

Selecting Mathematical Software for Dependability Assessment of Computer Systems Described by Stiff Markov Chains

Vyacheslav Kharchenko^{1,2}, Oleg Odarushchenko², Valentina Odarushchenko¹
and Peter Popov³

¹National Aerospace University “KhAI”, Kharkov, Ukraine

{v.kharchenko, v.odarushchenko}@khai.edu

²RPC “Radiy”, Kirovograd, Ukraine

skifs2005@mail.ru

³Centre for Software Reliability, City University London, United Kingdom

ptp@csr.city.ac.uk

Abstract. Markov and semi-Markov models are widely used in dependability assessment of complex computer-based systems. Model stiffness poses a serious problem both in terms of computational difficulties and in terms of accuracy of the assessment. Selecting an appropriate method and software package for solving stiff Markov models proved to be a non-trivial task. In this paper we provide an empirical comparison of two approaches to dealing with stiffness – stiffness avoidance and stiffness-tolerance. The study includes several well known techniques and software tools used for solving Kolmogorov’s differential equations derived from complex stiff Markov models. In the comparison we used realistic cases studies developed by others in the past: i) a computer system with hardware redundancy and diverse software, and ii) a queuing system with a server break-down and repair. The results indicate that the accuracy of the known methods is significantly affected by the stiffness of the Markov models, which led us to developing a procedure (an algorithm) for selecting the optimal method and tool for solving a given stiff Markov model. The algorithm is, also included in the paper.

Keywords. Markov chains, stiffness, stiffness-avoidance, stiffness-tolerance, computer based systems, availability, multi-fragmentation

Key terms. MathematicalModeling, Method, SoftwareSystems

1 Introduction

Dependability of computer systems is assessed using probabilistic models in which reliability and availability are typically used as measures of interest. Markov chains (MC) [1, 2] are often preferred to reliability block diagrams and fault trees as MC can handle well complex situations such as failure/repair dependencies and shared repair resources [3].

Dependability assessment of complex computer systems is an essential part of the development process as it either allows for demonstrating that relevant regulations have been met (e.g. as in safety critical applications) and/or for making informed decisions about the risks due to automation (e.g. in applications when poor dependability may lead to huge financial losses). Achieving these goals, however, requires accurate assessment. Assessment errors may lead to wrong or suboptimal decisions.

System modellers are often interested in *transient measures*, which provide more useful information than steady-state measures. The main computational difficulty when MC are used is the size of the models (i.e. their largeness), which is known to affect the accuracy of the transient numerical analysis.

It is not unusual for modern complex systems to have a very large state space: often it may consist of tens of thousands of states. Additional difficulty in solving such models is the model *stiffness* [5], which is the focus of this paper. In practice stiffness in models of computer systems is caused by: i) in case of repairable systems the rates of failure and repair differ by several orders of magnitude [4]; ii) fault-tolerant computer systems (CS) use redundancy. The rates of simultaneous failure of redundant components are typically significantly lower than the rates of the individual components [4]; iii) in models of reliability of modular software the modules' failure rates are significantly lower than the rates of passing the control from a module to a module [4].

In practice it is useful to detect the model stiffness as early as possible. If the model is stiff, using a small integration step is usually a necessary step for obtaining an accurate solution. On the other hand, models with moderate stiffness may allow for obtaining accurate solutions without using a small integration step, thus saving computational resources. With some numerical methods decreasing the step of integration may be even counterproductive as it may simply not improve the solution accuracy.

The assessment methods and tools must provide high confidence in the assessment results. In many cases various regulation bodies would require the tools used in the development to be certified to meet stringent quality requirements. The stiffness of an MC can make it difficult to meet this requirement. Careful selection of the method and tools used to solve accurately and efficiently stiff MCs is needed.

In the last 30 years, many approaches have been developed to deal efficiently with the MC stiffness [4, 5, 6, 7]. They can be split into two groups - "stiffness-tolerance" (STA) and "stiffness-avoidance" approaches (SAA) [5]. The main feature of STA is solving a stiff MC using *special numerical methods* that can provide highly accurate results. The limitations of STA are: i) STA cannot deal effectively with large models, and ii) computational efficiency is difficult to achieve when highly accurate solutions are sought. The SAA solution, on the other hand, is based on an *approximation algo-*

rithm which converts a stiff MC to a non-stiff chain first, which typically has a significantly smaller state space [4]. An advantage of this approach is that it can deal effectively with large stiff MCs. Achieving high accuracy with SAA, however, may be problematic.

A number of software tools have been developed and applied to solving models of complex systems such as SHARP, Save [8], Reno, λ Predict, Möbius, etc. Among them is the utility developed by some of the authors of this paper (ODU) developed more than 15 years ago which is based on EXPMETH [21] and has been validated extensively on a range of models [18 - 20]. The utility uses the algorithm of *modified exponential method*. In addition, a number of off-the-shelf mathematical software packages exist which can be used for solving Markov models, e.g. Maple (Maplesoft), Mathematica (Wolfram Research) and MATLAB (Mathworks) which use standard methods for solving differential equations. These math packages enjoy high reputation among the respective customers earned over several decades by providing a wide range of solutions and good support with regular updates.

In this paper we present an empirical study using two systems: i) a computer system with hardware redundancy and diverse software under the assumptions that the rate of failure of software may vary over time, and ii) a queuing system with a server break-down and repair [4]. The solution of the first system is based on the principle of multi-fragmentation [9], one of the efficient methods of solving an MC in case the model parameters change over time. The main idea of this principle is that the MC is represented as a set of fragments with *identical structure*, but which differ in the values of one or more parameters. Each of the systems included in the study is described by a stiff MC. The main difference between the two systems is that the ratio between the stiffness indices (to be defined below) of the first and the second system is 10^3 . In other words we chose them to be quite different in terms of their stiffness index so that we could study if the stiffness index impacts the accuracy of the different methods for solving the respective models. Both systems were solved using STA. The second system was also solved using SAA. Thus we could compare the accuracy of STA and SAA when applied to solving the same (the second) system and how these compare with the exact solution, which for the second system is available in [10], [11].

We report that indeed the stiffness index impacts significantly the accuracy of the solution methods. We also offer a selection procedure which allows one to choose (among the many available for solving stiff MCs) the solution method that provides the best accuracy given the value of the stiffness index of the MC to be solved. We provide also a justification for our recommendations based on the comparison of the different methods when applied to the chosen two systems described by stiff MCs.

The numerical transient analysis of MCs is faced with two computational difficulties – the model stiffness and largeness, which can affect the accuracy of the solutions obtained. Several numerical methods are widely used that address these difficulties, among them the Rosenbrock method [13], [14], the TR-BDF2 [5], [7], [14], the Jensen's method (uniformization) [5], the implicit Runge-Kutta method [5], [6], [12], and the modified Gir method [6].

The Rosenbrock method is the one-step numerical method that has the advantage of being relatively simple to understand and implement. For moderate accuracy (tolerances of order $10^{-4} - 10^{-5}$) and systems of moderate-size ($N \approx 10$) the method allows for obtaining solutions which in terms of achieved accuracy are comparable with the more complex algorithms. If a low accuracy is acceptable, then this method is attractive. When larger systems are solved the Rosenbrock method becomes less accurate and reliable [13].

TR-BDF2 is a second order accurate A-stable and L-stable single step composite method that uses one step of trapezoidal rule (TR) and one step of BDF2 (second order backward difference formula) [7]. [14] demonstrated that TR-BDF2 deals well with increased stiffness and only requires little extra computations as the parameter values or the mission time are increased. TR-BDF2 is also recommended for use if low accuracy is acceptable [5].

The Jensen method (also known as *uniformization* or randomization) [15] involves the computation of Poisson probabilities. It was extensively modified [5], [14], [16] to deal with the stiffness problem. It achieves greater accuracy than TR-BDF2 but still deals poorly with stiffness in extreme cases. [14] recommends that the Jensen method be used only in cases of moderately stiff models.

The implicit Runge-Kutta method is a single step numerical method that deals with the problem of stiffness and is one of the most computationally efficient methods for achieving high accuracy [6].

Also an aggregation/disaggregation technique for transient solution of stiff MCs was developed by K. S. Trivedi and A. Bobbio [4]. The technique can be applied to any MC, for which the transition rates can be grouped into *two separated sets of values*: one of the sets would include the “slow” states and the second set would include the “fast” states [4]. After aggregating the fast transition rates the MC is reduced to a smaller non stiff MC, which can be solved efficiently using a standard numerical technique [4].

In the rest of the paper the method by Trivedi and Bobbio [4] is referred to as an SAA while the other methods surveyed above are referred to as an STA.

The focus of this study is a comparison of the accuracy of the solution obtained with different methods when applied to the same system. Of particular interest is how the solutions are affected by the stiffness of the system under study.

The rest of the paper is organized as follow: in the section 2 we describe formally the stiffness problem and the stiffness index introducing informally the idea of how stiffness index may impact the accuracy of the numerical methods used in solving the systems. In section 3 we present the comparison results. In section 4 we present a procedure for selecting the optimal solution method and tool based on the stiffness index of an MC. In section 5 we present the conclusions and the problems left for future research.

2 Comparative Analysis of Evaluation Techniques

2.1 The Stiffness Problem

Stiffness is an undesirable property of many practical MCs that pose difficulties in finding transient solutions. There is no commonly adopted definition of “stiffness” but a few of the most widely used ones are summarized below.

The Cauchy problem $\frac{du}{dx} = F(x, u)$ is said to be stiff on the interval $[x_0, X]$ if for x from this interval the following condition is fulfilled:

$$s(x) = \frac{\max_{i=1, n} |\operatorname{Re}(\lambda_i)|}{\min_{i=1, n} |\operatorname{Re}(\lambda_i)|} \gg 1, \quad (1)$$

where the $s(x)$ – denotes the index of stiffness (stiffness index) and λ_i – are the eigenvalues of a Jacobian matrix ($\operatorname{Re} \lambda_i < 0, i = 1, 2, \dots, n$) [6].

Also the index of stiffness of an MC was defined in [5], [14] as the product of the largest total exit rate from a state and the length of solution interval ($=\lambda_i t$), where λ_i are the eigenvalues of the Jacobian matrix.

A system of differential equations (DE) is said to be stiff on the interval $[0, t]$ if there exists a solution component of the system that has variation on that interval that is large compared to $1/t$. Thus, the length of the solution interval also becomes a measure of stiffness [14], [17].

We use the index of stiffness in the empirical evaluations that follow as a measure of discrimination between the MCs with different indices of stiffness: high-stiffness ($s(x) \geq 10^3$), moderate-stiffness ($10^2 < s(x) < 10^3$) and low-stiffness ($s(x) \leq 10^2$).

Here we provide an illustration of how the index of stiffness can affect the accuracy achievable by numerical methods of solving a system of the DEs, which describes a stiff MC.

The most common type of a stiff linear system of DEs is the system in which the eigenvalues can be divided into two groups, based on the difference in their modulus values. The eigenvalues of the first group with large modulus values determine the solution behaviour in the boundary layer. Their corresponding components are rapidly decreasing. The eigenvalues of the second group with small modulus values determine the solution behaviour out of the boundary layer. The index of stiffness (1) is the ratio between the maximum value from the first group and the minimum value from the second one.

To describe in detail the influence of this separation on the stability of the numerical methods let us consider a DE matrix with constant coefficients,

$$y' = Ay, A = (a_{ij}), i, j = 1, \dots, M \quad (2)$$

$$y(0) = y_0, y_i = (y_0^i), i = 1, \dots, M \quad (3)$$

Matrix A is a simple-structured matrix, which means that it has M linearly independent eigenvectors and λ_i, e_i – are the eigenvalues and the corresponding eigenvectors of matrix A, respectively. The solution of the Cauchy problem (2) with initial conditions (3) is presented in (4):

$$y(x) = \sum_{i=1}^M C_i e^{\lambda_i x} e_i \tag{4}$$

Each component $C_i e^{\lambda_i x} e_i$ present in the solution (4) is proportional to one of the eigenvectors and is integrated independently of the other components.

If matrix A has large absolute negative eigenvalues a very small step h would be required on the whole integration interval. With a large integration interval this limitation would cause an increase of the local round-off errors, which would become a serious problem if high overall accuracy is sought.

As an example let's consider a system of two differential equations. Without loss of generality let us assume that $|\lambda_1| \gg |\lambda_2|$.

The exact solution (4) will take the following form:

$$y(x) = C_1 e^{\lambda_1 x} e_1 + C_2 e^{\lambda_2 x} e_2 \tag{5}$$

The first component of of the solution will decrease rapidly on the interval $\tau = 1/|\lambda_1|$, and after that will become extremely small. In this interval this component influences the solution $y(x)$. The second component changes on the interval $\tau = 1/|\lambda_2|$. The second interval is much wider than the first one. In this interval the second component influences the solution $y(x)$. In the first interval the rate of solution change is high and it is dominated by the change of the first component. In the second interval, the rate of change is small and is dominated by the change of the second component. As we can see the values of the given coefficients affect the behaviour of the transient solution. On the first interval, the so called *boundary layer*, in order to achieve an accurate solution in the presence of rapid changes, the step-size must satisfy the condition $h \ll 1/|\lambda_1|$. In the second interval the same condition on the step is required, because the corresponding component of the DE must decrease.

The example despite its simplicity provides a clear illustration of how different eigenvalues may lead to the need for changing the step-size in different integration intervals so that accurate results may be obtained. The value of the stiffness index (1) clearly affects the accuracy achievable with a given solution method. The higher the stiffness index the stricter the requirements imposed on the stability of the chosen numerical method.

We study in detail the impact of the stiffness index on the achievable accuracy with different numerical methods using two stiff MCs with substantially different stiffness indices, $s(x)$. The first stiff MC is a model of a computer system with hardware redundancy and diverse software (S₂₂) [21]. The second system is a queuing system with server break-down and repair (M/M/1/k) [5]. The first system is of moderate-stiffness, with $s(x) = 4 \cdot 10^2$ (1), where $\max |\text{Re}(\lambda_i)| = 0.2$ and $\min |\text{Re}(\lambda_i)| = 0.0005$. The MC of the second system is of high-stiffness, with $s(x) = 3.0001 \cdot 10^4$, where $\max |\text{Re}(\lambda_i)| = 3.0001$ and $\min |\text{Re}(\lambda_i)| = 0.0001$. Both MCs are of equal size – 20 states. The

study was conducted using the functions for solving stiff DEs implemented in the mathematical package MATLAB – *ode23s*, *ode23tb*, *ode15s*, which implement the Rosenbrock method, the TR-BDF2 and the method of backward differentiation, respectively.

Table 2. Experiment results

Method	Parameter	$t_1=[0,1000]$	
		S_{22}	M/1/M/m
Rosenbrock	NSS	93	172
	FE	2141	4302
	NLS	279	516
TR-BDF2	NSS	114	210
	FE	269	506
	NLS	361	692
Backward differentiation	NSS	74	150
	FE	111	203
	NLS	89	179

Table 1 summarizes the parameters obtained with the different methods for solving stiff DE: the number of successful steps (NSS), the function evaluations (FE) and the number of solutions of the linear systems (NLS). The time interval is $t_1=[0;1000]$. The Table shows that even when the model largeness is the same the solution for a system of high-stiffness, M/1/M/m, requires nearly twice as many steps, function evaluations and linear system solutions.

2.2 Stiffness-Tolerance and Stiffness-Avoidance Approaches

Stiffness-Tolerance Approach. The main idea of this approach is using methods that are stable for solving stiff models. These can be split into two broadly classes: “classical” numerical methods for solution of stiff DEs and “modified” numerical methods used for finding a solution in special cases.

(a) The classical (non-modified) numerical methods for solving stiff DEs use special single-step and multi-step integration methods. Examples of such methods are the implicit Runge-Kutta, the TR-BDF2, the Rosenbrock method, the exponential method, the implicit Gir method described in [5], [6], [7], [13], respectively. The implicit Runge-Kutta, TR-BDF2 and Rosenbrock method are implemented by several mathematical off-the-shelf software packages and are usually considered the most accurate methods for solving stiff ODEs.

(b) An example of the modified numerical methods is the exponential modified method. The original algorithm was presented in [6] and is based on the evaluation of the matrix exponent. In [6] this method is recommended as one of the most effective algorithms for solving the class of ODE systems with a high value of the Lipchitz constant, and as a special part of a stiff ODE. As a modification part, an automated adaptive step of integration can be implemented [6]. As the method has a multi-step

algorithm the given modification can increase the accuracy of the solution. The amount of computations and the machine time needed for the solution of stiff DEs can be reduced, too [6].

The solution provided by using any numerical method is expected to be accurate. However, typically the result obtained with numerical method include errors coming from different sources, such as: *problem statement error* - an inherent error, due to various simplifications introduced in order to make the problem (analytically) tractable; *truncation error* - the error related to truncating the infinite series after a finite number of terms are computed; *round-off error* - the type of error that arises in every arithmetic operation carried out on a computer; *initial error* - the error related to the presence of approximate parameters in the mathematical formulae. Ideally we would like to control each of these components of the computational error.

Stiffness-Avoidance Approach. The basic idea of this approach is a model transformation by identifying and eliminating the stiffness from the model, which would bring two benefits: i) a reduction of the largeness of the initial MC, and ii) efficiency in solving a non-stiff model using standard numerical methods. The approach was named an aggregation/disaggregation technique for transient solution of stiff MCs. The technique, developed by K. S. Trivedi, A. Bobbio and A. Reibmann [4], [11], can be applied to any MC with transition rates that can be grouped into two separate sets of values – the set of *slow* and the set of *fast* states [4].

While the transformation of the initial stiff MC brings benefits in terms of efficiency, to the best of our knowledge, no systematic study has been undertaken of the impact of the transformation (from a stiff to a non-stiff MC on the accuracy of the solution. In addition, since the method is not supported by standard off-the-shelf tools, the scope for human error in applying it is non-negligible.

3 Examples and Results

3.1 Example Systems Used in the Studies

In this subsection we provide a brief description of the systems that were solved using STA and SAA. The first system was solved using both approaches, while the second one – using only the SAA. The solution of the second system using STA was presented in [11, 12].

System 1: A Fault-Tolerant Computer System. The first system, Fig. 1, used in the study is a fault-tolerant computer system with two hardware channels, on which diverse control software is run.

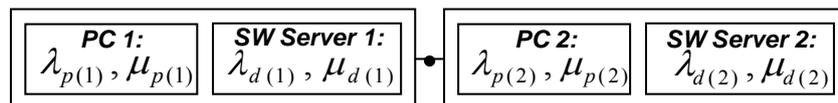


Fig. 1. Reliability block diagram of the chosen fault-tolerant system

In this case increasing system reliability is achieved via redundant hardware-software components with identical hardware structure that supports “hot backup” [21]. We assume that software run of the hardware channels is diverse [18], i.e. non-identical but functionally equivalent software copies are deployed on the hardware channels. The architecture thus offers protection against software design faults. Two channel configurations are very widely used in many safety-critical application, e.g. in instrumentation of nuclear plants, for instance Quad 3000 SIS critical control and safety application. Also similar architectures are used in many business-critical applications, such as the fault-tolerant servers (Blade Server NS50000c, IBM z10, Sun SPARC Enterprise M9000 [21]).

Informally, the operation of the system is as follows. Initially the system is working correctly – both hardware and software channels deliver the service as expected. If during operation one of the hardware channels has failed, the system operation will be failed over to the second channel until the first channel is “repaired”. Similarly, a software component may fail, in which case a failover will take place to the other channel, etc. In addition, we assume that the rates of failure and repair of software will vary over time, e.g. as a result of executing the software in partitions as discussed in [19]. We implement this assumption based on the research work [21]. This assumption captures a plausible phenomenon – variation of software failure rates - which is well accepted in practice: various software ‘aging effects’ are indeed modelled by an increased rate of software failure.

Model Parameters. The model parameters are as follows: *i*) $\lambda_{p(1)}$, $\mu_{p(1)}$ and $\lambda_{p(2)}$, $\mu_{p(2)}$ – hardware failure and repair rates of the first and second hardware channels, respectively; *ii*) $\lambda_{d(1)}$, $\lambda_{d(2)}$ – the initial software failure rate of the 1st and 2nd software versions; *iii*) $\Delta\lambda_d$ – the step of failure rate decrease after the software recovers from a failure; *iv*) $\mu_{d(1)}$ and $\mu_{d(2)}$ – the initial software repair rate of the 1st and the 2nd software versions; *v*) $\Delta\mu_d$ – the step of software repair rate decrease after the software recovers from a failure. We also assume that the values of system failure and repair rates of both the hardware components and of the software versions are equal: $\lambda_{p(1)} = \lambda_{p(2)}$, $\mu_{p(1)} = \mu_{p(2)}$, $\lambda_{d(1)} = \lambda_{d(2)}$, $\mu_{d(1)} = \mu_{d(2)}$ [21].

The Markov transition graph for the system presented in Fig. 1 is shown in Fig. 2. The model is built using the principle of multi-fragmentation [9]. Using this principle the model can be divided into N fragments that are with the same structure but may differ in one or more parameter values [21]. The number of fragments N in the MC depends on the number of expected undetected software faults N_d , the value of which can be estimated using probabilistic prediction models (6) [21]:

$$N = N_d + 1 \quad (6)$$

The sum of the failure rates of both software versions is defined as (7):

$$\Lambda_d = \lambda_{d(1)} + \lambda_{d(2)} \quad (7)$$

This MC of System 1 consists of the following states: $SF_1 = \{S_1, S_4, \dots, S_{n+1}\}$ – the set of states when both the hardware and software on both channels are working correctly, $SF_2 = \{S_2, S_5, \dots, S_{3n+2}\}$ – the set of states in which one of the hardware channel

has failed, $SF_3 = \{S_3, S_6, \dots, S_{3n+3}\}$ – the set of states in which one of the software versions has failed.

The system's operation is described next. At time t_0 the system operates correctly in S_1 . At random moment, t_n , a hardware or a software component failure occurs. In case of a hardware failure the system moves to state S_2 . The rate of this transition is $2\lambda_p$ and recovers from this state back to S_1 with rate μ_p . In case of software failure the system moves to state S_3 and recovers from this failure by moving to state S_4 with rate μ_d . The states S_1, S_2 and S_3 form the first fragment of the model, S_4, S_5 and S_6 – form fragment 2, etc. We assume that the internal fragments rates μ_d and Λ_d decrease by $\Delta\mu_d$ and $\Delta\lambda_d$, respectively, as the system moves between fragments from left to right.

From the (Fig. 2) we derive the matrix of the system transition rates (8), where $i=(1, \dots, n)$ is the number of system fragments:

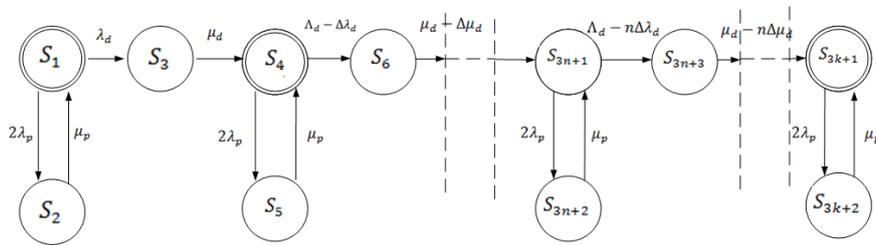


Fig. 2. Model of the system to be studied

$$\begin{pmatrix}
 -(2\lambda_p + \lambda_d) & \mu_p & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 \\
 2\lambda_p & -\mu_p & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 \\
 \lambda_d & 0 & -\mu_d & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & \mu_d & -(\Lambda_d - i\Delta\lambda_d + 2\lambda_p) & \mu_p & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 2\lambda_p & -\mu_p & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & \Lambda_d - i\Delta\lambda_d & 0 & -(\mu_d - i\Delta\mu_d) & \dots & 0 & 0 & 0 & 0 & 0 & 0 \\
 \dots & \dots \\
 0 & 0 & 0 & 0 & 0 & 0 & \dots & -(\mu_d - n\Delta\mu_d) & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & \dots & \mu_d - n\Delta\mu_d & -(\Lambda_d - n\Delta\lambda_d + 2\lambda_p) & \mu_p & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 2\lambda_p & -\mu_p & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & \Lambda_d - n\Delta\lambda_d & 0 & -(\mu_d - n\Delta\mu_d) & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & \mu_d - n\Delta\mu_d & -2\lambda_p & \mu_p \\
 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 2\lambda_p & -\mu_p
 \end{pmatrix} \tag{8}$$

System 2: A Queuing System with Server Breakdown and Repair. The second system was described in details by K.S. Trivedi and A. Bobbio in [4, 10]. The authors consider an M/M/1/k queuing system where m is the system capacity. The first “M” denotes a Poisson arrival process, the second “M” – denoted an exponentially distributed service times. The system has 1 server and k buffers to hold customers waiting for a service. The resulting model operates in 22 states [11]. Fig. 3 shows the Markov transition graph for the system.

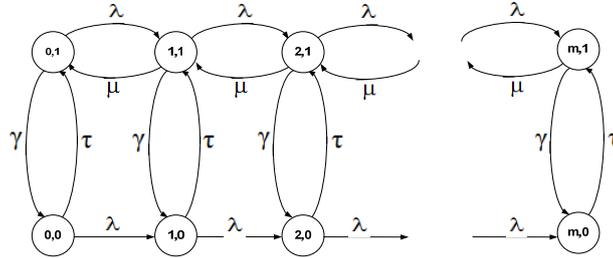


Fig. 3. The state diagram of the M/M/1/m queuing system with a server breakdown and repair

Model Parameters. The λ and μ are the arrival and the service rates, respectively, while γ and τ denote the server failure and failure repair rates. The main assumption is that the rates λ and μ are fast while γ and τ are slow [4]. Using the parameters presented in [11] we computed the index of stiffness for this system to be $s(x) = 3 \cdot 10^4$. Under the terms of this paper this system as classified as a *high-stiffness system*. The system was solved on the same interval $[0, 1000)$ [11].

3.2 Experimental Results

In this Subsection we present the results obtained using the approaches described in Section 2. The solutions for the MC of high-stiffness (the queuing system) is referred to as *Experiment 1*. The solution for the MC of moderate-stiffness (the fault-tolerant computer system) with changing parameters, would be referred to as *Experiment 2*. Lastly, we define an MC of low-stiffness: this is a variant of the fault-tolerant computer system with two hardware channels in which the model parameters have been changed so that the stiffness index has become low ($s(x) = 50$). This solution will be referred to as *Experiment 3*.

Experiment 1. A solution of the *queuing system with sever breakdown and repair* using SAA was presented in [4], [11]. In [4] as a measure of interest the authors used the expected number of customers in the queue, $E[N]$. In [11], in addition, the approximate result (calculated using SAA) and the exact values of $P_{22}(t)$ - the probability of having all buffers full and the server down – were plotted together.

Now we turn our attention to solve the MC of high-stiff using the STA to solve this model. We used two methods that fall into the STA category: using the EXPMETH utility; using the functions for solving stiff DEs implemented in the mathematical package Mathematica.

We used EXPMETH with initial data as follows: the matrix of coefficients was set as defined in [11]; the mission time was set to $t = 1000$; the accuracy of the solution sought was set to 10^{-6} ; the number of steps was set to $n = 20$. As Table 2 illustrates we obtained a negative result, where the probabilities were calculated for every 50 hours of the mission time, $S = \{S_1, S_2, S_3, \dots, S_{20}\}$.

Table 2. Results obtained for Experiment 1 with ODU (EXPMETH)

$t \backslash S_n$	S_1	...	S_{20}
50	1.76231646111739250e+0100- 4.15365304345735515e+0100	...	1.06048815978458797e+0095- 3.22465594380700072e+0094
...
1000	1.88981041657022775e+2054- 4.45414711917994213e+2054	...	1.13720867698699733e+2049- 3.45794216161554014e+2048

The results from using the functions for solving stiff DEs implemented in Mathematica are summarised in Fig. 4 in which the exact solution for $P_{22}(t)$ for $\lambda_1=1.0$, given in [11], is compared with the results obtained with Mathematica.

Based on the empirical evidence with the STA methods applied to *Experiment 1* we conclude that STA cannot provide highly accurate solution for MCs of high-stiffness.

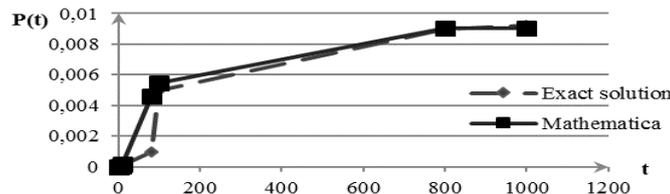


Fig. 4. Results of solving Experiment 1 model using Mathematica

Experiment 2. To solve a moderately-stiff MC (Experiment 2) using SAA we used the algorithm described in [4] and the uniformization method. STA solutions are also obtained using EXPMETH and the mathematical package Mathematica, respectively.

The mission time was set again to $t=1000$, $s(x)=4*10^2 (1)$, where $\max |\text{Re}(\lambda_i)|=0.2$ is the value of μ_d , the software repair rate in the initial model fragment and $\min |\text{Re}(\lambda_i)|=0.0005$ is the value of μ_d for the software repair rate in the final model fragment.

A comparison between the solutions is shown in Table 3 for values of the probabilities that the system is working in each of the states: $\{S_1, S_4, S_7, S_{10}, S_{13}, S_{16}, S_{19}\}$ at $t=500$. This set represents the states without failure (operational states, i.e. both channels work correctly without hardware or software failure).

Fig. 5 shows the results for system availability obtained with EXPMETH and Mathematica. For this system (Experiment 2) we used the result obtained with EXPMETH as an *exact solution* [18]. A discussion of the discrepancies between the solutions obtained with various packages is available in [18].

Based on this experiment we can conclude that for MCs of moderate - stiffness both the SAA and STA can be used. We note that the STA methods would produce an accurate solution faster than SAA when applied to small to moderate systems.

We also note that the particular mathematical package, Mathematica, detects automatically the stiffness of the DE's using a built in "StiffnessTest". In addition the user

of this package can use "StiffnessSwitching", the basic idea as the name suggests being that the package will switch automatically between stiff and non-stiff solvers depending on the outcome of the stiffness test. The non-stiff solver uses the "ExplicitModifiedMidpoint" base method, while the stiff solver uses the "LinearyImplicitEuler" base method [22]. Such special methods can be useful in case of solving an MC of small to moderate size. Finally, we note that the mathematical package offers convenience, but at same time significant effort is required to construct the necessary functions in case of large models, which introduces scope for human errors, e.g. while entering the initial data.

Table 3. Results comparison. System 1

State/t=500	SAA	STA	
		EXPMETH	Mathematica
$S_1(t)$	0,536010	0,537050	0,537052
$S_4(t)$	0,323950	0,321630	0,321634
$S_7(t)$	0,078610	0,078540	0,078542
$S_{10}(t)$	0,010180	0,009810	0,009814
$S_{13}(t)$	0,000640	0,000620	0,000618
$S_{16}(t)$	0,000021	0,000020	0,000023
$S_{19}(t)$	0	0	0

EXPMETH can be more effective in terms of usability. With *Experiment 2* Mathematica required a function with 7 arguments, while EXPMETH only required 4 arguments and a matrix of coefficients of the DEs.

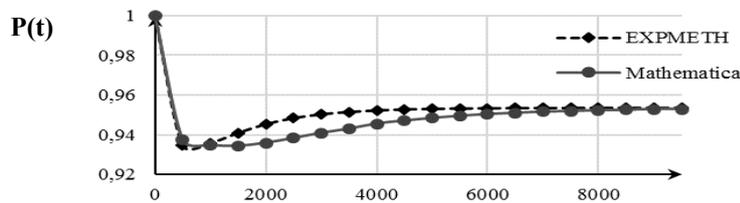


Fig. 5. Comparison of STA methods applied to Experiment 2

Experiment 3. We solved the model of a system with low-stiffness (*Experiment 3*) using the STA only. Based on the justification in Subsection 2.1 we concluded that SAA are the best for solving MCs of high-stiffness and large MCs of moderately-stiffness. In case of MCs of low-stiffness the use of STA can take less time and still provides an accurate solution. As in the previous experiment we consider a mission time $t=1000$, the $s(x)=50$ (1), where $\max |\operatorname{Re}(\lambda_i)|=0.2$ – the value of μ_d software repair rate in the initial model fragment and $\min |\operatorname{Re}(\lambda_i)|=0.004$ – the value of μ_d software repair rate in the final model fragment. Fig. 6 shows the results of the comparison of system availability, $P_a(t)$, obtained with EXPMETH and Mathematica. The

results are practically indistinguishable. Based on this experiment we can conclude that for MCs of low-stiffness the STA can provide an accurate result.

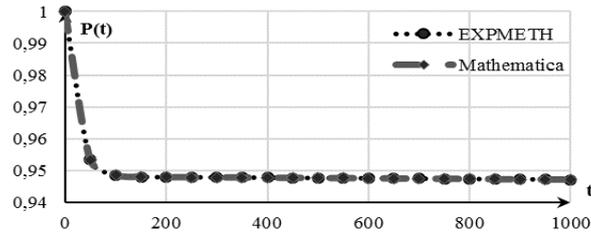


Fig. 6. Comparison of results with STA methods applied to Experiment 3

4 Selection of a Solution Method and of a Software Tool

Based on the results presented in the previous Subsection we propose the following selection procedure (Fig. 7), which takes into account the index of stiffness of the MC under consideration. The first layer of the algorithm takes the index of stiffness, $s(x)$ given by (1), as an initial separator. The system in question is assigned to one of the three classes: high-stiffness, moderate-stiffness or low-stiffness.

An MC of high-stiffness. If the value of $s(x)$ is greater or equal than 10^3 the system model is defined as an MC of high-stiffness. We move to the left branch of the algorithm. If in the system under consideration the parameters change over time, we propose that the principle of multi-fragmentation (MFM) be used. Otherwise this step is skipped. Based on the results from *Experiment 1*, we propose that SAA be used. In this case using specialized software will provide the most accurate solution. The use of STA in this case can produce a solution of low accuracy, which may be unacceptable.

An MC of moderate-stiffness. If the index of stiffness is in the interval $[10^2, 10^3]$ we move to the branch in the middle of the diagram. As in the previous case we propose that MFM be used if in the system under consideration the parameters vary over time. If the parameters do not change the procedure suggests that the initial MC (IMC) be used. On the third layer we propose that the largeness be used as an additional separator. As a theoretical separator the number of system states $n=1000$ can be used. If the number of model states is greater than n – the model is considered large, otherwise the model is not large. To provide effective results in case of a moderately-stiff large MC we propose the use of SAA and specialized tools. Indeed, one of the main features of SAA is the reduction of the system state space. For moderately-stiff MCs which are not large, based on the results of *Experiment 2*, we propose that STA be used with either EXPMETH or other specialized tools.

An MC of low-stiffness. If the index of stiffness is low, $s(x) \leq 10^2$, we move to the right branch of the algorithm. On the second layer we use the same separator as in previous cases. If the system under consideration includes parameter changes then MFM is needed, otherwise it is not needed and the initial MC can be used. In the third layer the system largeness is used as a separator. In case of a large MC of low-

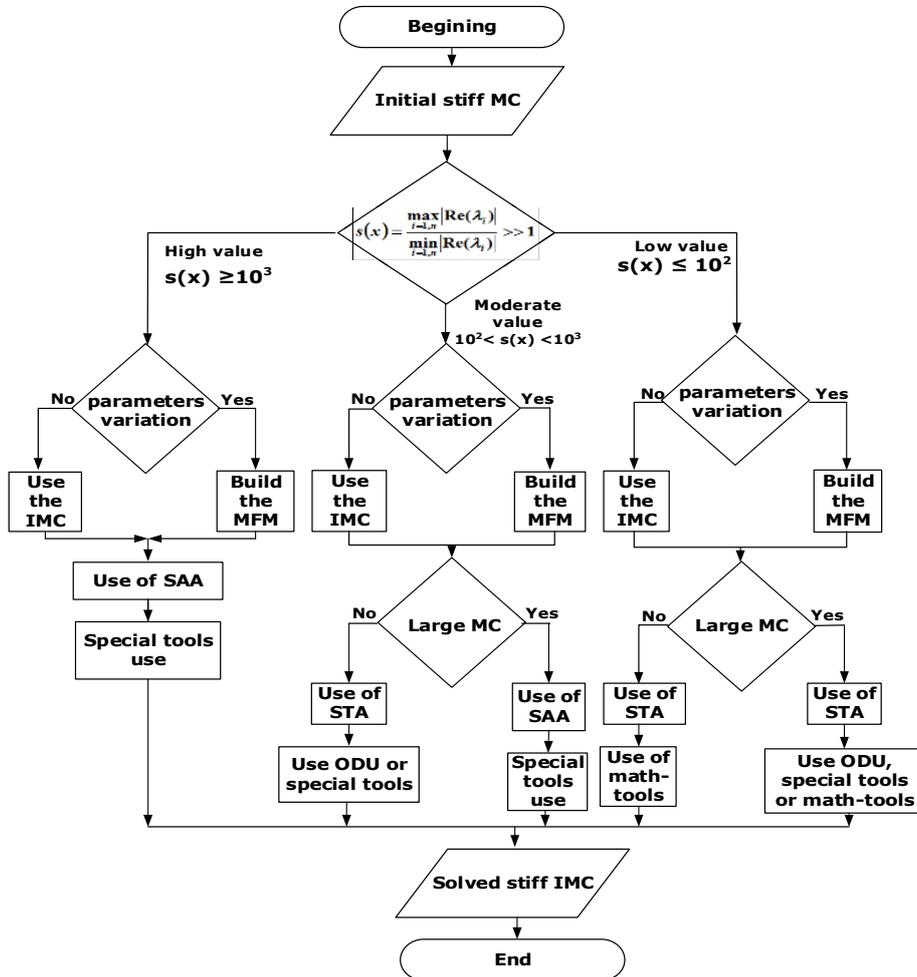


Fig. 7. An algorithm of selecting an optimal method for solving MCs based on the stiffness index and the size (largeness) of an MC

stiffness we propose that STA be used; as a tool we would recommend either EXPMETH or another specialized tool. These specialized tools were developed for special problems solution so they can provide more convenient data representation and satisfy the requirements of high accuracy. In the case of an MC of low-stiffness which is “not large” we propose the use of STA and EXPMETH or mathematical packages.

5 Conclusion

As a result of empirical studies we noticed that the value of stiffness index and the size of the MCs can affect the accuracy of the solutions achievable using different

methods. One of the interesting results is that we can effectively use the SAA to solve a large moderately-stiff MC when the parameters vary, which was the focus in previous research work [18]. In our future work we intend to extend the algorithm presented in the paper and take into account the most effective approach that can deal with large MCs: largeness-tolerant and largeness-avoidance approaches. As a result we are hoping to define the best combination of “largeness-stiffness” approaches that can be applied effectively to systems with variable parameters.

References

1. Volkov, L.: Managing the Operation of the Aircraft Systems: Tutorial. Vyshaya Shkola, Moscow (1981) (In Russian)
2. Ventsel', E., Ovcharov, L.: Probability Theory and its Applications in Engineering. Nauka, Moscow (2000) (In Russian)
3. Archana, S., Srinivasan, R., Trivedi, K. S.: Availability Models in Practice. In: Proc. Int. Workshop on Fault-Tolerant Control and Computing (FTCC-1), May 22-23, Seoul, Korea (2000)
4. Bobbio, A., Trivedi, K. S.: An Aggregation Technique for Transient Analysis of Stiff Markov Chains. IEEE Transactions on Computers, C-35, 803–814 (1986)
5. Malhotra, M., Muppala, J.K., Trivedi, K. S.: Stiffness-Tolerant Methods for Transient Analysis of Stiff Markov Chains. Microelectronic Reliability, 34(11), 1825–1841 (1994)
6. Arushanyan, O., Zaletkin, S.: Numerical Solution of Ordinary Differential Equations using FORTRAN. Moscow State University, Moscow (1990) (In Russian)
7. Bank, R. E. et al.: Transient Simulation of Silicon Devices and Circuits. IEEE Transactions on Electron Devices, 32(10), 1992–2007 (1985)
8. Geist, R., Trivedi, K. S.: Reliability Estimation of Fault-Tolerant Systems: Tools and Techniques. Computer. 23, 52–61 (1990)
9. Kharchenko, V., Timonkin, G., Sychev, V.: Fundamentals of Design and Constructions the Automated Control Systems for Aircraft Technical State Control. Study Guide. KVKIU, Kharkov (1992) (In Russian)
10. Nicola, V.F.: Markovian Models of Transactional System Supported by Check Pointing and Recovery Strategies. Part 1: a Model with State-Dependent Parameters. Eindhoven Univ. Technol., Eindhoven, the Netherlands, EUT Rep. 82-E-128 (1982)
11. Reibman A., Trivedi K. S., Kumar S., Ciardo G.: Analysis of Stiff Markov Chains. ORSA Journal on Computing, 1(2), 126–133 (1989)
12. Hayrer, E., Vanner, G.: Solution of Ordinary Differential Equations. Stiff and Differential-Algebraic Problems. Mir, Moscow (1999) (In Russian)
13. Press, W. H., Teukolsky S. A., Vetterling W. T., Flannery B. P.: Numerical Recipes. The Art of Scientific Computing. 3d edition. Cambridge University Press (2007)
14. Reibman, A., Trivedi, K.S.: Numerical Transient Analysis of Markov models. Comput. Opns. Res., 15(1), 19–36 (1988)
15. Jensen A.: Markoff Chains as an Aid in the Study of Markoff Processes. Skand. Aktuarietidskrift, 36, 87–91 (1953)
16. Fox, B. L., Glynn, P. W.: Computing Poisson probabilities. Commun, ACM 31(4), 440–445 (1985)
17. Miranker, L.: Numerical Methods for Stiff Equations and Singular Perturbation Problems. Dordrecht, Holland (1981)

18. Kharchenko, V., Popov, P., Odarushchenko, O., Zhadan, V.: Empirical Evaluation of Accuracy of Mathematical Software Used for Availability Assessment of Fault-Tolerant Computer Systems. *RT&A* #03(26), 7, 85–97 (2012)
19. Littlewood, B., Popov, P., Strigini, L.: Modelling Software Design Diversity – a Review. *ACM Computing Surveys*, 33(1), 177–208 (2001)
20. Popov, P., Manno, G.: The Effect of Correlated Failure Rates on Reliability of Continuous Time 1-Out-of-2 Software. In: *Proc. Computer Safety, Reliability, and Security (SAFECOMP 2011)*, Naples, Italy (2011)
21. Kharchenko, V., Odarushchenko, O., Ponochovny, Y., Zhivilo, S., Odarushchenko, E., Kharibin, O., Odarushchenko, V.: High Availability Systems and Technologies. In: Kharchenko, V. (ed.). *Lectures, National Aerospace University “KhAI”* (2012)
22. Wolfram Mathematica 9 Documentation Center, <http://reference.wolfram.com/mathematica/tutorial/NDSolveStiffnessSwitching.html>