EMPIRICAL EVALUATION OF ACCURACY OF MATHEMATICAL SOFTWARE USED FOR AVAILABILITY ASSESSMENT OF FAULT-TOLERANT COMPUTER SYSTEMS

Vyacheslav Kharchenko

National Airspace University "KhAI", Kharkov, Ukraine

e-mail: v_s_kharchenko@ukr.net

Peter Popov

Centre for Software Reliability, City University London, United Kingdom

e-mail: ptp@csr.city.ac.uk

Oleg Odarushchenko, Valentina Zhadan

Poltava National Technical University named after Yuri Kondratiuk, Poltava, Ukraine

e-mail: skifs2005@mail.ru, valentinaodarus@gmail.com

ABSTRACT

Dependability assessment is typically based on complex probabilistic models. Markov and semi-Markov models are widely used to model dependability of complex hardware/software architectures. Solving such models, especially when they are stiff, is not trivial and is usually done using sophisticated mathematical software packages.

We report a practical experience of comparing the accuracy of solutions stiff Markov models obtained using well known commercial and research software packages. The study is conducted on a contrived but realistic cases study of computer system with hardware redundancy and diverse software under the assumptions that the rate of failure of software may vary over time, a realistic assumption. We observe that the disagreement between the solutions obtained with the different packages may be very significant. We discuss these findings and directions for future research.

1. INTRODUCTION

Dependability of computer systems is evaluated using probabilistic models in which the measure of interest is typically reliability, availability, etc. Often Markov chains are used in this process [1, 2, 3].

System modelers are often interested in transient measures, which provide more useful information than the steady-state measures. As models grow in size, closed-form solutions of transient measures become infeasible and in practice the models typically are solved using numerical methods.

Accurate dependability assessment of complex computer systems is an important issue. In many cases the accuracy of the assessment, e.g. in safety critical applications or in applications when poor dependability may lead to huge financial losses, is an essential part of the development process. The assessment methods and tools must provide high confidence in the assessment results and in many cases various regulation bodies would require the tools used in development to be certified to meet stringent quality requirements. To the best of our knowledge no such requirements (for software quality) are *not* in place for the software packages used in assessment. The modelers/assessors of complex computer systems are left with the choice – either to use the most accurate assessment algorithms, typically developed by researchers, and use an implementation of those in specific assessment or instead use the solutions available out of the box in the best known off-the-shelf mathematical packages available on the market. The first option – own implementation

of research results is not ideal because of the poor quality of the software code that one should expect from a prototype implementation. The second option for achieving accurate assessment – using off-the-shelf math software – is the focus of this paper.

Among the best known off-the-shelf math packages are Maple (Maplesoft), Mathematica (Wolfram Research) and Mathcad (PTC). 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.

The difficulties with transient numerical analysis of Markov chains have been studied extensively in the past – we survey the important related research. The main difficulty in analysis is the stiffness of the models. Given the extensive work on the issue in 80s and 90s would expect that the availably software packages used by modelers would provide accurate solution to stiff Markov chains. Surprisingly, this does not seem to be the case as this paper illustrates.

The paper is organized as follows: in section 2 we state the problem formally and describe a contrived example of fault-tolerant computer system which we use to compare several well known mathematical packages. In section 3 we present the results obtained. In section 4 we discuss the findings and their implication. Section 5 provides a survey of the important results on solving stiff Markov chains reported by others. Finally, in section 6 we conclude the paper and suggest ways forward.

2. PROBLEM STATEMENT

In this paper we study the accuracy of popular mathematical packages likely to be used in the dependability assessment of complex fault tolerant computer systems. The method of study used is as follows:

- Define a model of the system to be used in comparison;
- Develop solutions for system dependability, e.g. transient system's availability in the interval [0,t], using the instruments available by the chosen packages. As a benchmark solution to compare the solutions obtained with off-the-shelf software packages we used a highly specialized software utility, EXPMETH, which is an implementation of a method for solving stiff Kolmogorov equations [4];
- Compare the obtained solutions.

EXPMETH was developed more than 15 years ago and validated extensively on a range of models [5]. It was developed in Pascal program language and used to assess availability of a chosen system model. We have chosen a system described by a stiff Markov chain: the ratio between the rates of failure and repair ranges between 4 and 8.

The system we chose in the study is a fault-tolerant computer system with two hardware channels each executing software control as shown in Figure 1.

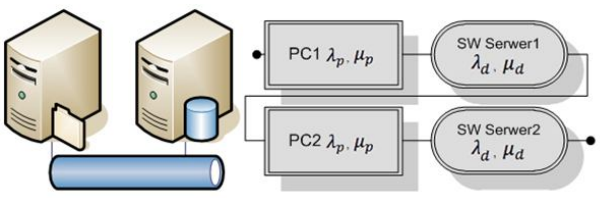

Figure 1. Reliability block diagram of the chosen fault-tolerant system

We assume that software run of the two hardware channels is diverse [5], i.e. non identical but functionally equivalent software copies are deployed, which offers protection against software design faults. 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 [6].

The model to be used in the study is shown in Figure 2. We omit a more detailed justification of the chosen model as this is outside the scope of this paper as the focus of the study is the accuracy of the solutions provided by off-the-shelf math packages. Yet, we would like to stress that the model is plausible. Two channel configurations is very widely used in many safety-critical application, e.g. in instrumentation of nuclear plants. The variation of failure rates and repair is also a plausible concept – software may well perform different tasks with different importance, which would justify different degree of testing, hence different rates of failure and repair in the respective partitions.

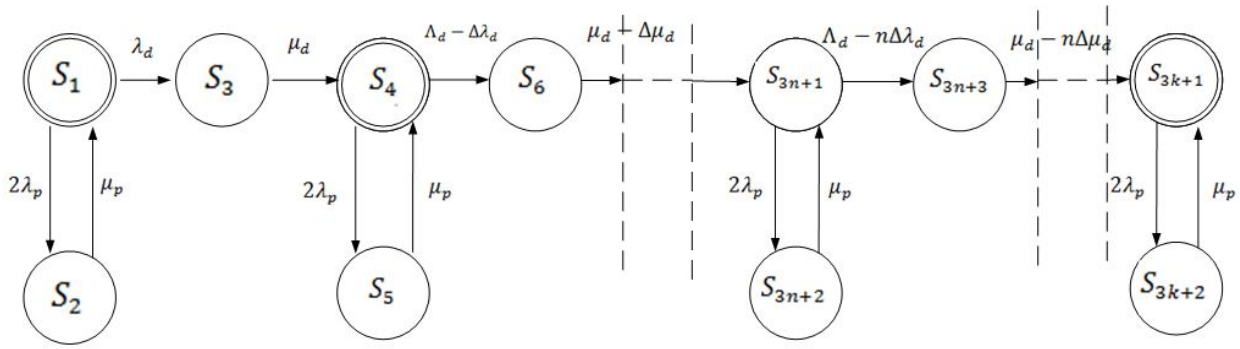

Figure 2. Model of the system to be studied. A 2-hardware channels, 2-software versions fault tolerant computer system with a variation of the rates of software failure and repair.

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 the 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, etc.

An important feature of the model is that as a result of software repair (e.g. restart of the failed channel) we assume that the rate of software failure of both channels will deteriorate by a small constant $\Delta\lambda_d$. Clearly this is just an assumption which might be unrealistic in many cases and its justification might require a detailed analysis of the application that software implements. However, the assumption captures a plausible phenomenon – variation of software failure rates which is well accepted in practice: various software 'aging effects' are indeed modeled by an increased rate of software failure. We conclude therefore, that the model is adequate for our purposes in this paper – a study of the accuracy of the solutions to stiff Markov chains offered by off-the-shelf math packages.

3. MODEL PARAMETERS

The model parameters are defined next:

- λ_p and μ_p – hardware failure and repair rates;
- λ_d and $\Delta\lambda_d$ – software initial failure rate and a step of failure rate decrease after software

recovers from failure;

- μ_d and $\Delta\mu_d$ – initial software repair rate and a step of software repair rate decrease after software recovers from failure.

System's behavior is modelled as a Markov process (Markov chain) that has a number of states, and the transition probabilities between these, $P_k^{(i)}$, depends only on the current state, i.e.

$$P\left\{x^{(i+1)} = x_k^{(i+1)}\Big|_{x^{(0)}\ldots x^{(i)}}\right\} = P\left\{x^{(i+1)} = x_k^{(i+1)}\Big|_{x^{(i)}}\right\} \quad (1)$$

The reader can notice that the model presented in Figure 2 is built with a set of similar fragments – the model fragments are topologically identical and only differ in terms of the values of model parameters.

From the model (Fig. 2) we can derive the following system of Kolmogorov equations:

For the initial fragment these are:

$$\frac{dP_1(t)}{dt} = -(2\lambda_p + \Lambda_d)P_1(t) + \mu_p P_2(t) \quad (2)$$

$$\frac{dP_2(t)}{dt} = -\mu_p P_2(t) + 2\lambda_p P_1(t) \quad (3)$$

$$\frac{dP_3(t)}{dt} = -\mu_d P_3(t) + \Lambda_d P_1(t) \quad (4)$$

For the internal fragments these are:

$$\frac{dP_{3i+1}}{dt} = -(\Lambda_d - i\Delta\lambda_d + 2\lambda_p)P_{3i+1}(t) + (\mu_d - (i-1)\Delta\mu_d)P_{3i}(t) + \mu_p P_{3i+2}(t) \quad (5)$$

$$\frac{dP_{3i+2}}{dt} = -\mu_p P_{3i+2}(t) + 2\lambda_p P_{3i+1}(t) \quad (6)$$

$$\frac{dP_{3i+2}}{dt} = -(\mu_d - i\Delta\mu_d)P_{3i+3}(t) + (\Lambda_d - i\Delta\lambda_d)P_{3i+1}(t) \quad (7)$$

And for the final fragment these are:

$$\frac{dP_{3k+1}}{dt} = -2\lambda_p P_{3k+1}(t) + \Delta\mu_d P_{3k}(t) + \mu_p P_{3k+2}(t) \quad (8)$$

$$\frac{dP_{3k+2}}{dt} = -\mu_p P_{3k+2}(t) + 2\lambda_p P_{3k+1}(t) \quad (9)$$

With the following initial conditions:

$$P_1(0) = 1, P_2(0) = 0, P_3(0) = 0, \ldots, P_{3i+1}(0) = 0, P_{3i+2}(0) = 0$$
$$P_{3i+3}(0) = 0, \ldots, P_{3k+1}(0) = 0, P_{3k+2}(0) = 0. \quad (10)$$

The availability function is defined as the sum of probabilities that system is in one of the states for which at least one of the channels is working, which is defined by the following sum:

$$P_a(\mathrm{t}) = \sum_{i:S_i \in MS_p} P_{S_i}(t) \tag{11}$$

4. RESULTS

Figure 2 represents a stiff Markov chain, characterized by the fact that some of the eigenvalues of Jacobean matrix $\frac{\partial f}{\partial y}$ are large in absolute value with negative real part, while some other eigenvalues are with small positive real part. Getting an accurate solution requires selecting a very small integrating step, which limits the distribution of the error. In such circumstances the classical implicit Runge-Kutta methods provide incorrect result even if a small step of integration is used. The error is caused by the rounding errors which accumulate over the large number of small steps. We consider only the implicit Runge-Kutta methods, because the explicit Runge-Kutta method can provide correct solution even if the system is stiff. In case of solving stiff ordinary differential equations (ODE) the numerical method have to satisfy the following condition:

1. convergence (the method has to converge to ODE);
2. special requirements for stability;
3. the method must pass successfully certain calculation tests.

The software utility EXPMETH implements a special numerical method of solving stiff differential equations – the "exponential" method, studied by O. Arushanyan and S. Zaliotkin [4]. It is based on an accurate representation and calculation of the matrix exponent. The results obtained with EXPMETH were used in the study as a reference solution (to compare the results obtained with the tools/methods). The exponential method was implemented step by step in each of the math packages included in the comparison: Mathcad 15, Maple 15 and Mathematica 8.0.1. We also used the standard solution built in the respective math packages for solving ordinary differential equations as detailed below:

1. In Mathcad 15 we used the built-in function *Stiffr(P,0,10000,D,J)*, with arguments:
   - *P* – the initial state vector of differential equations system;
   - *0, 10000* – time interval on which system availability was computed;
   - *D* – the system of differential equations (defined above);
   - *J* – eigenvalues of the respective Jacobean matrix.
2. In Maple 15 we used the built-in function *DSolve (odesys, numeric, vars, options)*, where:
   - *odesys* – is the set of ODE(s) and the initial/boundary conditions;
   - *numeric* – a parameter used to instruct dsolve to find a numerical solution;
   - *vars* - (optional) can be any indeterminate function of one argument, or a list of them such functions, representing the unknowns of the ODE problem;
   - *options* - equations of the form "keyword = value". In our case this parameter was used to select specify the method of integration, e.g. (stiff=true,method=rosenbrock) were used to set the stiff property to true and select the Rosenbrock method of solving the system of differential equations.
3. In Mathematica 8.0.1 we used the built-in function - *NDSolve{ODE},{t,1,10000}, Method->{"ExplicitRungeKutta}* , where:
   - *{ODE}* – defines the set of ODE(s) and the initial/boundary conditions;

- *{t,1,10000}* – indicates that the solution in time domain is sought on the interval [1,10000];
- *Method->{"ExplicitRungeKutta"}* – the explicit Runge-Kutta method was used.

Figure 3, 4 and 5 show the results obtained for system availability using each of the 3 math packages with the methods described above together with the results obtained using the exponential method computed both in the respective package and using EXPMETH utility.

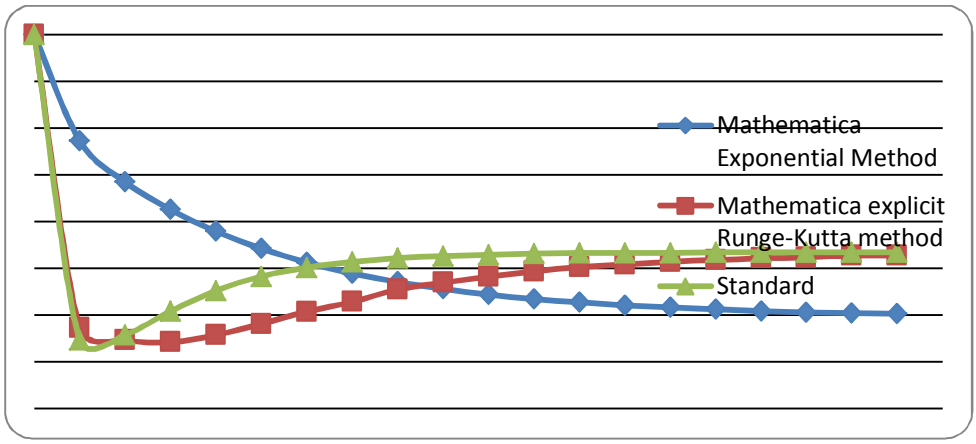

Figure 3. Availability function plots the results obtained with Mathematica 8.0.1 vs. EXPMETH.

The explicit Runge-Kutta method generally follows the solution provided by EXPMETH, while the exponential method is hopelessly inaccurate – starts with gross overestimation of system availability and gradually declines to a significant underestimation of system availability.

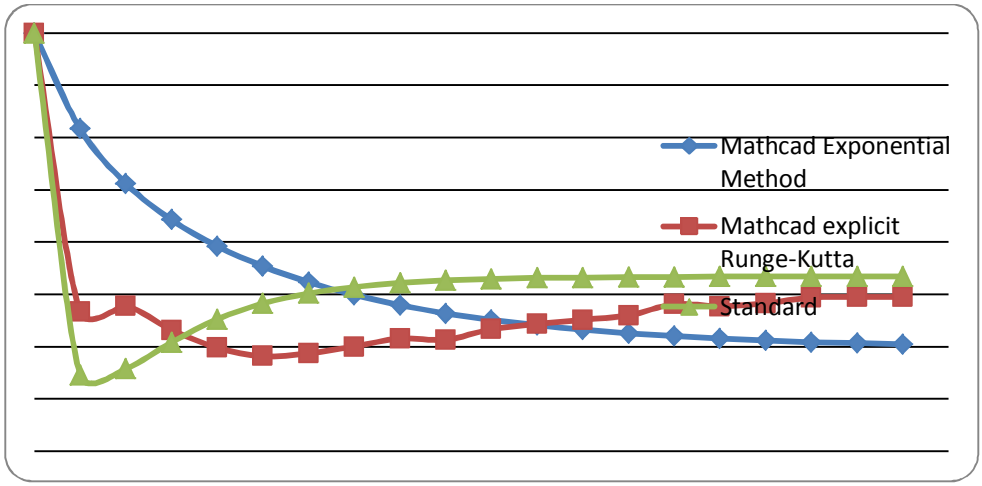

Figure 4. Availability function plots calculated with Mathcad 15 vs. EXPMETH.

The explicit Runge-Kutta method performs worst than in Mathematica although it also generally follows the solution provided by EXPMATH. The exponential method again is very poor – starts with gross overestimation of system availability and gradually declines to a significant underestimation of system availability.

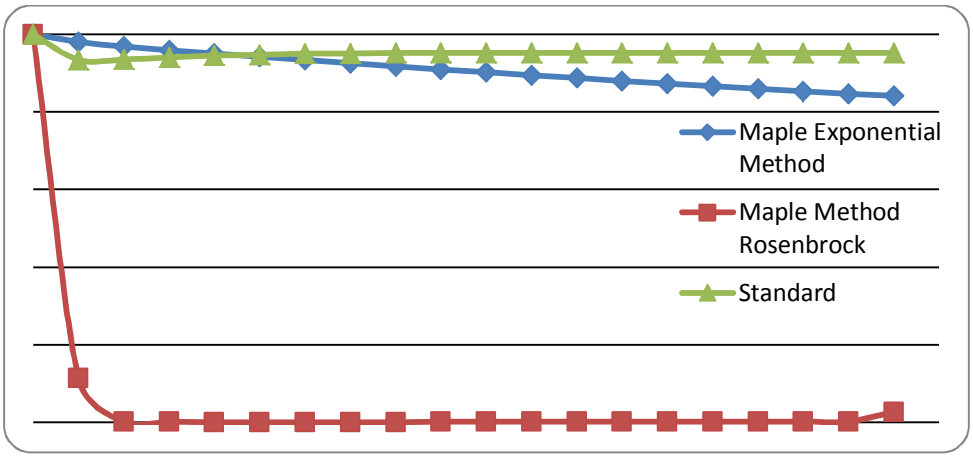

Figure 5. Availability function plots calculated with Maple 15 vs. EXPMETH.

It is really striking how inaccurate the Rosenbrock method is despite being explicitly said to target stiff Markov models. The exponential method shows the same pattern of poor accuracy – starts with gross overestimation of system availability which is gradually replaced by a significant underestimation of system availability.

We also included in the study the simulation solver of the well known tool Mobius, developed and maintained by the Performability group at the University of Illinois at Urbana Champaign (UIUC). Since this solver differs in nature from the other three, we will provide a more extensive description of how the system model shown in Figure 2 was developed using the SAN (stochastic activity networks) formalism of Mobius.

The fragments of the system model discussed above were explicitly specified as *atomic models*: the system model consists of 7 fragments (the initial fragment, 5 internal fragments and the final final fragment). Figure 6, 7, 8 and 9 illustrate of initial, two of the internal fragments and the final fragment build using the Mobius SAN formalism.

Tables 1 and 2 map the fragments to the atomic models: Table 1 shows the mapping between the states. Table 2 – shows the transitions between fragments.

Table 1. Fragments to the atomic models

fragment / state	Initial	Internal 1	Internal 2-5	Final
Both software components (SC) working	sw_working	sw_working	sw_working	sw_working
Both hardware components (HC) working	hw_working	hw_working	hw_working	hw_working
First SC failed	-	sw_fail	sw_failure1	sw_failure1
Second SC failed	-	-	sw_failure2	sw_failure2
First HC failed	hw_w1	hw_w1_int1	hw_w1_int2	hw_w1
Second HC failed	hw_w2	hw_w2_int1	hw_w2_int2	hw_w2
Both SC failed	-	syst_failed	syst_failed	syst_failed
Both HC failed	system_failed	system_failed	system_failed	system_failed

Table 2. Transitions between fragments

fragment / transition	Initial	Internal 1	Internal 2-5	Final
First HC failed	fail1	fail1_int1	fail1_int2	fail1
Second HC failed	fail2	fail2_int1	fail2_int2	fail2
First SC failed	-	sw_fail	sw_fail1	sw_fail1
Second SC failed	-	-	sw_fail2	sw_fail2
Recovery after first HC elimination	recov1	recov1_int2	recov1_int2	recov1
Recovery after second HC elimination	recov2	recov2_int2	recov2_int2	recov2
Recovery after first SC elimination	-	sw_recovery1	sw_recov1	sw_recov1
Recovery after second SC elimination	-	-	sw_recov2	sw_recov2

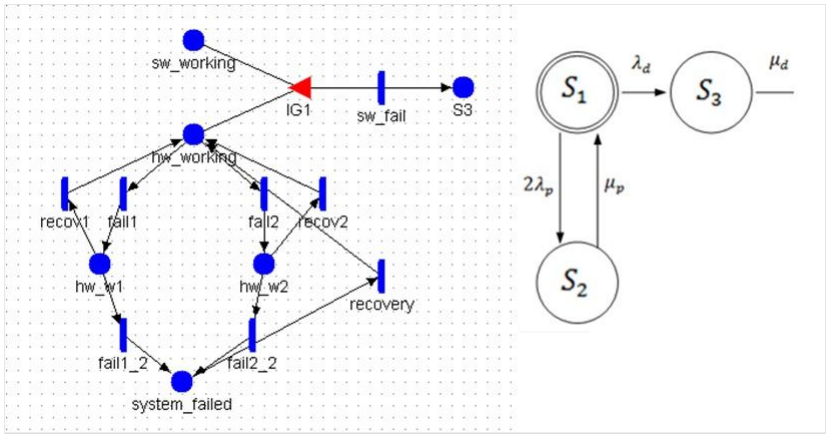

Figure 6. The atomic model of the initial fragment using Mobius SAN.

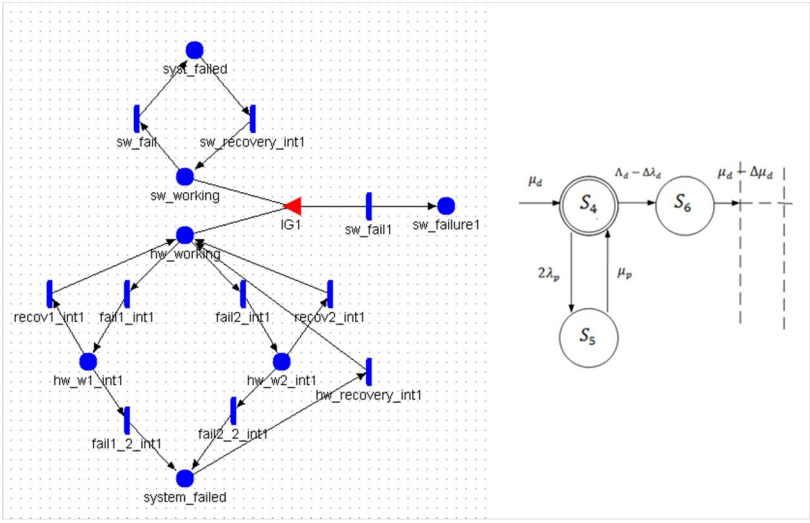

Figure 7. The atomic model showing the transition from the initial fragment to the first internal fragment.

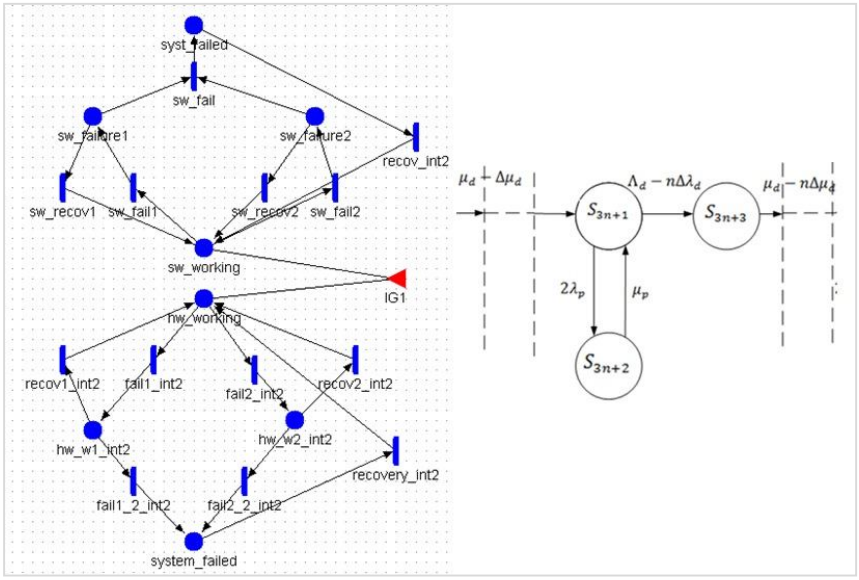

Figure 8. The atomic model with transitions between the internal fragments.

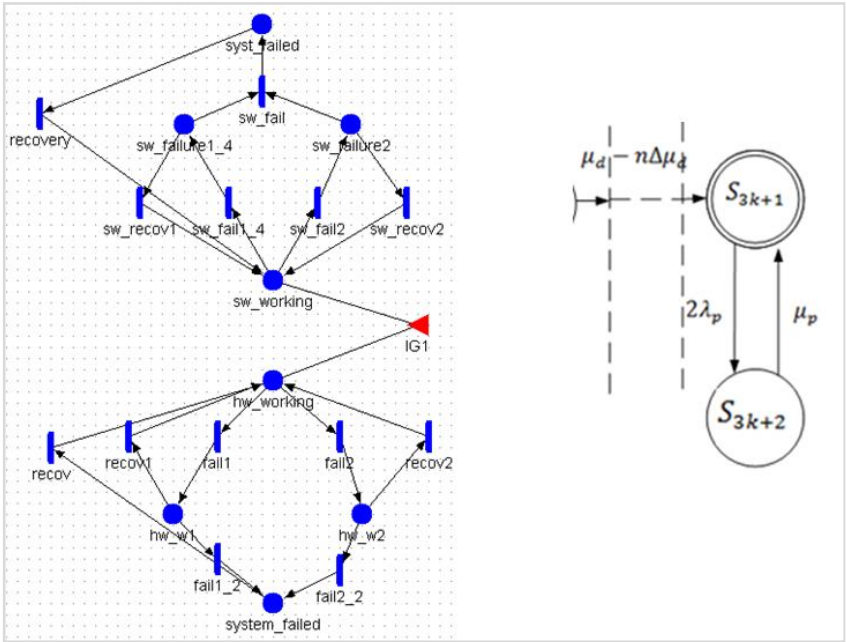

Figure 9. The atomic model with transitions between the last internal and the final fragment.

Figure 10 shows the system model using the SAN replication and joins. For further detail on the SAN syntax, the reader is encouraged to consult the SAN documentation.

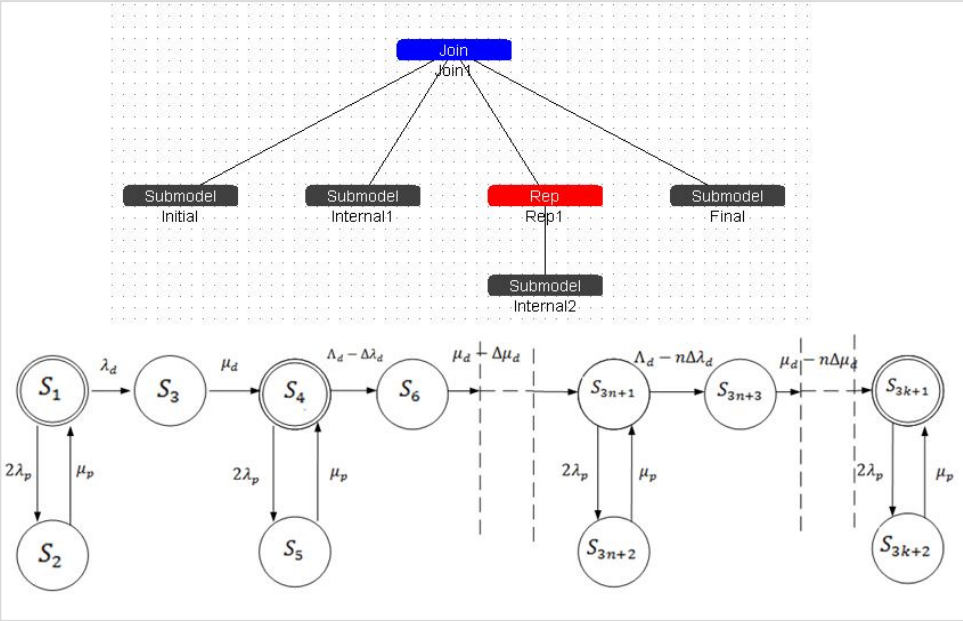

Figure 10. The system model, shown as a SAN compound model using REP and JOIN formalisms built in SAN.

System availability was computed via Monte Carlo simulation (simulation solver) with predefined confidence intervals. Figure 11 shows the results from the simulation solver with the respective confidence intervals against the reference availability obtained with the EXPETMETH utility.

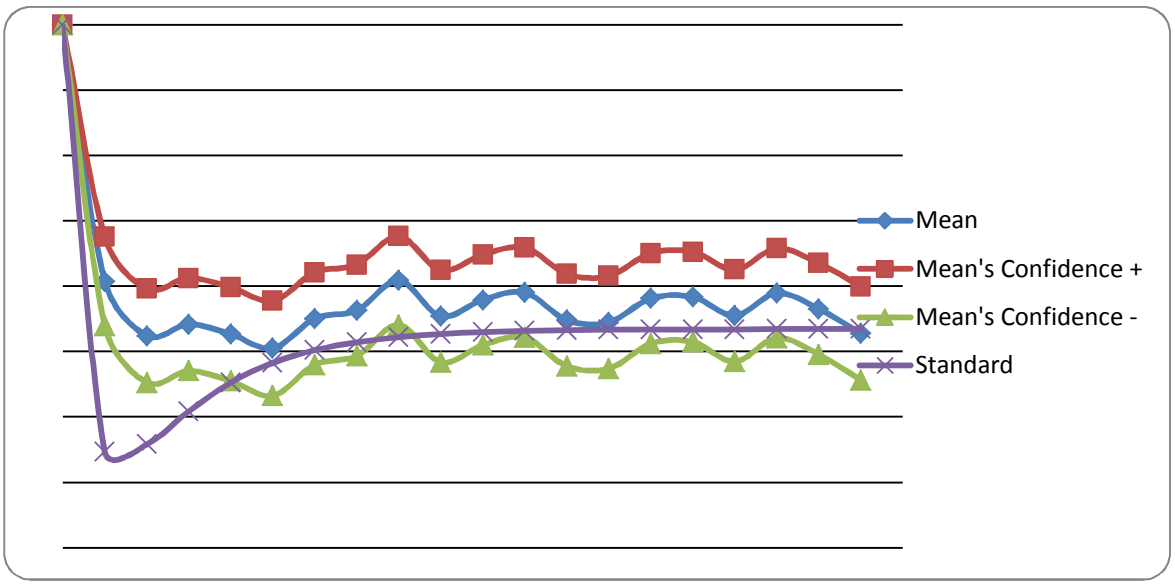

Figure 11. System availability calculated using SAN Mobius vs. EXPMETH.

The results obtained with all packages included in the comparison are shown in Figure 12.

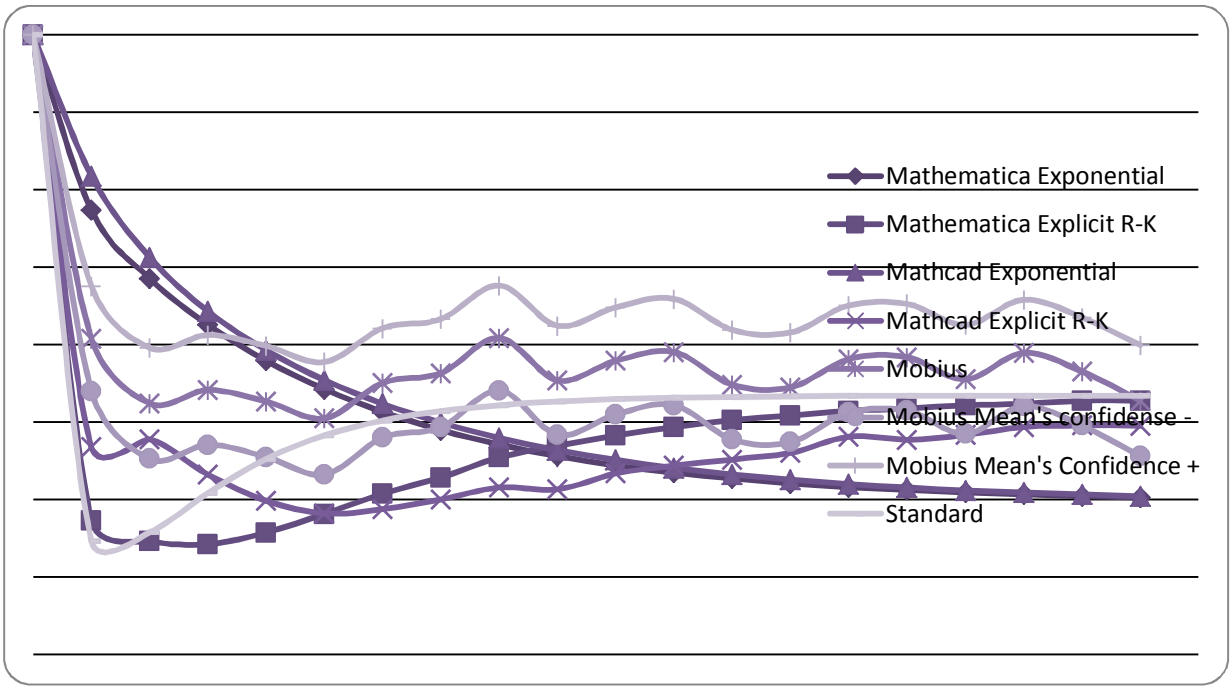

Figure 12. System availability function obtained with different tools/methods.

5. DISCUSSION

It is really striking how different the results obtained with different tools and methods are. The standard implementations built in the math packages for solving ordinary differential equations perform poorly: while the explicit Runge-Kutta method generally produce similar results (and generally follow those obtained with EXPMETH), the differences are non-negligible between Mathematica and Mathcad. The Rosenbrock method built in Maple to deal with stiff models seems to be an outlier – the availability it offers is too pessimistic.

Somewhat surprisingly, the exponential method implemented with the standard math packages offer results which are very different from those obtained with EXPMATH despite the fact that the algorithm is the same. One wonders what cause such a significant discrepancy.

The simulation solver seems to be close to the reference result obtained with EXPMETH – most of the time the EXPMETH computed availability is within the confidence intervals produced by the simulation solver.

The conclusions from this empirical comparison are alarming – the accuracy of the solutions offered by the popular math packages do not inspire high confidence! If one is to make a decision as to which of the results to trust one would really have very little to base their decision on. Clearly, further investigation is needed into which of the results should really be trusted.

6. RELATED RESEARCH

An extensive discussion of several problems related to obtaining accurate transient solution for Markov chains is presented in [7]. These authors note that stiffness is due to a large ratio of the model parameters but also, an observation made earlier by others, may be caused by the “mission time” if there exists a solution component whose variation is greater than 1/t, where t is the mission

time. They propose an extension to the standard methods for solving Kolmogorov equations such as Runge-Kutta and TR BDF2 method which offer *stability* despite the stiffness of the model to be solved.

Other relevant studies, [8] and [9], looked at the error of approximate methods of solving stiff Markov models using as a benchmark a M/M/1/k system, for which an exact solution is known.

In [10] and [11] the authors present an approximate method for solving stiff Markov models in which *aggregation* of the states is used: the states in the original model are divided into fast (i.e. those that have at least one fast outgoing transition), slow and fast recurrent (i.e. with fast outgoing and incoming transitions). The authors suggest that a good approximation of cumulative measures can be obtained when the original model is reduced to a model with slow states only and this derived model is solved. The authors experimentally evaluated the accuracy of the proposed method and report acceptable results for the cases when a non-stiff model is derived as a result of the proposed aggregation.

[12, 13] offer an empirical comparison of *uniformisation* methods of solving Markov chains. The authors point out that the standard uniformisation method proposed by Jensen performs poorly on stiff Markov models and concentrate on *Adaptive Uniformisation*. The problem is studied very extensively on complex contrived examples.

An interesting empirical study of the accuracy of scientific software (i.e. developed to process complex dataset from deep shelf oil exploration) was conducted by Less Hatton and Andy Roberts [14], which in nature is very similar to the study presented here. The authors conducted a thoroughly controlled experiment and compared the results from 15 independently developed very complex software packages, developed to the same specification. They report on the results obtained with 9 of the packages included in the study. All packages were subjected to the same large datasets collected from complex array of sensors. The results observed from the packages disagreed dramatically. The authors discussed why scientific software is of so poor quality.

7. CONCLUSIONS

The analysis of the results on system availability obtained with different off-the-shelf math packages allows us to draw the following conclusions:

- Each of the packages included in the study allows a modeler/assessor to evaluate system availability: each package either has a built in method which is appropriate for the task or allows one to build a routine (e.g. exponential method) to do so. The results obtained with the different packages, however, differ very significantly.
- The greatest discrepancy between the reference solution (obtained with EXPMATH) is observed with Maple 15 package when Rosenbrock's method for stiff Markov chains is used. The usage of the built-in function "dsolve" with 20 differential equations, leads to a large number of commands that are very similar in syntax, which greatly reduces the usability and seems to limit the scope for detecting and fixing faults. We scrutinized further the impact of model complexity on the accuracy of the solutions obtained with this method and observed that the accuracy deteriorates quickly with the increase of model complexity.
- The results obtained with Mathematical 8.0.1 and Mathcad 15 with the explicit Rung-Kutta method are closest to the reference solutions obtained with EXPMETH, which

is not surprising. We observed, however, that Mathematica 8.0.1 is sensitive to the initial period of operation: we observed a "drop" in availability at the beginning of the interval for which system's availability is computed.

- In terms of performance the packages performed as follows: Mathematica 8.0.1 took 01:16.9 seconds to compute the solution, Maple 15 – 07:44.3 seconds and Mathcad 15 – 00:25.34 seconds.
- The satisfactory solution was obtained using the simulation package Mobius.

In summary, if we are to rank the math packages included in the comparison, we would rank highest Mathematica 8.0.1 and Mathcad 15 (using the explicit Runge-Kutta solver) and the simulation solver of Mobius.

REFERENCES

[1] Volkov, L. (1981): Managing the operation of the aircraft systems: Tutorial. Moscow: Vyshaya Shkola. 368 p.

[2] Ventsel', E., Ovcharov, L. (2000): Probability theory and its applications in engineering. Moscow: Nauka. 480 p .

[3] Ventsel', E., Ovcharov, L. (1988): The theory of stochastic processes and its engineering applications, Moscow: Nauka. 384 p .

[4] Arushanyan, O., Zaletkin, S. (1990) : Numerical solution of ordinary differential equations using FORTRAN, Moscow: Moscow State University, 336 p.

[5] Littlewood, B., Popov, P. & Strigini, L. (2001): Modelling software design diversity - a review //ACM Computing Surveys. Vol. 33, №1. pp. 177 - 208.

[6] Popov, P. & Manno, G. (2011): The Effect of Correlated Failure Rates on Reliability of Continuous Time 1-Out-of-2 Software. in Computer Safety, Reliability, and Security (SAFECOMP 2011). Naples, Italy: Springer.

[7] Malthora, M., J.K. Muppala, & Trivedi K.S. (1994): Stiffness-Tolerant Methods for Transient Analysis of Stiff Markov Chains. Microelectronics Reliability. Vol. 34, № 11, pp. 1825-1841.

[8] Reibman, A., et al. (1989): Analysis of stiff Markov Chains. Operations Research Society of America. Vol. 1, № 2, pp. 126 - 133.

[9] Reibman, A. & Trivedi K.S. (1988): Numerical Transient Analysis of Markov models. Comput. Opns. Res. Vol. 15, № 1. pp. 19-36.

[10] Bobio, A. & Trivedi K.S. (1986): An Aggregation Technique for the Transient Analysis of Stiff Markov Chains // IEEE Transactions on Computers. Vol. C-35, № 9. pp. 803 - 814.

[11] Bobio, A. & Trivedi K.S. (1990): Computing Cumulative Measures of Stiff Markov Chains Using Aggregation // IEEE Transactions on Computers. Vol. 39, № 10. pp. 1291-1298.

[12] Diener, J.D. (1994): Empirical Comparison of Uniformisation Methods for Continuous Time Markov Chains, in Electrical and Computer Engineering // The University of Arizona. 97 p.

[13] Diener, J.D. & Sanders W.H. (1995): Empirical Comparison of Uniformisation Methods for Continuous-Time Markov Chains, in Computations with Markov Chains // W.J. Stewart, Editor. Kluwer Academic. pp. 547 - 570.

[14] Hatton, L. & Roberts A. (1994): How Accurate is scientific Software? // IEEE Transactions on Software Engineering. pp. 785 - 797.