking etc. Consequently, combustion products like smoke, gas, heat are found almost everywhere. Moreover,
sensors used for fire detection purposes show cross-sensitivities for other environmental components, e.g. dust, vapours etc. for smoke sensors, some gas sensors respond to various chemical components. Moreover, the ambient temperature varies significantly during the day and over the year in normal environments.

From field measurements in different No-fire environments it is known, that fire sensors generate signals of a more or less long duration and of more or less high amplitudes, which may or may not cause false or unwanted alarms. Such events occur at random in time with different event densities, depending on the installation environment.

Hence, a generator for such events should generate events at random points in time with random shapes, amplitudes and durations. Moreover it should enable the adjustment of

1. the dynamic range and quantization of the output signals,
2. the event density, i.e. the average number of events per time period with the aim of simulating more or less “dirty” environments,
3. the probability for the occurrence of high amplitudes, i.e. the probability for the event amplitude to exceed a given percentage of the adjusted dynamic range,
4. and/or the probability for the occurrence of long event durations, which usually cannot be adjusted independently of the event amplitude.
5. the sample rate,
6. and the test period in years.

3. Realization of the test generator.

The heart of the event generator is a pseudo random number generator, which produces statistically independent uniformly distributed random numbers and is initialized with a so called “seed value”. With the same seed value it always produces the same sequence of random numbers. Hence, all signals derived from this source can be reproduced. This is advantageous, because it is possible to test two different detection algorithms with identical event sequences. Alternatively, it is possible to test the same detection algorithm with different event sequences if a different initial seed value is used.
First it is necessary to generate those random points in time where the signal events occur with a predetermined event density \( \lambda \).

\[
\lambda = \frac{\text{number of events}}{\text{time period}} = \frac{n}{T} \tag{1}
\]

Since the time is quantized by the predetermined sample rate \( \Delta t \), there are \( \frac{T}{\Delta t} \) samples in the whole time interval \( T \). Thus, it is necessary to generate \( n \) numbers - independent of each other - using a random generator which produces uniformly distributed numbers in the range \( \frac{T}{\Delta t} \). These numbers represent the time-instances, in which events occur. The gaps between all of these time-instances are filled with nominal No-fire values.

The test time period \( T \) may be some years. Thus, \( \frac{T}{\Delta t} \) is usually a very large number. For a sample rate \( \Delta t = 1/\text{s} \) and a test period \( T = 1 \) year, \( \frac{T}{\Delta t} \approx 31 \cdot 10^6 \).

The method for generating events, which show typical characteristics of smoke sensor responses is a so-called “modified random walk process”. The underlying theory is discussed in the appendix together with the theoretical background for the calculation of the probability for the duration and height of such events. This method provides two independently adjustable parameters \( p \) and \( \Delta x \). \( p \) controls the probability for the duration of events, which is not independent of the maximum amplitude of such events, because long events usually show higher amplitudes than short events. The positive real parameter \( \Delta x \) is used to adjust the fluctuation of generated events, but does not change the probability for the duration of events. An additional vertical scale factor \( v \) provides the adjustment for the full dynamic range of the signals. A few generated typical events of comparatively long duration, which resemble the output signals of smoke sensors in the earliest phase of a fire, are shown in Fig. 1.

The probability of the length of events is calculated in the Appendix. The results are shown in Fig. 2. Obviously, most of the generated events are of a quite short duration \( n \). The lower \( p \), the lower the probability to obtain long events. Example: with the parameter \( p = 0.45 \) the probability of generating an event of length \( n = 150 \) is \( \approx 10^{-4} \), in other words, on the average, one of \( \approx 10^4 \) generated events shows a length of 150 samples, whereas with \( p = 0.35 \) one of \( \approx 10^6 \) generated events shows a length of 150 samples.

Moreover, it is necessary to find the probability for the amplitudes of long events to exceed a given threshold. The event generator is designed in such a way, that all samples
of an event must be positive and that the highest possible amplitude is $y(n)_{\text{max}} = 2n \cdot \Delta x$, although $y(n)_{\text{max}}$ is extremely unlikely to occur. Thus, the most frequently occurring events of short duration show comparatively low amplitudes. Since the parameter $\Delta x$ does not change the probability for the length of events, it is normalized to $\Delta x = 1$ for the calculations. Two contour diagrams in Fig. 3 show these probabilities for long events as functions of $n$ and the parameter $p$. The negative numbers on the contour lines indicate the $\log_{10}$ of these probabilities. Clearly, an event which exceeds a threshold value $y_s > 0$ at the $n$-th sample has a total length, which must be much longer than $n$. This happens quite seldom, according to Fig. 2 for large $n$. Moreover, the probability for the amplitude to exceed $y_s = 10$ or $y_s = 20$ is low (for example $10^{-4}$ for $n = 100$, $p = 0.35$ and $y_s = 10$). Hence, for $\Delta x = 1$ and $p < 0.45$ it is very unlikely to obtain events of a higher amplitude than 20. It is not impossible, however, because the event generator represents a pure stochastic model.

With the same generator it is also possible to generate events, which resemble a typical
Fig. 2: Comparison of simulation results (*) and theoretical calculations —— for the probability of the length of events. The vertical scale is logarithmic.

fire situation. With parameter $p > 0.5$ it is more likely that the generated event exceeds a certain saturation value rather than to return to zero beforehand. The parameter $\Delta x$, in this case, allows to adjust the amplitude fluctuation. A few examples of such events are shown in Fig. 4.

Events as shown in Fig. 1 and Fig. 4 are similar to smoke sensor events in No-fire and fire situations, respectively. Heat or gas sensor signals are usually smoother. The same event generator can be used, however, to generate heat and/or gas sensor events. This requires passing generated signals through adequately chosen low-pass filters. The corresponding theoretical considerations for this technique would exceed the scope and the number of permitted pages for this publication. It is possible however, to generate simultaneously smoke, heat, and gas events either mutually independent or jointly dependent.
Fig. 3: Contour lines as a function of $p$ and $n$ for the log10-Probability of events to exceed the value $y_s = 10 \cdot \Delta x$ (upper diagr.) or the value $y_s = 20 \cdot \Delta x$ (lower diagr.) at the $n$-th sample and all previous values of the event are $> 0$. Here: $\Delta x = 1$.

Example: the probability for an event to exceed the value $10 \cdot \Delta x = 10$ after 149 previous positive samples equals $10^{-5}$ if $p = 0.35$. 
Summary

One method has been discussed how to generate signals artificially, which resemble smoke sensor signals in fire and No-fire situations. This method is useful particularly for the test of the false/unwanted alarm behaviour of detection algorithms using computer simulations in time-lapse mode if the detection algorithm is available as a program code. The underlying well-defined random model can be extended to generate the smoother heat or gas-sensor signals as well. Thus, it may be used to test multi-sensor/multi-criteria detection algorithms. Although this pure stochastic model generates signal sequences of random nature for the length, height and shape of events, the complete chain of events may be reproduced, which is advantageous for comparison tests of the behaviour of different detection algorithms.

Fig. 4: Three examples of artificially generated smoke sensor fire signals
Appendix - Theoretical considerations concerning the event generator

A brief representation of the theoretical background for the model of the event generator is given in the following for those readers who might be interested in it.

The aim is to generate discrete-time functions (events) $y(k)$ with $k \in \{0, (1), n\}$ of random shape, length and amplitude. Only one pseudo-random generator is used for this purpose which generates uniformly distributed random numbers $x$ in the range $0 \ldots 2\Delta x$.

The generation procedure is as follows:

1. The first generated number forms the first sample of the event $y(1) = x(1)$, where $x^{(j)}$ denotes the $j$-th realization of the random generator.

2. The next generated number $x_h^{(2)}$ is compared with a variable threshold $p_{\text{thr}} = p \cdot 2\Delta x$ with $p$ in the range $0 \ldots 1$. If $x_h^{(2)} \leq p_{\text{thr}}$ the subsequent generated number $x^{(2)}$ is added (otherwise subtracted) from $y(1)$ and forms the second sample of the event $y(2) = y(1) + x^{(2)}$.

The second step above is repeated with always new random numbers $x_h^{(k)}$ and $x^{(k)}$ as long as $y(k) > 0 \quad \forall k \in 1 \ldots n - 1$. Hence, the event terminates at $y(n)$ if $y(n) < 0$. The last sample $y(n)$ is subsequently set to zero.

This process generates the required events. A few realizations of events with comparatively long duration are shown in Fig. 1.

Such events are sections of “modified random walk” sequences and are so-called “first order Markov sequences” if the pseudo-random generator produces independent numbers $x$. Though the Markov features are easy to prove, the proof is omitted here.

The event generation process can alternatively be described as follows:

$$y(1) = |x_1|, \quad y(2) = y(1) + x_2, \quad y(3) = y(2) + x_3, \ldots, \quad y(n) = |x_1| + \sum_{k=2}^{n} x_k \quad (2)$$

with probability densities (pdf’s) for the random variables $f_{|x_1|}(x) = f_{x_1}(x)$ and $f_{x}(x) = f_{x_k}(x)$ as shown in Fig. 5, i.e. all random variables $x_k \forall k \geq 2$ have the same pdf.

Clearly, $y(n)$ as shown in Eq. 2 can take positive or negative values for each $n \geq 2$ but, as an additional condition, each event is terminated on the first occurrence of a negative value.
The calculation of this probability is comparatively simple, but not sufficient, because it includes all events which show previous zero-crossings. Since the event generation process terminates automatically each event with the first zero-crossing, it is necessary to calculate instead the following probability

$$P([z_n < 0] \cap [y(n-1) > 0] \cap [y(n-2) > 0] \cap \ldots \cap [y(2) > 0] \cap [y(1) > 0]) = \int_{-\infty}^{0} \int_{0}^{\infty} \frac{Q}{q!} \int_{0}^{\infty} f_{z_n(n), y(n-1), \ldots, y(2), y(1)}(z_n, y(n-1), \ldots, y_2, y_1) \, dz_n \, dy_{n-1} \ldots dy_1 \tag{4}$$

which seems to be hopelessly complicated because this is an $n$-fold integral, to be calculated for all lengths $n$ in the range $2 < n \leq 300$ or even higher values. Nevertheless, it is possible to calculate this $n$-fold integral recursively.

At first it is necessary to calculate $f_{z_3(n), z_2(n), \ldots, z_1(n), y(n-1), \ldots, y(2), y(1)}(z_3, \ldots, z_1, y, y_{n-1}, \ldots, y_2, y_1)$ which is shown for $n=3$ only; the extension to higher $n$ is straightforward. Consider the following equation system of random variables: $z_3 = y_3 / y_2$, $z_2 = y_2$, $z_1 = y_1$, where the abbreviation $y(i) = y_i \forall i$ is used.

$$f_{z_3, z_2, z_1}(z_3, z_2, z_1) = \sum f_{y_3, y_2, y_1}(y_3(z_3), y_2(z_2), y_1(z_1)) \frac{|J(z_3, y_3)|}{|J(z_3, y_1)|} \tag{5}$$

$|J(z, y)|$ is the Jacobian determinant, which is calculated for the given simple equation.
system as follows:

\[ |J(z_j, y_i)| = \begin{vmatrix}
\frac{\partial z_3(y_3, y_2)}{\partial y_3} & \frac{\partial z_3(y_3, y_2)}{\partial y_2} & \frac{\partial z_3(y_3, y_2)}{\partial y_1}
\frac{\partial z_2(y_2)}{\partial y_3} & \frac{\partial z_2(y_2)}{\partial y_2} & \frac{\partial z_2(y_2)}{\partial y_1}
\frac{\partial z_1(y_1)}{\partial y_3} & \frac{\partial z_1(y_1)}{\partial y_2} & \frac{\partial z_1(y_1)}{\partial y_1}
\end{vmatrix} = \begin{vmatrix}
\frac{1}{y_2} - \frac{y_3}{(y_2)^2} & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{vmatrix} = \frac{1}{|y_2|}
\]

(6)

With the given equation system and Eq.’s 5 and 3 follows:

\[ f_{z_3, z_2, z_1}(z_3, z_2, z_1) = |z_2| \cdot f_{y_3, y_2, y_1}(z_3 \cdot z_2, z_2, z_1) \]

\[ = |z_2| \cdot f_x(z_3 \cdot z_2 - z_2) \cdot f_x(z_2 - z_1) \cdot f_x(z_1) \]

\[ = |z_2| \cdot f_x(z_2 \cdot (z_3 - 1)) \cdot f_x(z_2 - z_1) \cdot f_x(z_1) \]

and the extension to higher \( n \) results in:

\[ f_{z_n \ldots z_2, z_1}(z_n, \ldots, z_2, z_1) = |z_{n-1}| \cdot f_x(z_{n-1} \cdot (z_{n-1} - 1)) \cdot f_x(z_{n-1} - z_{n-2}) \cdot \ldots \]

\[ \ldots \cdot f_x(z_2 - z_1) \cdot f_x(z_1) \]

(7)

Hence, Eq. 4 can be rewritten using Eq. 7:

\[ P([z_n < 0] \cap [z_{n-1} > 0] \cap [z_{n-2} > 0] \cap \ldots \cap [z_2 > 0] \cap [z_1 > 0]) = \]

\[ = \int_{-\infty}^{0} \int_{0}^{\infty} \int_{0}^{\infty} |z_{n-1}| \cdot f_x(z_{n-1} \cdot (z_{n-1} - 1)) \cdot f_x(z_{n-1} - z_{n-2}) \cdot \ldots \\
\ldots \cdot f_x(z_2 - z_1) \cdot f_x(z_1) \; dz_n \; dz_{n-1} \ldots dz_1 \\
\]

\[ = \int_{0}^{\infty} \int_{-\infty}^{\infty} f_x(z_{n-1} \cdot (z_{n-1} - 1)) \; dz_n \cdot |z_{n-1}| \cdot f_x(z_{n-1} - z_{n-2}) \cdot \ldots \\
\ldots \cdot f_x(z_3 - z_2) \int_{0}^{\infty} f_x(z_2 - z_1) \cdot f_x(z_1) \; dz_1 \; dz_2 \ldots \; dz_{n-1} \\
\]

\[ = \int_{0}^{\infty} g(z_{n-1}) \cdot |z_{n-1}| \cdot f_x(z_{n-1} - z_{n-2}) \cdot \ldots \\
\ldots \cdot f_x(z_3 - z_2) \underbrace{\int_{0}^{\infty} f_x(z_2 - z_1) \cdot f_x(z_1) \; dz_1}_{g_1(z_2)} \; dz_2 \ldots \; dz_{n-1} \\
\]

(8)

This arrangement of the equation looks slightly more friendly and shows how this \( n \)-fold integration can be carried out recursively. Unfortunately, the lower limits of the integrals which determine \( g_i(z_{i+1}) \forall i \in \{1, (1, n - 2) \} \) are not \(-\infty\). If they were \(-\infty\), these integrals
would denote the $n - 3$-fold convolution $f_x \ast \ldots \ast f_x$ of the same densities convoluted with $f_{k|x}$ and at least for sufficiently large $n$ all integrals of the second line of Eq. 8 could easily be approximated by a Gaussian density. Nevertheless, since the functions $f_x$ and $f_{k|x}$ are known (see Fig. 5) it is even possible to calculate a closed form solution of Eq. 8 although this would be a laborious and time consuming task for large $n$.

Fortunately, for the given $f_x$ and $f_{k|x}$ all functions $g_i(z_{i+1})$ are zero outside the range $-2\Delta x \leq z_{i+1} \leq (i + 1)2\Delta x$ thus, a numerical solution is possible. Moreover, a modified successive convolution procedure is applicable, as shown in the following, using the unit step function $s(z)$:

1. step: \[ g_1(z_2) = f_x(z_2) \ast f_{k|x}(z_2) \]
2. step: \[ g_2(z_3) = f_x(z_3) \ast [s(z_3) \cdot g_1(z_3)] \]
3. step: \[ g_3(z_4) = f_x(z_4) \ast [s(z_4) \cdot g_2(z_4)] \]
    \[ \vdots \]
n-2. step: \[ g_{n-2}(z_{n-1}) = f_x(z_{n-1}) \ast [s(z_{n-1}) \cdot g_{n-3}(z_{n-1})] \]
last step: calculate \[ \int_0^\infty |z_{n-1}| \cdot g(z_{n-1}) \cdot g_{n-2}(z_{n-1}) \, dz_{n-1} \tag{9} \]

The function $g(z_{n-1}) = \int_0^\infty f_x(z_{n-1} \cdot (z_{n-1} - 1)) \, dz_n$ in the last step of the recursive scheme above can be determined separately by visualizing $f_x(z_{n-1} \cdot (z_{n-1} - 1))$ (see Fig. 6 and 5).

\[ g(z_{n-1}) = \int_{-\infty}^0 f_x(z_{n-1} \cdot (z_{n-1} - 1)) \, dz_n \]

\[ = \begin{cases} 
1 - \frac{p}{2\Delta x} \cdot \frac{2\Delta x}{z_{n-1}} & \text{for } \frac{2\Delta x}{z_{n-1}} > 1, \text{ and } z_{n-1} > 0 \\
0 & \text{for } \frac{2\Delta x}{z_{n-1}} < 1, \text{ and } z_{n-1} > 0 \\
\frac{p}{2\Delta x} \cdot \frac{2\Delta x}{z_{n-1}} & \text{for } \frac{2\Delta x}{z_{n-1}} > 1, \text{ and } z_{n-1} < 0 
\end{cases} \tag{10} \]
Since the integration in the last step of Eq. 9 is carried out only for positive $z_{n-1}$ the absolute signs for $|z_{n-1}|$ can be omitted and only the first two results of Eq. 10 are used. Thus, integral for the last step in Eq. 9 can be re-written as follows:

$$\int_{0}^{\infty} |z_{n-1}| \cdot g(\frac{z_{n-1}}{\Delta x}) \cdot g_{n-2}(z_{n-1}) \, dz_{n-1} =$$

$$= (1 - p) \int_{0}^{2\Delta x} \left(1 - \frac{z_{n-1}}{2\Delta x}\right) \cdot g_{n-2}(z_{n-1}) \, dz_{n-1} \quad (11)$$

Although a closed form solution for these calculations is possible, it would be far too time-consuming. For this reason, the whole scheme has been programmed in MATLAB and carried out recursively. A comparison between the results achieved with a C-programmed computer simulation of the event generator and the theoretical justification, as elaborated above, is shown in Fig. 2. Obviously, there are only marginal differences, which increase for low probabilities and small values of $p$. This is certainly due to the shortcomings of numerical calculations. However, the results show, that for long event lengths $n$ the probability of occurrence decreases rapidly with $p$ and $n$. For example, with $p = 0.35$ and $n = 150$ this probability is $\approx 10^{-6}$. Note that the vertical scale is logarithmic.

**Calculation of the probability for events to exceed a certain amplitude.**

In order to get a feeling for the amplitudes of events of considerably long duration it is necessary to calculate the probability for the $n$-th sample to exceed a given value $y_s$ and all previous samples to be $> 0$, i.e.:

$$P([y(n) > y_s] \cap [y(n-1) > 0] \cap [y(n-2) > 0] \cap \ldots \cap [y(2) > 0] \cap [y(1) > 0]) =$$

$$= \int_{y_s}^{\infty} \int_{0}^{\infty} \ldots \int_{0}^{\infty} f(y(n), y(n-1), \ldots , y(2), y(1)) \, dy_n \, dy_{n-1} \ldots dy_1$$

$$= \int_{y_s}^{\infty} \int_{0}^{\infty} \ldots \int_{0}^{\infty} f_{\mathbf{y}}(y_n - y_{n-1}) \cdot \ldots \cdot f_{\mathbf{y}}(y_2 - y_1) \cdot f_{\mathbf{y}}(y_1) \, dy_n \, dy_{n-1} \ldots dy_1 \quad (12)$$

This $n$-fold integration can be re-arranged in a similar way as shown in Eq. 8 for the first $n - 1$ integrals and the same recursive scheme of Eq. 9 for the first $n - 1$ steps can be used except for the last $n'$th step. The highest possible value of $y(n)$ is $y_{\text{max}} = 2n \cdot \Delta x \forall n$, which equals the upper limit of the last integral. Thus, the whole scheme can be calculated numerically and two results are shown Fig. 3.