

Estimating Probability of Failure of a Complex System Based on Inexact Information about Subsystems and Components, with Applications to Aircraft Maintenance

Christelle Jacob¹, Didier Dubois², Janette Cardoso¹, Martine Ceberio³, Vladik Kreinovich³, and Ildar Batyrshin⁴

¹ Institut Supérieur de l'Aéronautique et de l'Espace (ISAE), DMIA department,
Campus Supaéro, 10 avenue Édouard Belin, Toulouse, France
`jacob@irit.fr`, `cardoso@isae.fr`

² Institut de Recherche en Informatique de Toulouse (IRIT), ADRIA department,
118 Route de Narbonne 31062 Toulouse Cedex 9, France
`dubois@irit.fr`

³ University of Texas at El Paso, Computer Science Dept., El Paso, TX 79968, USA
`mceberio@utep.edu`, `vladik@utep.edu`

⁴ Instituto Mexicano de Petróleo, Ejec Central Lázaro Cardenas Norte 152, Col. San Bartolo Atepehuacan México D.F., C.P. 07730 `batyr@imp.mx`

Abstract. In many real-life applications (e.g., in aircraft maintenance), we need to estimate the probability of failure of a complex system (such as an aircraft as a whole or one of its subsystems). Complex systems are usually built with redundancy allowing them to withstand the failure of a small number of components. In this paper, we assume that we know the structure of the system, and, as a result, for each possible set of failed components, we can tell whether this set will lead to a system failure. For each component A , we know the probability $P(A)$ of its failure with some uncertainty: e.g., we know the lower and upper bounds $\underline{P}(A)$ and $\bar{P}(A)$ for this probability. Usually, it is assumed that failures of different components are independent events. Our objective is to use all this information to estimate the probability of failure of the entire the complex system. In this paper, we describe a new efficient method for such estimation based on Cauchy deviates.

Keywords: complex system, probability of failure, interval uncertainty

1 Formulation of the Problem

It is necessary to estimate the probability of failure for complex systems. In many practical situations applications, we need to estimate the probability of failure of a complex system. The need for such estimates come from the fact that in practice, while it is possible (and desirable) to minimize the risk, it is not possible to completely eliminate the risk: no matter how many precautions we take, there are always some very low probability events that can potentially lead

to a system's failure. All we can do is to make sure that the resulting probability of failure does not exceed the desired small value p_0 . For example, the probability of a catastrophic event is usually required to be at or below $p_0 = 10^{-9}$.

For example, in aircraft design and maintenance, we need to estimate the probability of a failure of an aircraft as a whole and of its subsystems. On the design stage, the purpose of this estimate is to make sure that this probability of failure does not exceed the allowed probability p_0 . On the maintenance stage, we perform this estimate to decide whether a maintenance is needed: if the probability of failure exceeds p_0 , this means that we need to perform some maintenance to decrease this probability to the desired level p_0 (or below).

Information available for estimating system's probability of failure: general description. Complex systems consist of subsystems, which, in turn, consists of components (or maybe of sub-subsystems which consist of components). So, to estimate the probability of failure of a complex system, we need to take into account:

- when the failure of components and subsystems lead to the failure of the complex system as a whole, and
- how reliable are these components and subsystems.

When the failure of components and subsystems lead to the failure of the complex system as a whole? Complex systems are usually built with redundancy allowing them to withstand the failure of a small number of components. Usually, we know the structure of the system, and, as a result, for each possible set of failed components, we can tell whether this set will lead to a system failure. So, in this paper, we will assume that this information is available.

How reliable are components and subsystems? What do we know about the reliability of individual components? For each component A , there is a probability $P(A)$ of its failure. When we have a sufficient statistics of failures of this type of components, we can estimate this probability as the relative frequency of cases when the component failed. In some case, we have a large number of such cases, and as a result, the frequency provides a good approximation to the desired probability – so that, in practice, we can safely assume that we know the actual values of these probabilities $P(A)$.

In some case, we only have a few failure cases, not enough to get an accurate estimate for $P(A)$. In this case, the only information that we can extract from the observation is the interval $\mathbf{P}(A) = [\underline{P}(A), \overline{P}(A)]$ that contains the actual (unknown) value of this probability.

This situation is rather typical for aircraft design and maintenance, because aircrafts are usually built of highly reliable components – at least the important parts of the aircraft are built of such components – and there are thus very few observed cases of failure of these components.

Component failures are independent events. In many practical situations, failures of different components are caused by different factors. For example, for an aircraft, possible failures of mechanical subsystems can be caused by the material fatigue, while possible failures of electronic systems can be caused by the interference of atmospheric electricity (e.g., when flying close to a thunderstorm). In this paper, we assume that failures of different components are independent events.

What we do in this paper. Our objective is to use all this information to estimate the probability of failure of the entire the complex system. In this paper, we describe a new method for such estimation.

Comment. In this paper, we assumed that failures of different components are independent events. Sometimes, we know that the failures of different components are caused by a common cause; corresponding algorithms are described, e.g., in [1–3, 8].

2 Simplest Case: Component Failures are Independent and Failure Probabilities $P(A)$ Are Exactly Known

Let us start our analysis with the simplest case when the component failures are independent and the failure probabilities $P(A)$ for different components A are known exactly.

As we mentioned, we assume that there exists an efficient algorithm that, given a list of failed components, determines whether the whole system fails or not.

In this case, it is always possible to efficiently estimate the probability P of the system's failure by using Monte-Carlo simulations. Specifically, we select the number of simulations N . Then, for each component A , we simulate a Boolean variable $failing(A)$ which is true with probability $P(A)$ and false with the remaining probability $1 - P(A)$. This can be done, e.g., if we take the result r of a standard random number generator that generates values uniformly distributed on the interval $[0, 1]$ and select $failing(A)$ to be true if $r \leq P(A)$ and false otherwise: then the probability of this variable to be true is exactly $P(A)$.

Then, we apply the above-mentioned algorithm to the simulated values of the variables $failing(A)$ and conclude whether for this simulation, the system fails or not. As an estimate for the probability of the system's failure, we then take the ratio $p \stackrel{\text{def}}{=} f/N$, where f is the number of simulations on which the system failed. From statistics, it is known that the mean value of this ratio is indeed the desired probability, that the standard deviation can be estimated as $\sigma = \sqrt{p \cdot (1 - p)/N} \leq 0.5/\sqrt{N}$, and that for sufficiently large N (due to the Central Limit Theorem), the distribution of the difference $P - p$ is close to normal. Thus, with probability 99.9%, the actual value P is within the three-sigma interval $[p - 3\sigma, p + 3\sigma]$.

This enables us to determine how many iterations we need to estimate the probability P with accuracy 10% (and certainty 99.9%): due to $\sigma \leq 0.5/\sqrt{N}$, to guarantee that $3\sigma \leq 0.1$, it is sufficient to select N for which $3 \cdot 0.5/\sqrt{N} \leq 0.1$, i.e., $\sqrt{N} \geq (3 \cdot 0.5)/0.1 = 15$ and $N \geq 225$. It is important to emphasize that this number of iterations is the same no matter how many components we have – and for complex systems, we usually have many thousands of components.

Similarly, to estimate this probability with accuracy 1%, we need $N = 22,500$ iterations, etc. These numbers of iterations work for all possible values P . In practical applications, the desired probability P is small, so $1 - P \approx 1$, $\sigma \approx \sqrt{P/N}$ and the number of iterations, as determined by the condition $3\sigma \leq 0.1$ or $3\sigma \leq 0.01$, is much smaller: $N \geq 900 \cdot P$ for accuracy 10% and $N \geq 90,000 \cdot P$ for accuracy 1%.

Comment. In many cases, there are also efficient analytical algorithms for computing the desired probability of the system's failure; see, e.g., [4–6, 15].

3 Important Subcase of the Simplest Case: When Components are Very Reliable

In many practical applications (e.g., in important subsystems related to aircrafts), components are highly reliable, and their probabilities of failure $P(A)$ are very small. In this case, the above Monte-Carlo technique for computing the probability P of the system's failure requires a large number of simulations, because otherwise, with high probability, in all simulations, all the components will be simulated as working properly.

For example, if the probability of a component's failure is $P(A) = 10^{-3}$, then we need at least a thousand iteration to catch a case when this component fails; if $P(A) = 10^{-6}$, we need at least a million iterations, etc.

In such situations, Monte-Carlo simulations may take a lot of computation time. In some applications, e.g., on the stage of an aircraft design, it may be OK, but in other case, e.g., on the stage of routine aircraft maintenance, the airlines want fast turnaround, and any speed up is highly welcome.

To speed up such simulations, we can use a re-scaling idea; see, e.g., [8, 9]. Specifically, instead of using the original values $P(A)$, we use re-scaled (larger) values $\lambda \cdot P(A)$ for some $\lambda \gg 1$. The value λ is chosen in such a way that the resulting probabilities are larger and thus, require fewer simulations to come up with cases when some components fail. As a result of applying the above Monte-Carlo simulations to these new probabilities $\lambda \cdot P(A)$, we get a probability of failure $P(\lambda)$.

In this case, one can show that while the resulting probabilities $\lambda \cdot P(A)$ are still small, the probability $P(\lambda)$ depends on λ as $P(\lambda) \approx \lambda^k \cdot P$ for some positive integer k .

Thus, to find the desired value P , we repeat this procedure for two different values $\lambda_1 \neq \lambda_2$, get the two values $P(\lambda_1)$ and $P(\lambda_2)$, and then find both unknown

k and P from the resulting system of two equations with two unknowns: $P(\lambda_1) \approx \lambda_1^k \cdot P$ and $P(\lambda_2) \approx \lambda_2^k \cdot P$.

To solve this system, we first divide the first equation by the second one, getting an equation $P(\lambda_1)/P(\lambda_2) \approx (\lambda_1/\lambda_2)^k$ with one unknown k , and find $k \approx \ln(P(\lambda_1)/P(\lambda_2))/(\lambda_1/\lambda_2)$. Then, once we know k , we can find P as $P \approx P(\lambda_1)/\lambda_1^k$.

4 Monotonicity Case

Let us start with the simplest subcase when the dependence of the system's failure on the failure of components is monotonic. To be precise, we assume that if for a certain list of failed components, the system fails, it will still fail if we add one more components to the list of failed ones. In this case, the smaller the probability of failure $P(A)$ for each component A , the smaller the probability P that the system as a whole will fail. Similarly, the larger the probability of failure $P(A)$ for each component A , the larger the probability P that the system as a whole will fail.

Thus, to compute the smallest possible value $\underline{P}$ of the failure probability, it is sufficient to consider the values $\underline{P}(A)$. Similarly, to compute the largest possible value $\overline{P}$ of the failure probability, it is sufficient to consider the values $\overline{P}(A)$. Thus, in the monotonic case, to compute the range $[\underline{P}, \overline{P}]$ of possible values of overall failure probability under interval uncertainty, it is sufficient to solve two problems in each of which we know probabilities with certainty:

- to compute $\underline{P}$, we assume that for each component A , the failure probability is equal to $\underline{P}(A)$;
- to compute $\overline{P}$, we assume that for each component A , the failure probability is equal to $\overline{P}(A)$.

5 In Practice, the Dependence is Sometimes Non-Monotonic

In some practically reasonable situations, the dependence of the system's failure on the failure of components is non-monotonic; see, e.g., [8]. This may sound counter-intuitive at first glance: adding one more failing component to the list of failed ones suddenly makes the previously failing system recover, but here is an example when exactly this seemingly counter-intuitive behavior makes perfect sense.

To increase reliability, systems include duplication: for many important functions, there is a duplicate subsystem ready to take charge if the main subsystem fails. How do we detect that the main system failed? Usually, a subsystem contains several sensors; sensors sometimes fail, as a result of which their signal no longer reflect the actual value of the quantity they are supposed to measure. For example, a temperature sensor which is supposed to generate a signal proportional to the temperature, if failed, produces no signal at all, which the system

will naturally interpret as a 0 temperature. To detect the sensor failure, subsystems often use statistical criteria. For example, for each sensor i , we usually know the mean m_i and the standard deviation σ_i of the corresponding quantity. When these quantities are independent and approximately normally distributed, then, for the measurement values x_i , the sum $X^2 \stackrel{\text{def}}{=} \sum_{i=1}^n \frac{(x_i - m_i)^2}{\sigma_i^2}$ is the sum of n standard normal distributions and thus, follows the chi-square distributed with n degrees of freedom. So, if the actual value of this sum exceeds the threshold corresponding to confidence level $p = 0.05$, this means that we can confidently conclude that some of the sensors are malfunctioning.

If the number n of sensors is large, then one malfunctioning sensor may not increase the sum X^2 too high, and so, its malfunctioning will not be detected, and the system will fail. On the other hand, if all n sensors fail, e.g., show 0 instead of the correct temperature, each term in the sum will be large, the sum will exceed the threshold – and the system will detect the malfunctioning. In this case, the second redundant subsystem will be activated, and the system as a whole will thus continue to function normally.

This is exactly the case of non-monotonicity: when only one sensor fails, the system as a whole fails; however, if, in addition to the originally failed sensor, many other sensors fail, the system as a whole becomes functioning well.

6 General Case: the Problem is NP-Hard

A possible algorithm. For each component A , by using the formula of full probability, we can represent the probability P of the system's failure as follows:

$$P = P(A) \cdot P(F|A) + (1 - P(A)) \cdot P(F|\neg A),$$

where $P(F|A)$ is the conditional probability that the system fails under the condition that the component A fails, and $P(F|\neg A)$ is the conditional probability that the system fails under the condition that the component A does not fail. The conditional probabilities $P(F|A)$ and $P(F|\neg A)$ do not depend on $P(A)$, so the resulting dependence of P on $P(A)$ is linear. A linear function attains its minimum and maximum at the endpoints. Thus, to find $\underline{P}$ and $\overline{P}$, it is not necessary to consider all possible values $P(A) \in [\underline{P}(A), \overline{P}(A)]$, it is sufficient to only consider two values: $P(A) = \underline{P}(A)$ and $P(A) = \overline{P}(A)$.

For each of these two values, for another component A' , we have two possible options $P(A') = \underline{P}(A')$ and $P(A') = \overline{P}(A')$; thus, in this case, we need to consider $2 \times 2 = 4$ possible combinations of values $P(A)$ and $P(A')$.

In general, when we have k components $A_1, \dots, A_k$, it is sufficient to consider 2^k possible combinations of values $\underline{P}(A_i)$ and $\overline{P}(A_i)$ corresponding to each of these components. This procedure requires times which grows as 2^k . As we mentioned earlier, when k is large, the needed computation time becomes unrealistically large.

Natural question. The fact that the above algorithm requires unrealistic exponential time raises a natural question: is it because our algorithm is inefficient or is it because the problem itself is difficult?

The problem is NP-hard. In the general case, when no assumption is made about monotonicity, the problem is as follows:

- we have a propositional formula F with n variables A_i – each variable A_i is true if the corresponding component fails, and the formula F is true if the system as whole fails;
- for each component i , we know the interval $[\underline{P}(A_i), \overline{P}(A_i)]$ that contains the actual (unknown) $P(A_i)$ that this component fails;
- we assume that the failures of different components are independent events.

Different values $P(A_i) \in [\underline{P}(A_i), \overline{P}(A_i)]$ lead, in general, to different values of the probability P that F is true (i.e., that the system failed). Our objective is to compute the range $[\underline{P}, \overline{P}]$ of possible values of this probability.

It is known that, in general, this problem is NP-hard see, e.g., [8].

7 Case of Narrow Intervals: Towards a Cauchy Deviate Method

Case of narrow intervals. In many practical situations, the intervals $[\underline{P}(A), \overline{P}(A)]$ are narrow. If we represent them in the form

$$[\tilde{P}(A) - \Delta(A), \tilde{P}(A) + \Delta(A)],$$

with $\tilde{P}(A) = \frac{\underline{P}(A) + \overline{P}(A)}{2}$ and $\Delta(A) = \frac{\overline{P}(A) - \underline{P}(A)}{2}$, then values $\Delta(A)$ are small, so we can safely ignore terms which are quadratic or of higher order in terms of $\Delta P(A)$.

Linearization: analysis of the problem. In the case of narrow intervals, the difference $\Delta P(A) \stackrel{\text{def}}{=} P(A) - \tilde{P}(A)$ is bounded by $\Delta(A)$ and thus, also small: $|\Delta P(A)| \leq \Delta(A)$. Hence, we can expand the dependence of the desired system failure probability $P = P(P(A), \dots) = P(\tilde{P}(A) + \Delta P(A), \dots)$ into Taylor series and keep only terms which are linear in $\Delta P(A)$: $P \approx \tilde{P} + \sum_A c_A \cdot \Delta P(A)$, where

$$\tilde{P} \stackrel{\text{def}}{=} P(\tilde{P}(A), \dots) \text{ and } c_A \stackrel{\text{def}}{=} \frac{\partial P}{\partial P(A)}.$$

For those A for which $c_A \geq 0$, the largest value of the sum $\sum_A c_A \cdot \Delta P(A)$ (when $\Delta P(A) \in [-\Delta(A), \Delta(A)]$) is attained when $\Delta P(A)$ attains its largest possible value $\Delta(A)$. Similarly, when $c(A) < 0$, the largest possible values of the sum is attained when $\Delta P(A) = -\Delta(A)$. In both cases, the largest possible value

of the term $c_A \cdot \Delta P(A)$ is $|c_A| \cdot \Delta(A)$. Thus, the largest possible value of P is equal to $\tilde{P} + \Delta$, where

$$\Delta \stackrel{\text{def}}{=} \sum_A |c_A| \cdot \Delta(A).$$

Similarly, one can show that the smallest possible value of P is equal to $\tilde{P} - \Delta$, so the range of possible values of the failure probability P is $[\tilde{P} - \Delta, \tilde{P} + \Delta]$.

We already know how to compute $\tilde{P}$ – e.g., we can use the Monte-Carlo approach. How can we compute Δ ?

How to compute Δ : numerical differentiation and its limitations. A natural idea is to compute all the partial derivatives c_A and to use the above formula for Δ . By definition, c_A is the derivative, i.e.,

$$c_A = \lim_{h \rightarrow 0} \frac{P(\tilde{P}(A) + h, \tilde{P}(B), \tilde{P}(C), \dots) - P(\tilde{P}(A), \tilde{P}(B), \tilde{P}(C), \dots)}{h}.$$

By definition of the limit, this means that to get a good approximation for c_A , we can take a small h and compute

$$c_A = \frac{P(\tilde{P}(A) + h, \tilde{P}(B), \tilde{P}(C), \dots) - P(\tilde{P}(A), \tilde{P}(B), \tilde{P}(C), \dots)}{h}.$$

This approach to computing derivatives is called *numerical differentiation*.

The problem with this approach is that each computation of the value $P(\tilde{P}(A) + h, \tilde{P}(B), \tilde{P}(C), \dots)$ by Monte-Carlo techniques requires a lot of simulations, and we need to repeat these simulations again and again as many times as there are components. For an aircraft, with thousands of components, the resulting increase in computation time is huge. Moreover, since we are interested in the difference $P(\tilde{P}(A) + h, \dots) - P(\tilde{P}(A), \dots)$ between the two probabilities, we need to compute each of these probabilities with a high accuracy, so that this difference would be visible in comparison with the approximation error $\sim 1/\sqrt{N}$ of the Monte-Carlo estimates. This requires that we further increase the number of iterations N in each Monte-Carlo simulation and thus, even further increase the computation time.

Cauchy deviate techniques: reminder. There exists a known techniques for computing the value $\sum_A |c_A| \cdot \Delta(A)$ faster, a techniques based on using Cauchy distribution; see, e.g., [11, 14]. This technique is based on using *Cauchy distribution*, i.e., probability distributions with the probability density $\rho(z) = \frac{\Delta}{\pi \cdot (z^2 + \Delta^2)}$; the value Δ is called the *scale parameter* of this distribution, or simply a *parameter*, for short.

Cauchy distribution has the following property: if z_A corresponding to different A are independent random variables, and each of z_A is distributed according to the Cauchy law with parameter $\Delta(A)$, then their linear combination

$z = \sum_A c_A \cdot z_A$ is also distributed according to a Cauchy law, with a scale parameter $\Delta = \sum_A |c_A| \cdot \Delta(A)$.

Therefore, if we take random variables δ_A which are Cauchy distributed with parameters $\Delta(A)$, then the difference

$$c \stackrel{\text{def}}{=} P(\tilde{P}(A) + \delta_A, \tilde{P}(B) + \delta_B, \dots) - P(\tilde{P}(A), \tilde{P}(B), \dots) = \sum_A c_A \cdot \delta_A$$

is Cauchy distributed with the desired parameter Δ . So, repeating this experiment N_c times, we get N_c values $c^{(1)}, \dots, c^{(N_c)}$ which are Cauchy distributed with the unknown parameter, and from them we can estimate Δ . The bigger N_c , the better estimates we get.

Comment. To avoid confusion, we should emphasize that the use of Cauchy distributions is a computational technique, *not* as assumption about the actual distribution: indeed, we know that the actual value of $\Delta P(A)$ is bounded by $\Delta(A)$, but for a Cauchy distribution, there is a positive probability that the simulated value is larger than $\Delta(A)$.

Cauchy techniques: towards implementation. In order to implement the above idea, we need to answer the following two questions:

- how to simulate the Cauchy distribution;
- how to estimate the parameter Δ of this distribution from a finite sample.

Simulation can be based on the functional transformation of uniformly distributed sample values: $\delta_A = \Delta(A) \cdot \tan(\pi \cdot (r_A - 0.5))$, where r_A is uniformly distributed on the interval $[0, 1]$.

In order to estimate Δ , we can apply the Maximum Likelihood Method

$$\rho(c^{(1)}) \cdot \rho(c^{(2)}) \cdot \dots \cdot \rho(c^{(N_c)}) \rightarrow \max,$$

where $\rho(z)$ is a Cauchy distribution density with the unknown Δ . When we substitute the above-given formula for $\rho(z)$ and equate the derivative of the product with respect to Δ to 0 (since it is a maximum), we get an equation

$$\frac{1}{1 + \left(\frac{c^{(1)}}{\Delta}\right)^2} + \dots + \frac{1}{1 + \left(\frac{c^{(N_c)}}{\Delta}\right)^2} = \frac{N_c}{2}.$$

Its left-hand side is an increasing function that is equal to 0 ($< N_c/2$) for $\Delta = 0$ and $> N_c/2$ for $\Delta = \max |c^{(k)}|$; therefore the solution to this equation can be found by applying a bisection method to the interval $[0, \max |c^{(k)}|]$.

It is important to mention that we assumed that the function P is reasonably linear when the values Δ_A are small: $|\Delta_A| \leq \Delta(A)$. However, the simulated values δ_A may be larger than $\Delta(A)$. When we get such values, we do not use the original function P for them, we use a normalized function that is equal to P within the given intervals, and that is extended linearly for all other values; we will see, in the description of an algorithm, how this is done.

Cauchy deviate technique: main algorithm.

- Apply P to the values $\tilde{P}(A)$ and compute $\tilde{P} = P(\tilde{P}(A), \tilde{P}(B), \dots)$.
- For $k = 1, 2, \dots, N_c$, repeat the following:
 - use the standard random number generator to compute n numbers $r_A^{(k)}$ that are uniformly distributed on the interval $[0, 1]$;
 - compute Cauchy distributed values $c_A^{(k)} = \tan(\pi \cdot (r_A^{(k)} - 0.5))$;
 - compute the largest value of $|c_A^{(k)}|$ so that we will be able to normalize the simulated measurement errors and apply P to the values that are within the box of possible values: $K = \max_A |c_A^{(k)}|$;
 - compute the simulated measurement errors $\delta_A^{(k)} := \Delta(A) \cdot c_A^{(k)} / K$;
 - compute the simulated probabilities $P^{(k)}(A) = \tilde{P}(A) + \delta_A^{(k)}$;
 - estimate $P(P^{(k)}(A), P^{(k)}(B), \dots)$ and then compute

$$c^{(k)} = K \cdot (P(P^{(k)}(A), P^{(k)}(B), \dots) - \tilde{P});$$

- Compute Δ by applying the bisection method to solve the corresponding equation.

Resulting gain and remaining limitation. By using the Monte-Carlo techniques, we make sure that the number of iterations N_c depends only on the accuracy with which we want to find the result and not on the number of components. Thus, when we have a large number of components, this method is faster than numerical differentiation.

The computation time of the new algorithm is smaller, but it is still not very fast. The reason is that the Cauchy method was originally designed for situations in which we can compute the exact value of $P(P^{(k)}(A), P^{(k)}(B), \dots)$. In our problem, these values have to be computed by using Monte-Carlo techniques, and computed accurately – and each such computation requires a lot of iterations.

Final idea. To further decrease the needed number of simulations, we can use the following idea (see, e.g., Section 5.4 of [14]). In our simulations, for each combination of values Δ_A , the corresponding Monte-Carlo simulation produce not the actual probability $P(\tilde{P}(A) + \delta_A, \tilde{P}(B) + \delta_B, \dots)$, but an *approximate* value $\tilde{P}(\tilde{P}(A) + \delta_A, \tilde{P}(B) + \delta_B, \dots) = P(\tilde{P}(A) + \delta_A, \tilde{P}(B) + \delta_B, \dots) + c_n$ that differs from the desired probability by a random variable c_n which is normally distributed with mean 0 and variance $\sigma^2 = \frac{\tilde{P} \cdot (1 - \tilde{P})}{N}$. As a result, the difference $c \stackrel{\text{def}}{=} \tilde{P}(\tilde{P}(A) + \delta_A, \tilde{P}(B) + \delta_B, \dots) - \tilde{P}$ between the two observed probabilities can be represented as $c = c_c + c_n$, where $c_c \stackrel{\text{def}}{=} P(\tilde{P}(A) + \delta_A, \tilde{P}(B) + \delta_B, \dots) - \tilde{P}$ is, as we have mentioned, Cauchy distributed with parameter Δ , while

$$c_n = \tilde{P}(\tilde{P}(A) + \delta_A, \tilde{P}(B) + \delta_B, \dots) - P(\tilde{P}(A) + \delta_A, \tilde{P}(B) + \delta_B, \dots)$$

is normally distributed with mean 0 and known standard deviation σ .

The components c_c and c_n are independent. Thus, for $c = c_c + c_n$, for the characteristic function $\chi(\omega) \stackrel{\text{def}}{=} E[\exp(i \cdot \omega \cdot c)]$, we have

$$E[\exp(i \cdot \omega \cdot c)] = E[\exp(i \cdot \omega \cdot c_c) \cdot \exp(i \cdot \omega \cdot c_n)] = E[\exp(i \cdot \omega \cdot c_c)] \cdot E[\exp(i \cdot \omega \cdot c_n)],$$

i.e., $\chi(\omega) = \chi_c(\omega) \cdot \chi_n(\omega)$, where $\chi_c(\omega)$ and $\chi_n(\omega)$ are characteristic functions of c_c and c_n . For Cauchy distribution and for the normal distribution, the characteristic functions are known: $\chi_c(\omega) = \exp(-|\omega| \cdot \Delta)$ and $\chi_n(\omega) = \exp(-\omega^2 \cdot \sigma^2)$. So, we conclude that $\chi(\omega) = \exp(-|\omega| \cdot \Delta - \omega^2 \cdot \sigma^2)$. Hence, to determine Δ , we can estimate $\chi(\omega)$, compute its negative logarithm, and then compute Δ (see the formula below).

Since the value $\chi(\omega)$ is real, it is sufficient to consider only the real part $\cos(\dots)$ of the complex exponent $\exp(i \cdot \dots)$. Thus, we arrive at the following algorithm:

Algorithm. First, we use a lengthy Monte-Carlo simulation to compute the value. $\tilde{P} = P(\tilde{P}(A), \tilde{P}(B), \dots)$. Then, for $k = 1, 2, \dots, N$, we repeat the following:

- use a random number generator to compute n numbers $r_A^{(k)}$, that are uniformly distributed on the interval $[0, 1]$;
- compute $\delta_A^{(k)} = \Delta_i \cdot \tan(\pi \cdot (r_A^{(k)} - 0.5))$;
- use Monte-Carlo simulations to find the frequency (probability estimate) $\tilde{P}(\tilde{P}(A) + \delta_A^{(k)}, \tilde{P}(B) + \delta_B^{(k)}, \dots)$ and then

$$c^{(k)} = \tilde{P}(\tilde{P}(A) + \delta_A^{(k)}, \tilde{P}(B) + \delta_B^{(k)}, \dots) - \tilde{P};$$

- for a real number $\omega > 0$, compute $\chi(\omega) = \frac{1}{N} \cdot \sum_{k=1}^N \cos(\omega \cdot c^{(k)})$;
- compute $\Delta = -\frac{\ln(\chi(\omega))}{\omega} - \sigma^2 \cdot \frac{\omega}{2}$.

Comment. Of course, we also need, as before, to “reduce” the simulated values δ_A to the given bounds $\Delta(A)$.

Acknowledgments. C. Jacob was supported by a grant from @MOST Prototype, a joint project of Airbus, Institut de Recherche en Informatique de Toulouse (IRIT), Laboratoire d’Analyse et d’Architecture des Systèmes (LAAS), ONERA, and Institut Supérieur de l’Aéronautique et de l’Espace (ISAE). V. Kreinovich was supported by the Nat’l Science Foundation grants HRD-0734825 and DUE-0926721 and by Grant 1 T36 GM078000-01 from the Nat’l Institutes of Health.

The authors are thankful to the anonymous referees for valuable suggestions.

References

1. Ceberio, M., Kreinovich, V., Chopra, S., Longpré, L., Nguyen, H. T., Ludaescher, B., and Baral, C.: Interval-type and affine arithmetic-type techniques for handling uncertainty in expert systems, *Journal of Computational and Applied Mathematics*, 199(2), 403–410 (2007)

2. Chopra, S.: Affine arithmetic-type techniques for handling uncertainty in expert systems, Master's thesis, Department of Computer Science, University of Texas at El Paso, 2005.
3. Chopra, S.: Affine arithmetic-type techniques for handling uncertainty in expert systems, *International Journal of Intelligent Technologies and Applied Statistics*, 1(1), 59–110 (2008)
4. Dutuit, Y., Rauzy, A.: Approximate estimation of system reliability via fault trees, *Reliability Engineering and System Safety*, 87(2), 163–172 (2005)
5. Flage, R., Baraldi, P., Ameruso, F., Aven, T., Zio, E.: Handling of epistemic uncertainties in fault tree analysis: a comparison between probabilistic, possibilistic, and hybrid approaches, In: Bris, S., Guedes Sares, C., Martorell, S. (eds.), *Reliability, Risk and Safety: Theory and Applications*, Proceedings of the European Safety and Reliability Conference ESREL'2009, Prague, September 7–10, 2009 (2010)
6. Guth, M. A.: A probability foundation for vagueness and imprecision in fault tree analysis, *IEEE Transactions on Reliability*, 40(5), 563–570 (1991)
7. Interval computations website <http://www.cs.utep.edu/interval-comp>
8. Jacob, C., Dubois, D., Cardoso, J., Ceberio, M., Kreinovich, V.: Estimating Probability of Failure of a Complex System Based on Partial Information about Subsystems and Components, with Potential Applications to Aircraft Maintenance, Proceedings of the International Workshop on Soft Computing Applications and Knowledge Discovery SCAKD'2011, Moscow, Russia, June 25, 2011 (2011)
9. Jaksurat, P., Freudenthal, E., Ceberio, M., Kreinovich, V.: Probabilistic approach to trust: ideas, algorithms, and simulations, Proceedings of the Fifth International Conference on Intelligent Technologies InTech'04, Houston, Texas, December 2–4, 2004 (2004)
10. Jaulin, L., Kieffer, M., Didrit, O., Walter, E.: *Applied Interval Analysis*, Springer, London (2001)
11. Kreinovich, V., Ferson, S.: A new Cauchy-based black-box technique for uncertainty in risk analysis, *Reliability Engineering and Systems Safety*, 85(1–3), 267–279 (2004)
12. Kreinovich, V., et al.: *Computational Complexity and Feasibility of Data Processing and Interval Computations*, Kluwer, Dordrecht (1997)
13. Moore, R. E., Kearfott, R. B., Cloud, M. J.: *Introduction to Interval Analysis*, SIAM Press, Philadelphia, Pennsylvania (2009)
14. Trejo, R., Kreinovich, V.: Error estimations for indirect measurements: randomized vs. deterministic algorithms for 'black-box' programs, In: Rajasekaran, S., Pardalos, P., Reif, J., Rolim, J. (eds.), *Handbook on Randomized Computing*, Kluwer, 673–729 (2001)
15. Troffaes, M., Coolen, F.: On the use of the imprecise Dirichlet model with fault trees, Proceedings of the Mathematical Methods in Reliability Conference, Glasgow, July 2007 (2007)
16. Walley, P.: *Statistical reasoning with imprecise probabilities*. Chapman & Hall: New York (1991)