A Software Tool for Computing the Probability of a System Failure and the Reliability for a Complex Repairable System in Discrete Time[1]

Andreas Brandt

Wirtschaftswissenschaftliche Fakultät, Humboldt-Universität zu Berlin,
Spandauer Str. 1, D-10178 Berlin, Germany
e-mail: brandt@wiwi.hu-berlin.de

Manfred Brandt

Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB),
Takustr. 7, D-14195 Berlin, Germany
e-mail: brandt@zib.de

Abstract

We consider a monotone repairable system consisting of n different components having exponential distributed lifetimes with parameters $\lambda_\ell,\ \ell = 1, ..., n$. The components will be checked at the same time instants $id_I,\ i = 1, 2, ...$, where d_I is a given length of an interval. Within the check at the end of the time intervals $((i–1)d_I, id_I]$ with probability $r_\ell,\ \ell = 1, ..., n$ a failure of the ℓ-th component is detected and repaired. The failure and detecting mechanisms for the components are assumed to be stochastically independent. The state of the system is given by a monotone structure function. For the stationary probability p_{SF} that a system failure occurs within a fixed interval $((i–1)d_I, id_I]$ we give explicit formulae basing on a straight forward approach as well as on certain differences of reliabilities. The quantity p_{SF}/d_I corresponds to the failure rate of the system. The algorithms were implemented in a C-program, which can be compiled by any C-compiler independently of the used platform in principle. For highly reliable systems the algorithms remain stable. The CPU-time is acceptable up to $n = 20$ components in case of UNIX.

Mathematics Subject Classification (MSC 1991): 60K10, 60J20, 68-04.

Keywords: complex monotone repairable system; exponential life times; discrete time; reliability; stationary failure rate; software tool.

Technical Report: 10. 12. 1997.

1 Description of the system

The considered technical system consists of n different *components* $C_1, \ldots, C_n$ having a random *exponential distributed lifetime*, where the *failure rate* of the ℓ-th component is denoted by λ_ℓ.

The n components will be checked at the (same) time instants $t_i := id_I$, $i = 1, 2, \ldots$, where d_I is a given length of a time interval. Within the check at the end of the time interval $I_i := (t_{i-1}, t_i]$ with probability r_ℓ a failure of the ℓ-th component is detected and repaired and with probability $1 - r_\ell$ not detected. The failure and detecting mechanisms for the single components are

[1] This work was supported by a grant from the BMW Rolls-Royce AeroEngines.

stochastically independent. If the ℓ-th component is functioning at the beginning of the interval $I_i = (t_{i-1}, t_i]$, i.e. immediately after t_{i-1}, then the probability that the ℓ-th component fails within this interval is given by

$$c_\ell = 1 - \exp(-\lambda_\ell d_I). \tag{1.1}$$

Let

$$X_\ell(t) := \begin{cases} 1, & \text{if the } \ell\text{-th component is up (functioning) at the time instant } t \\ 0, & \text{if the } \ell\text{-th component is down (fails) at the time instant } t \end{cases}$$

describe the state of the ℓ-th component. The vector process

$$X(t) := (X_1(t), \ldots, X_n(t)), \qquad t \in \mathbb{R}_+ \tag{1.2}$$

gives the states of the n components and has the state space

$$\mathbb{X}_n = \{0, 1\}^n. \tag{1.3}$$

The state of the technical system is given by the states $x_1, \ldots, x_n$ of the single components and by the structure function

$$\phi : \mathbb{X}_n \to \{0, 1\}, \tag{1.4}$$

cf. e.g. [BP], [B], [BF], [SS]. Here $z = \phi(x_1, \ldots, x_n) = 1$ (0) means, that the system is up (down). We suppose that the system is monotone, i.e.

$$\phi(x_1, \ldots, x_n) \le \phi(y_1, \ldots, y_n) \quad \text{for} \;\; x_\ell \le y_\ell, \; \ell = 1, ..., n;$$

$$\phi(0, ..., 0) = 0, \;\; \phi(1, ..., 1) = 1.$$

The structure function can be given e.g. by a table, by an arithmetical representation or by its minimal path vectors, cf. e.g. [BP], [B].

If the components of the system have the reliabilities $p_\ell = P(X_\ell = 1)$, $\ell = 1, \ldots, n$, then the reliability of the system with the random state vector $X = (X_1, \ldots, X_n)$ is given by

$$P(\phi(X) = 1) = \sum_{x \in \mathbb{X}_n} \phi(x) \prod_{\ell=1}^{n} p_\ell^{x_\ell} (1 - p_\ell)^{1 - x_\ell}, \tag{1.5}$$

cf. [BP], p. 17, [B], p. 78.

Now, let us consider the stationary regime of the components. If the sequences

$$
\begin{aligned}
X(t_i+0) &= (X_1(t_i+0),\dots,X_n(t_i+0)), \quad i=0,1,2,\dots,\\
X(t_i-0) &= (X_1(t_i-0),\dots,X_n(t_i-0)), \quad i=1,2,\dots
\end{aligned}
$$

are stationary, then the technical system is in a statistical equilibrium. This stationarity is approximately fulfilled after a longer time of operation of the system. Then obviously,

$$
\begin{aligned}
Z(t_i+0) &:= \phi(X_1(t_i+0),\dots,X_n(t_i+0)), \quad i=0,1,2,\dots, && (1.6)\\
Z(t_i-0) &:= \phi(X_1(t_i-0),\dots,X_n(t_i-0)), \quad i=1,2,\dots && (1.7)
\end{aligned}
$$

are stationary sequences, too. The stationary probability p_{SF} that the system fails within the interval $I_1=(0,t_1]$ is given by

$$p_{SF} = P(Z(t_0+0)=1,\ Z(t_1-0)=0).$$

The quantity $\lambda_{SF} := p_{SF}/d_I$ corresponds to the system failure rate. Because of the stationarity p_{SF} is also the probability that the system fails within a fixed interval $I_i=(t_{i-1},t_i]$, $i=1,2,\dots$ and equal to the probability p_{SR} that the system changes from the down-state to the up-state by repair at the end of a fixed interval $I_i=(t_{i-1},t_i]$, $i=1,2,\dots$ Representations of p_{SF} are given by the formulae (6.14), (6.16) and (6.22) in [BB], part III:

$$
\begin{aligned}
p_{SF} &= \sum_{x,y\in\mathbb{X}_n}(1-\phi(x))\phi(y)\prod_{\ell=1}^{n}\pi_\ell^-(x_\ell)\Big(1\!\!1\{x_\ell=y_\ell\}(1-r_\ell)+1\!\!1\{y_\ell=1\}r_\ell\Big), && (1.8)\\
p_{SF} &= \sum_{x,y\in\mathbb{X}_n}\phi(x)(1-\phi(y))\prod_{\ell=1}^{n}\pi_\ell(x_\ell)\Big(1\!\!1\{x_\ell=y_\ell\}(1-c_\ell)+1\!\!1\{y_\ell=0\}c_\ell\Big), && (1.9)\\
p_{SF} &= \sum_{x\in\mathbb{X}_n}\phi(x)\Big(\prod_{\ell=1}^{n}\pi_\ell(1)^{x_\ell}\pi_\ell(0)^{1-x_\ell}-\prod_{\ell=1}^{n}\pi_\ell^-(1)^{x_\ell}\pi_\ell^-(0)^{1-x_\ell}\Big), && (1.10)
\end{aligned}
$$

where $\pi_\ell(x_\ell)$ denotes the stationary probability that the ℓ-th component is in state x_ℓ at the time instants t_i+0, i.e. immediately after t_i, and $\pi_\ell^-(x_\ell)$ denotes the stationary probability that the ℓ-th component is in state x_ℓ at the time instants t_i-0, i.e. immediately before t_i. These probabilities are given by (2.5) and (2.4) in [BB], part III:

$$\pi_\ell(0)=\frac{c_\ell(1-r_\ell)}{c_\ell+r_\ell-c_\ell r_\ell}, \qquad \pi_\ell(1)=\frac{r_\ell}{c_\ell+r_\ell-c_\ell r_\ell}, \tag{1.11}$$

$$\pi_\ell^-(0)=\frac{c_\ell}{c_\ell+r_\ell-c_\ell r_\ell}, \qquad \pi_\ell^-(1)=\frac{r_\ell(1-c_\ell)}{c_\ell+r_\ell-c_\ell r_\ell}. \tag{1.12}$$

Because of

$$\sum_{x\in\mathbb{X}_n}\prod_{\ell=1}^{n}\pi_\ell(1)^{x_\ell}\pi_\ell(0)^{1-x_\ell}=\sum_{x\in\mathbb{X}_n}\prod_{\ell=1}^{n}\pi_\ell^-(1)^{x_\ell}\pi_\ell^-(0)^{1-x_\ell}=1$$

from (1.10) we get the following representation of p_{SF} via dual reliabilities:

$$p_{SF} = \sum_{x\in\mathbb{X}_n} (1-\phi(x))\Big(\prod_{\ell=1}^{n} \pi_\ell^-(1)^{x_\ell}\pi_\ell^-(0)^{1-x_\ell} - \prod_{\ell=1}^{n} \pi_\ell(1)^{x_\ell}\pi_\ell(0)^{1-x_\ell}\Big). \tag{1.13}$$

Remark. A straight forward computation of the probability that the system fails within an interval yields (1.9) and the computation of the probability that the system changes from the down-state to the up-state yields (1.8). Formula (1.10) can be considered as the difference of the stationary reliabilities at the beginning and the end of an interval and follows in view of the dynamics and the monotonicity of the system:

$$\begin{aligned} p_{SF} &= P(Z(t_0+0)=1,\ Z(t_1-0)=0) \\ &= P(Z(t_0+0)=1) - P(Z(t_0+0)=1,\ Z(t_1-0)=1) \\ &= P(Z(t_0+0)=1) - P(Z(t_1-0)=1). \end{aligned}$$

This result is a discrete time analogue of a corresponding result for multicomponent repairable systems in continuous time proved by Ross [R], cf. also [BF], [FS], [BFL], [BS].

Compared to (1.10) the representation (1.13) of p_{SF} has in case of highly reliable systems the advantage that its use avoids the computation of differences of numbers much larger than the quantity to be computed as far as possible. For numerical stability and high accuracy over a wide range of parameters the use of formulae (1.8) or (1.9) seems to be the better one. But the computation of p_{SF} using these formulae has the disadvantage of a much higher complexity. For an efficient numerically stable computation of p_{SF} for $n \geq 10$ the representation (1.8) is used in the form

$$\begin{aligned} p_{SF} = &\sum_{x\in\mathbb{X}_n} (1-\phi(x))\Big(\prod_{\ell=1}^{n} \pi_\ell^-(x_\ell)\Big) \\ &\Bigg(\sum_{y_1,\dots,y_{n-8}\in\{0,1\}} \Big(\prod_{\ell=1}^{n-8}\big(1\!\!1\{x_\ell=y_\ell\}(1-r_\ell)+1\!\!1\{y_\ell=1\}r_\ell\big)\Big) \\ &\qquad \sum_{y_{n-7},\dots,y_n\in\{0,1\}} \phi(y)\Big(\prod_{\ell=n-7}^{n}\big(1\!\!1\{x_\ell=y_\ell\}(1-r_\ell)+1\!\!1\{y_\ell=1\}r_\ell\big)\Big)\Bigg) \end{aligned} \tag{1.14}$$

or the representation (1.9) is used in the form

$$\begin{aligned} p_{SF} = &\sum_{x\in\mathbb{X}_n} \phi(x)\Big(\prod_{\ell=1}^{n} \pi_\ell(x_\ell)\Big) \\ &\Bigg(\sum_{y_1,\dots,y_{n-8}\in\{0,1\}} \Big(\prod_{\ell=1}^{n-8}\big(1\!\!1\{x_\ell=y_\ell\}(1-c_\ell)+1\!\!1\{y_\ell=0\}c_\ell\big)\Big) \\ &\qquad \sum_{y_{n-7},\dots,y_n\in\{0,1\}} (1-\phi(y))\Big(\prod_{\ell=n-7}^{n}\big(1\!\!1\{x_\ell=y_\ell\}(1-c_\ell)+1\!\!1\{y_\ell=0\}c_\ell\big)\Big)\Bigg) \end{aligned} \tag{1.15}$$

depending on the ratio of the number of states, where the system is up, i.e. where $\phi(x) = 1$, and of the number of all possible states of the system. For programming the states $x \in \mathbb{X}_n$ are translated into unsigned integers by the correspondence

$$(x_1, \ldots, x_n) \in \mathbb{X}_n \longrightarrow \sum_{i=1}^{n} x_i 2^{i-1} \in \{0, 1, \ldots, 2^n - 1\}.$$

In this way the fast processing of integers by most of the processors can be used. Moreover, the programming language C allows an efficient manipulation of single bits.

2 The program PSF.C

The program PSF.C allows the computation of the stationary probability p_{SF} of a system failure within a fixed interval $I_i = (t_{i-1}, t_i]$ and of the reliability of a complex monotone system for up to 15 components. The program is written in the programming language C and can be compiled by any C-compiler independently of the used platform in principle.

Compiling for MSDOS requires the memory model "compact" and using far pointers due to the memory structure. In this case using the preprocessor directive `'#define assembler'` it is possible to link assembler code for a faster numerically stable computation of the stationary probability p_{SF} of a system failure (approximately doubling the speed).

Compiling for UNIX allows the computation for up to 31 components in principle because of the linear memory structure and of the larger region for unsigned integers, but the program would require up to more than 4 gigabytes of RAM. Therefore, in case of UNIX the maximum number of components is limited to 24 by a preprocessor directive (requiring approximately 32 megabytes of RAM for 24 components). Using the GNU-compiler the source code may be compiled by the command line `'gcc -o psf psf.c -O3'`.

If the program "psf" is started with its initializing file (the binary file "psf.ini" in case of MSDOS, ".psfini" in case of UNIX) then the program automatically uses the structure function and the parameters used at the end of the last session and the main menu appears on the screen:

```
Structure function of the system                    (s)
Probability of a system failure within an interval  (p)
Reliability of the system                           (r)
Quit the program                                    (q)
Please input 's', 'p', 'r' or 'q'                    _
```

If the program "psf" is started without its initializing file, e.g. if the program is started for the very first time or if its initializing file has been removed, then it appears the message:

```
Read structure function from file?  (y/n)?           _
```

After the input of `'y'` and of the name of a corresponding file the structure function will be read from this file and the main menu will appear. After the input of `'n'` the user has to input the number of components and then the values of the structure function. The input of any value of the structure function defines automatically also all those values of the structure function which are already known because of the monotonicity of the system. In this state the program can be aborted immediately by the input of `'q'`. After the input of the structure function the main menu will appear.

Within the main menu the input of `'q'` quits the program regularly (the used structure function and the used parameters are written to the initializing file). The input of `'s'`, `'p'` or `'r'` within the main menu leads to sub menus.

2.1 The sub menu for the structure function

Within the main menu the input of `'s'` leads to a sub menu for the structure function:

```
Read the structure function from file              (r)
Input the structure function                       (i)
Edit the structure function                        (e)
k-out-of-n system                                  (k)
Show the minimal path vectors                      (p)
Write the structure function to file               (w)
Back to the main menu                              (m)
Please input 'r', 'i', 'e', 'k', 'p', 'w' or 'm'    _
```

Now, after the input of `'w'` the user is requested for the name of a file, where the structure function will be saved as a binary file. A binary file is chosen for avoiding much larger text files. After the input of `'r'` and of the name of a corresponding file the structure function will be read from this file.

The input of `'i'` allows the input of a new structure function analogously to the input of the structure function after the very first start of the program. The input of any value of the structure function defines again automatically also all those values of the structure function which are already known because of the monotonicity of the system. Therefore, the input automatically defines a monotone system. In this state by the input of `'q'` it is possible to return to the sub menu immediately without changing the structure function.

The input of `'e'` allows to edit the values of the structure function. Pressing the return key leads to the next value(s) of the structure function. The number of newly displayed values can be changed by the input of `'>n'` or of `'<n'`, where n stands for the number of values to be displayed (a positive integer); `'>n'` means in positive and `'<n'` in negative direction. The input of `'0'` or `'1'` changes the value of the structure function correspondingly and also all those values of the structure function whose change is necessary in order to get a monotone structure function again. Changed values will be labeled by `'***'`. The input of `'q'` quits editing without changing the structure function, the input of `'c'` quits editing and changes the structure function correspondingly.

The input of `'k'` allows the input of the structure function for a k-out-of-n system simply by the input of the number n of components and of the least number k of components in order to get a functioning system. This feature may also be used for the input of other structure functions by editing a suitable k-out-of-n system, cf. the example in Section 3.

After the input of `'p'` the first 10 minimal path vectors are displayed. Pressing the return key leads to the next 10 minimal path vectors, the input of 'q' leads back to the sub menu.

The input of 'm' leads back to the main menu.

2.2 The sub menu for the stationary probability of a system failure within a fixed interval

Within the main menu the input of 'p' leads to a sub menu for the computation of the stationary probability p_{SF} that the system fails within a fixed interval $I_i = (t_{i-1}, t_i]$:

```
Read the parameters of the components from file       (r)
Input the parameters of the components                (i)
Edit the parameters of the components                 (e)
Write the parameters of the components to file        (w)
Probability of a system failure via reliabilities     (p)
Probability of a system failure numerically stable    (s)
Back to the main menu                                 (m)
Please input 'r', 'i', 'e', 'w', 'p', 's' or 'm'       _
```

If the parameters c_ℓ and r_ℓ for $\ell = 1, \ldots, n$, cf. Section 1, are not yet available by the program, e.g. if the structure function has been changed to a structure function for a larger system within the session, then in this sub menu the items 'e', 'w', 'p' and 's' are deactivated.

After the input of 'w' the user is requested for the name of a file, where the parameters c_ℓ and r_ℓ for $\ell = 1, \ldots, n$ will be saved as a text file. After the input of 'r' and of the name of a corresponding file these parameters will be read from this file.

The input of 'i' allows the input of the parameters c_ℓ and r_ℓ for $\ell = 1, \ldots, n$. The input of 'e' allows editing these parameters, changed values of the parameters will be labeled by '***'.

After the input of 'p' the stationary probability p_{SF} that the system fails within a fixed interval $I_i = (t_{i-1}, t_i]$ will be computed fast using formula (1.13) via dual reliabilities. The dual reliabilities instead of the reliabilities are used for avoiding differences of numbers much larger than the quantity to be computed as far as possible. But because of these differences the result may be numerically unstable. In this case it appears the error message:

```
Computation of a system failure via dual
reliabilities is numerically unstable.
```

The input of 's' leads to the computation of p_{SF} using formulae (1.8) or (1.9). The computation of p_{SF} using these formulae yields a result of a high accuracy, but the complexity of this computation is much higher than the complexity of the computation using formula (1.13). Therefore, the computation may take some time depending on the number of components. If the program has been compiled for MSDOS then the input of 'q' aborts the computation of p_{SF} and leads back to the sub menu. Using UNIX the program can always be aborted by 'control c', but in this case at the next session the program will start with the structure function and the parameters used at the beginning of the aborted session. Therefore, in case of a larger system using UNIX it could be useful to save a new structure function and new parameters (by the input of 'w' within the corresponding sub menus) or to quit and restart the program before starting the computation of p_{SF} by `'s'` within the sub menu for the stationary probability of a system failure.

The input of `'m'` leads back to the main menu.

2.3 The sub menu for the reliability of the system

Within the main menu the input of `'r'` leads to a sub menu for the computation of the reliability of the system in case of given reliabilities $p_\ell = P(X_\ell = 1)$ for $\ell = 1, \ldots, n$ of the single components, cf. Section 1:

```
Read the reliabilities of the components from file    (r)
Input the reliabilities of the components             (i)
Edit the reliabilities of the components              (e)
Write the reliabilities of the components to file     (w)
Compute the reliability of the system                 (c)
Back to the main menu                                 (m)
Please input 'r', 'i', 'e', 'w', 'c' or 'm'            _
```

If the parameters $p_\ell = P(X_\ell = 1)$ for $\ell = 1, \ldots, n$ are not yet available by the program, then in this sub menu the items `'e'`, `'w'` and `'c'` are deactivated.

After the input of `'w'` the user is requested for the name of a file, where the reliabilities p_ℓ for $\ell = 1, \ldots, n$ of the components will be saved as a text file. After the input of `'r'` and of the name of a corresponding file these reliabilities will be read from this file.

The input of `'i'` allows the input of the reliabilities p_ℓ for $\ell = 1, \ldots, n$ of the components. The input of `'e'` allows editing these reliabilities, changed values of the reliabilities of the components will be labeled by `'***'` again.

After the input of `'c'` the reliability of the system will be computed fast using formula (1.5) and with high accuracy (internal the dual reliability is computed for higher accuracy in case of highly reliable systems). The reliability and also the dual reliability will be displayed in order to get accurate results for highly reliable systems, too.

The input of `'m'` leads back to the main menu.

3 Example

We consider a structure function describing the state of a system with $n = 9$ components:

$$\begin{aligned}\phi(x_1, \ldots, x_9) = 1 - {} & (1 - x_2x_3x_4x_5x_6x_7x_8x_9)(1 - x_1x_3x_4x_5x_6x_7x_8x_9)\\ & (1 - x_1x_2x_4x_5x_6x_7x_8x_9)(1 - x_1x_2x_3x_5x_6x_7x_8x_9)\\ & (1 - x_1x_2x_3x_4x_6x_8x_9)(1 - x_1x_2x_3x_4x_6x_7x_9)(1 - x_1x_2x_3x_4x_6x_7x_8)\\ & (1 - x_1x_2x_3x_4x_5x_8x_9)(1 - x_1x_2x_3x_4x_5x_7x_9)(1 - x_1x_2x_3x_4x_5x_7x_8)\\ & (1 - x_1x_2x_3x_4x_5x_6x_9)(1 - x_1x_2x_3x_4x_5x_6x_8)(1 - x_1x_2x_3x_4x_5x_6x_7).\end{aligned} \tag{3.1}$$

The input of the structure function can be done within the sub menu for the structure function e.g. by firstly constructing the 8-out-of-9 system (input of 'k') and then editing this structure function (input of 'e'). After the input of the structure function (3.1) the input of 'p' within the sub menu for the structure function yields immediately the minimal path vectors of this structure function:

```
Minimal path vectors:

        (1,1,1,1,1,1,1,0,0)
        (1,1,1,1,1,1,0,1,0)
        (1,1,1,1,1,0,1,1,0)
        (1,1,1,1,0,1,1,1,0)
        (1,1,1,1,1,1,0,0,1)
        (1,1,1,1,1,0,1,0,1)
        (1,1,1,1,0,1,1,0,1)
        (1,1,1,1,1,0,0,1,1)
        (1,1,1,1,0,1,0,1,1)
        (1,1,1,0,1,1,1,1,1)

        (1,1,0,1,1,1,1,1,1)
        (1,0,1,1,1,1,1,1,1)
        (0,1,1,1,1,1,1,1,1)

There are 13 minimal path vectors.
```

Now, we change to the sub menu for the computation of the stationary probability p_{SF} that the system fails within a fixed time interval $I_i = (t_{i-1}, t_i]$ by the input of 'm' and then of 'p'.

We assume that the components have identical failure rates λ_ℓ and identical probabilities r_ℓ of a successful repair at the end of a fixed time interval $I_i := (t_{i-1}, t_i]$. We choose $d_I = 2.5\,h$ and $\lambda_\ell = 10^{-6}\,h^{-1}$, $r_\ell = 0.99$ for $\ell = 1, \ldots, 9$. Formula (1.1) yields $c_\ell \approx 2.5 \cdot 10^{-6}$ for $\ell = 1, \ldots, 9$. After the input of 'i' we are requested for the input of the parameters:

```
Please input 'r', 'i', 'e', 'w', 'p', 's' or 'm'       i

        c(1)=   2.5e-6
        r(1)=   .99
        c(2)=   2.5e-6
        r(2)=   _
```

After the input of the 18 values for the parameters the program automatically returns to the sub menu for the computation of p_{SF}. Now, the input of 'p' yields:

```
Probability of a system failure via reliabilities:     1.721573914e-10
```

The input of 's' yields:

```
Probability of a system failure numerically stable:    1.721573914e-10
```

Both values are computed very fast due to the small number $n = 9$ of components. In this case and apart from extreme parameters also in all other cases both methods of computation have yielded the same value for the probability p_{SF} that the system fails within a fixed interval $I_i = (t_{i-1}, t_i]$. For our example so we get

$$p_{SF} \approx 1.721573914 \cdot 10^{-10}.$$

Thus for the system failure rate we obtain

$$\lambda_{SF} = p_{SF}/d_I \approx 6.886 \cdot 10^{-11}\, h^{-1}.$$

References

[BP] Barlow, R.E., Proschan, F., Statistische Theorie der Zuverlässigkeit. Akademie-Verlag, Berlin 1978. (Transl. from English: Statistical Theory of Reliability and Life Testing, Rhinehart and Winston, New York.)

[B] Beichelt, F., Zuverlässigkeits- und Instandhaltungstheorie. B.G. Teubner, Stuttgart 1993.

[BF] Beichelt, F., Franken, P., Zuverlässigkeit und Instandhaltung. Verlag Technik, Berlin 1983.

[BB] Brandt, A., Brandt, M., Reliability analysis for technical systems with control, repair and maintenance. Technical reports I–III. Berlin 1997.

[BFL] Brandt, A., Franken, P., Lisek, B., Stationary stochastic models. J. Wiley & Sons, Chichester/Akademie-Verlag, Berlin 1990.

[BS] Brandt, A., Sulanke, H., On the calculation and approximation of the mean time between failure of a system with two and three component subsystems. J. Inf. Process. Cybernetics (EIK) 22 (1986) 12, 635–645.

[FS] Franken, P., Streller, A., Reliability analysis of complex repairable systems by means of marked point processes. J. Appl. Probab. 17 (1980) 154–167.

[R] Ross, S.M., On the calculation of asymptotic system reliability characteristics. In: Reliability and Fault Tree Analysis; SIAM, Philadelphia 1975.

[SS] Shaked, M., Shanthikumar, J.G., Reliability and Maintainability. Chapter 13 in Heyman, M.J., Sobel, M.J. (Eds.) Handbooks in OR & MS, Vol. 2. Elsevier Science Publishers B.V. (North-Holland) 1990.

[Z] Zacks, S., Introduction to reliability analysis. Springer-Verlag, New York 1992.