An Experimental Study on Iterative Methods to Compute Transient Solutions of Large Markov Models

Antoine Rauzy
IML/CNRS
163, avenue de Luminy
13288 Marseille Cedex 09
FRANCE
arauzy@iml.univ-mrs.fr

Abstract: In this article, we report results of an experimental study on six iterative methods to compute the transient probabilities of large Markov models: full matrix exponentiation, forward Euler method, explicit Runge-Kutta methods of order 2 and 4 and Adams-Bashforth multi-steps methods of order 2 and 4. We suggest a simple but efficient implementation of these algorithms. We discuss how to tune their few parameters. We present experimental results that contradict the literature.

Keywords: Markov models, iterative algorithms to compute transient probabilities.

1. Introduction

Large Markov models naturally arise from reliability and dependability studies. By large, we mean models with several tens thousands of states and several hundreds thousands of transitions. For such models, an encoding of matrices by $n \times n$ arrays would exceed memory capacities of computers at hand. Fortunately, matrices issued of reliability models are in general sparse (they contain a lot of null entries). Therefore, it is possible to use much more economical encodings than $n \times n$ arrays.

Classical approaches to solve Markov models use operations on matrices, such as inversion or Gaussian elimination. These operations have in general non-sparse results, even when their arguments are sparse [Ste94]. Methods have been proposed in the literature that keep matrices sparse by performing only products of matrices by column vectors [Ste94,RT88]. These methods are called iterative for they compute the results by successive approximations.

In this article, we present results of an experimental study on six iterative methods to compute transient probabilities of large Markov models: full matrix exponentiation, forward Euler method, explicit Runge-Kutta methods of order 2 and 4 and Adams-Bashforth multi-steps methods of order 2 and 4.

Its contribution is as follows. First, we propose a very simple implementation for these methods. Second, we study how to tune their (few) parameters in order to achieve a good efficiency and a good accuracy. Third, we show that, conversely to what is commonly claimed in most of the textbooks, the forward Euler method is probably the best candidate for the assessment of large Markov models coming from reliability and dependability studies.

The remainder of this article is organized as follows. Section 2 introduces the problem. Section 3 describes algorithms under study and discusses their implementation. Section 4 shows how to assess various quantities (beyond transient probabilities) with minor modifications of these algorithms. Section 5 reports experimental results on artificial (however significant) examples. Finally, section 6 presents results on more realistic examples.

2. Position of the Problem

We assume that the reader is familiar with Markov terminology (for a good introduction, see for instance reference [Ste94]).

Consider a homogeneous, continuous time Markov chain with n states. Let Q be the $n \times n$ matrix whose elements $q_{i,j}$ denote the rate of transition of the chain from state i to state j. Let $\pi(t)$ be the vector of length n of probabilities to be in state i at time t. $\pi(0)$ thus denotes the vector of initial probabilities. It can be shown [Ste94] that,

$$\pi(t) \quad = \quad e^{t.Q}.\pi(0) \tag{1}$$

where $e^{t.Q}$ is defined as follows.

$$e^{t.Q} \quad = \quad \lim_{n\to\infty} \sum_{k=0}^{n} \frac{(t.Q)^k}{k!} \tag{2}$$

In the case of Markov chains, this series converges, at least theoretically. The problem is twofold. First, in order to assess $\pi(t)$ efficiently one should perform only products of matrices by vectors. Second, because of rounding errors, coefficients of matrices should be kept small (preferably between 0 and 1). To tackle this problem, two ideas can be applied [Ste94].

First, the following equality holds.

$$\frac{(t.Q)^k}{k!} \times \pi \quad = \quad \frac{t.Q}{k} \times \left(\frac{(t.Q)^{k-1}}{(k-1)!} \times \pi \right) \tag{3}$$

Therefore, the k-th term of the series (2) can be obtained from the $(k-1)$-th term.

Second, the following equality holds.

$$e^{t.Q} \times \pi \quad = \quad e^{dt.Q} \times \left(e^{(t-dt).Q} \times \pi \right) \tag{4}$$

Therefore, the computation of $e^{t.Q}.\pi(0)$ can be replaced successive computations of $\pi(t_i)=e^{dti.Q}.\pi(t_{i-1})$ such that $t_0=0$, $t_i=t_{i-1}+dt_i$, $dt_1+dt_2+\ldots=t$ and the dt_i's are small.

Combining these two ideas lead to several algorithms. These algorithms differ on the way they assess $e^{dt.Q}.\pi$. The choice of dt is also an important issue that we shall discuss in details in the next section.

3. Six Algorithms

3.1. The basic algorithmic scheme

Equalities (1) to (4) can be used to define the following algorithmic scheme.

```
select $\pi(0)$
$\pi \leftarrow \pi(0)$
$t \leftarrow 0$
while $t < T$ do                                  (5)
    select $dt$
    $t \leftarrow t + dt$
    $\pi \leftarrow e^{dt.Q}.\pi$
```

Where T denotes the mission time and Q stands for the infinitesimal general of the problem, i.e. the $n \times n$ matrix whose elements $q_{i,j}$ $(i \neq j)$ denote the rate of transition of the chain from state i to state j, and whose elements $q_{i,i}$ are defined as $q_{i,i}=1 - \sum_{j \neq i} q_{i,j}$.

To make the scheme (5) a concrete algorithm, it remains to answer the following questions.

- How to select $\pi(0)$? In general, $\pi(0)$ is given with the matrix Q. So, we won't discuss this point here.
- How to select dt? We shall discuss this major issue in this section.
- How to assess $e^{dt.Q}.\pi$? The different ways to assess this quantity define the different algorithms under study in this article.

If T is large enough, the stationary probabilities may be reached before the loop is completed. It is therefore a good idea to introduce a convergence test. This test aims to exit the loop as soon as two consecutive values of π can be considered as sufficiently close one another. In our implementation, this is achieved as follows. We select a threshold ε. The loop is exited if the following inequality holds (assuming $\pi_i(t+dt) \neq 0$).

$$\max_i \left(\frac{|\pi_i(t) - \pi_i(t+dt)|}{\pi_i(t+dt)} \right) \leq \varepsilon \tag{6}$$

where π_i denotes the i-th value in the vector π. Other norms can be defined to test the convergence [QSS00]. This one works fine. However, one of the surprise of the present study was that ε has to be chosen quite low (say 10^{-9}) in order not to stop too early.

3.2. Six Algorithms

Full matrix exponentiation: The first algorithm based on scheme (5) we can consider consists in computing successively the terms of the series (2) until the convergence criterion (6) is satisfied. This algorithm, so-called (full) matrix exponentiation, is denoted by EXP in sequel. It is often considered as too costly [Ste94]. This is the reason why approximations of $e^{dt.Q}.\pi$ have been suggested.

Forward Euler Method: The first approximation consists in remarking that, since dt has anyway to be small, the whole series can be approximated by its first two terms.

$$e^{dt.Q}.\pi \approx \pi + dt.Q.\pi \tag{7}$$

This method, so called forward Euler method in the literature, has a bad reputation. Most of the many textbooks we consulted claim that it is dubious because dt must be chosen very small to achieve a good accuracy. None of the experiments we performed confirms this claim. We denote this method by FEM in the sequel.

Explicit Runge-Kutta methods: Runge-Kutta methods are probably the most popular methods to solve ordinary differential equations [PTVF95,SAP97]. For the purpose of the present study, we consider the Runge-Kutta methods of order 2 and 4, denoted in the sequel by RK2 and RK4. RK2 is as follows (the reader may refer to references [Ste94], [PTVF95] and [SAP97] for a mathematical justification of RK2 and RK4).

$$e^{dt.Q}.\pi \approx \pi + \frac{1}{2}(\kappa_1 + \kappa_2) \tag{8}$$

with

$$\begin{aligned}\kappa_1 &= dt.Q.\pi \\ \kappa_2 &= dt.Q.(\pi + \kappa_1)\end{aligned}$$

RK4 is as follows.

$$e^{dt.Q}.\pi \approx \pi + \frac{1}{6}\left(\kappa_1 + 2\kappa_2 + 2\kappa_3 + \kappa_4\right) \tag{9}$$

with

$$\begin{aligned}\kappa_1 &= dt.Q.\pi \\ \kappa_2 &= dt.Q.(\pi + \frac{1}{2}\kappa_1) \\ \kappa_3 &= dt.Q.(\pi + \frac{1}{2}\kappa_2) \\ \kappa_4 &= dt.Q.(\pi + \kappa_3)\end{aligned}$$

Explicit Adams-Bashforth multi-steps methods: The underlying idea of multi-steps methods is orthogonal to the one of Runge-Kutta methods. The value of $\pi(t+dt)$ is computed from the values of $\pi(t)$, $\pi(t-dt)$, $\pi(t-2.dt) \ldots \pi(t-k.dt)$.

The Adams-Bashforth method of order 2 (AB2) is as follows.

$$\pi(t+dt) = \pi(t) + \frac{1}{2}\left(3.\rho(t) - \rho(t-dt)\right) \tag{10}$$

where $\rho(s)$ denotes $dt.Q.\pi(s)$.

The Adams-Bashforth method of order 4 (AB4) is as follows.

$$\pi(t+dt) = \pi(t) + \frac{1}{24}\left(55.\rho(t) - 59.\rho(t-dt) + 37.\rho(t-2.dt) - 9.\rho(t-3.dt)\right) \tag{11}$$

See references [Ste94], [QSS00] and [SAP97] for mathematical justifications.

3.3. Discussion

The methods presented above are said explicit for the value of $\pi(t+dt)$ is computed from the value of $\pi(t)$. In implicit methods, the value of $\pi(t+dt)$ is defined by a relation between $e^{dt.Q}.\pi(t)$ and $\pi(t)$. A system of equations has to be solved to get it. An iteration of an implicit method is therefore much more costly than one of an explicit method. Experiments we performed for this study showed that it is very dubious that, in practice, implicit methods can be more efficient than explicit ones.

The order of an explicit method characterizes the error it makes. A method is of order k if the error is in $O(dt^k)$. It can be shown [Ste94,SAP97] that the forward Euler method is of order 1, while Runge-Kutta methods presented above are respectively of order 2 and 4 (hence their names). It is often claimed that the estimation of the error is a major issue of the computation of transient solutions. However, according to reference [Ste94], methods proposed in the literature can estimate only the local error, i.e. the error made by one step of the algorithm. Just summing up the local errors is certainly a very rough approximation of the global error. No satisfactory method has been proposed to estimate the global error.

3.4. Implementation Issues

In order to implement the algorithms presented above, we need basically one operation: the product $Q.\pi$ of a sparse matrix Q by a column vector π. Therefore, the main issue is the choice of data structures to encode matrices and vectors. We suggest to encode vectors as arrays and matrices as lists (or arrays) of cells. Each cell represents a non-zero entry of the matrix. It contains three data: the row and column indices of the cell and its value. The diagonal elements of the matrix are not explicitly represented.

The following procedure computes $Q.\pi$ and adds the result to the vector σ.

```
$forall\ cell \in Q$
    $\sigma_{cell.row} \leftarrow \sigma_{cell.row} - cell.value \times \pi_{cell.row}$                (12)
    $\sigma_{cell.column} \leftarrow \sigma_{cell.column} + cell.value \times \pi_{cell.row}$
```

Note that we didn't initialize the vector σ. The above procedure computes actually $Q.\pi+\sigma$, which can be useful in some cases. Note also that the same idea can be applied to compute $Q^T.\pi$, where Q^T denotes as usual the transpose of Q.

```
$forall\ cell \in Q$                                                              (13)
    $\sigma_{cell.row} \leftarrow \sigma_{cell.row} + cell.value \times (\pi_{cell.column} - \pi_{cell.row})$
```

This kind of implementations has been already proposed, at least implicitly, in the literature (see e.g. [RT88]). It is worth noticing that the encoding we propose is compatible with other techniques that take advantage of the sparsity of the matrix (see e.g. [PG80b]).

3.5. The choice of dt

The algorithmic scheme (5) assumes that a new value is selected for *dt* at each iteration. This value depends indeed on Q and possibly on π. If *dt* depends only on Q, then it remains constant through the whole process. $dt.Q$ can be computed once for all, which saves a lot of multiplications. If *dt* depends also on π, then its value must be actually computed at each iteration. To be interesting, such a computation must save more time than it costs. We didn't see any heuristics that could justify this overhead. So all the experiments we present in this article were realized with constant *dt's*.

The choice of an appropriate *dt* is a major issue of the implementation of iterative methods. On the one hand, if *dt* is chosen too large, either the approximation of $e^{dt.Q}.\pi$ is too rough (for FEM, KR2, RK4, AB2 and AB4 methods), or the coefficient of the matrix are too large and rounding errors make the computation diverge. Other the other hand, if *dt* is chosen too small, the computation is very expensive. Moreover, as noticed in reference [SAP97], the impact of rounding errors may increase as the number of operations increases.

If all the coefficients of the matrix $dt.Q$ range between 0 and 1, the expectation that rounding errors cause problems is minored. Hence, the following rule of thumb can be applied: choose *dt* such the product $\rho = dt.max_i(q_{i,i})$ is in the range [0,1]. The product ρ provides a mean to choose *dt* in a uniform way:

$$dt = \frac{\rho}{\max_i q_{i,i}} \tag{14}$$

In the remainder of this article, we keep the same meaning for ρ. Note that equality (14) depends only on the matrix Q (therefore *dt* can be kept constant through the computation). It is worth noticing that, strictly speaking, to make the system solvable, the matrix $I+qt.Q$ must be stochastic, which in turn means that ρ must be smaller than 1. In practice however, values greater than *1* can sometimes be used, even with much care (see section 4). In Reference [PG80b], it is suggested to take $\rho=1$.

Once the value of *dt* (or equivalently of ρ) is selected, the value of $\pi(t)$ can be assessed using any of the six algorithms presented above. However, it is not possible to estimate the error with a single run. References [Ste94] and [PTVF95] suggest to perform a second calculation with a smaller value for *dt*, say *dt/2*. If the two results are not close enough, the process is reiterated. It is worth noticing that one of the arguments in favour of the Runge-Kutta methods is that they can come with some means to estimate the error [Ste94,QSS00,SAP97]. We found this argument of a little help in practice for rounding errors make this estimation too rough to be actually useful.

4. Assessment of Sojourn Times and other Quantities of Interest

The mean sojourn time $\sigma_i(T)$ in each state *i* during the mission time *T* is defined as follows.

$$\sigma_i(T) \;=\; \int_0^T \pi_i(t)\,dt \tag{15}$$

This integral can be assessed numerically while computing the transient probabilities as follows (using a trapezoid rule for the numerical integration).

$$\sigma(T) \;\approx\; \sum_{t=0}^{T}\left(\frac{\pi(t)+\pi(t+dt)}{2}\right)\times dt \tag{16}$$

The additional cost of this integration is very low (for it requires only a traversal of the vector at each iteration of the loop (5)).

Mean sojourn times can then used to compute many quantities of interest such as the mean production of an installation through a time period. These quantities are in general defined as the mathematical expectation of a given random variable *X* the value of which is defined in each state. To get the result, it suffices to sum over the states *i* the product $\sigma_i(T).X(i)$.

5. Experimental Results

This section is devoted to experimental results. These results were obtained on a laptop computer with 1 gigabyte memory and running windows 2000. Test cases used in this section are artificial ones, although they are designed to be close to realistic models. They are scalable, which makes them suitable to test hypotheses. Markov graphs under study in this article have been obtained by compiling high level descriptions written in the AltaRica language [Rau02].

5.1. Convergence date

A first problem is to determine the date of convergence, i.e. the date at which stationary probabilities are reached. This question is of importance in order to test the efficiency of algorithms (because of the convergence test). It is meaningful only if the availability of systems of repairable components is to be assessed. Otherwise, the graph shows sink states and the convergence date is virtually infinite.

A priori, the convergence date doesn't depend on the chosen method. In practice, this is only true if we consider its order and not its exact value. The convergence date depends on the tolerance criterion ε of equation (6). Experimentally, we found that 10^{-9} is a good trade-off for ε. A greater value leads to too early convergences and therefore to rough results. A lower value increases uselessly running times.

Test case 1: In order to study this first problem, we considered the availability of systems made of *n* independent components $C_1 \ldots C_n$, with failure rates λ_i and repair rates μ_i. Systems with dependent components don't show different behaviors with that respect.

The interesting experimental result is that the convergence date depends mainly on the lowest μ_i, i.e. the greatest MTTR. The smallest the μ_i, the highest the convergence date.

Consider a system made of $n=10$ identical components. Let μ be fixed and let λ/μ varies. Table 5.1.1 gives convergence dates (the graph has $2^n=1024$ states and $n.2^n=10240$ transitions).

These results show that the value of λ doesn't care (at least if we assume that λ has a realistic value w.r.t. μ) and that the convergence date is never greater than *20* times the greater MTTR.

Table 5.1.2 gives convergence dates for four series of systems with $n=l+h$ components. l-components are those with the smallest μ. In the first column, systems are made of n identical components. In other columns, $l=1$ and $h=n-1$. In the second column, $\mu_h = 2.\mu_l$. In the third and fourth ones $\mu_h = 25.\mu_l$.

These results illustrate our claim that only the smallest μ cares for the convergence date. The first three columns show very similar values while the number and the repair rates of their components (but the first one) differ dramatically. The fourth column shows values ten times greater than the others, because the smallest μ is ten times lower.

	λ/μ			
μ	10^{-1}	10^{-2}	10^{-3}	10^{-4}
10^{-1}	179	195	197	197
10^{-2}	1790	1950	1970	1970
10^{-3}	17900	19500	19700	19700

Table 5.1.1. Test case 1: convergence dates for system made of *n* identical components.

n	$l=n, h=0$ $\mu_l=2\ 10^{-2}$ $\lambda_l=3\ 10^{-5}$	$l=1, h=n-l$ $\mu_l=2\ 10^{-2}$ $\lambda_l=3\ 10^{-5}$ $\mu_h=4\ 10^{-2}$ $\lambda_h=6\ 10^{-5}$	$l=1, h=n-l$ $\mu_l=2\ 10^{-2}$ $\lambda_l=3\ 10^{-5}$ $\mu_h=5\ 10^{-1}$ $\lambda_h=7.5\ 10^{-3}$	$l=1, h=n-l$ $\mu_l=3\ 10^{-2}$ $\lambda_l=3\ 10^{-6}$ $\mu_h=5\ 10^{-4}$ $\lambda_h=7.5\ 10^{-6}$
2	750	816.667	855.769	8557.69
3	866.667	875	888.889	8888.89
4	900	910	907.143	9071.43
5	930	933.333	920.69	9206.9
6	950	950	930	9300
7	964.286	956.25	938.71	9387.1
8	968.75	966.667	943.75	9437.5
9	977.778	975	950	9500
10	985	981.818	954.412	9544.12
11	990.909	983.333	958.571	9585.71
12	991.667	988.462	962.5	9625
13	996.154	992.857	966.216	9662.16
14	1000	993.333	968.421	9684.21
15	1000	996.875	971.795	9717.95

Table 5.1.2. Test case 1: convergence dates for system made of *n*=*l*+*h* components.

5.2. *Which value for dt ?*

The choice of *dt* has been discussed section 3.5. Equation (14) provides a uniform way to select *dt*, through the ratio ρ. The "good" value for *dt* (or equivalently for ρ) depends on the method, the mission time and the structure of the problem. In practice, the choice of a value for *dt* must be made *a priori*. As suggested in reference [Ste94], the idea could be to start with a reasonable value ρ_0 for ρ, to perform the computation, and then to perform it again for $\rho_0/2$, $\rho_0/4$, … until results stabilize. The problem is therefore to select a good starting value ρ_0.

As a first experiment, consider again a system made of *10* independent repairable components. Assume, as previously, we are interested in the availability of the system, i.e. in the probability of each of the *1024* states. The table 5.2.1 gives the probability that all components are failed at $t=8760$, with $\lambda_i=10^{-4}$ and $\mu_i=10^{-2}$ for $i=1\ldots10$, for the six methods and different values of ρ (empty cells indicate values outside [0,1]).

We obtained the same picture with different number of component, different values of *t* and different values of the λ_i's and the μ_i's, including mixing very different values ($\lambda_i=10^{-8}$, $\mu_i=10^{-1}$, $i=1\ldots10$) .

This latter experiment is of great interest. A Markov model is stiff [RT88] if, in one or more states, out transitions rates are of widely varying magnitudes. This typically arises in reliability and dependability studies when repair rates are $\approx 10^6$ times failure rates. It is commonly admitted that stiff models are those where accuracy and stability problems are the most important.

ρ	FEM	RK2	RK4	AB2	AB4	EXP
1/8	$9.05287\ 10^{-21}$	$9.05287\ 10^{-21}$	$9.05287\ 10^{-21}$	$9.05287\ 10^{-21}$	$9.05287\ 10^{-21}$	$9.05287\ 10^{-21}$
1/4	$9.05287\ 10^{-21}$	$9.05287\ 10^{-21}$	$9.05287\ 10^{-21}$	$9.05287\ 10^{-21}$	$9.05287\ 10^{-21}$	$9.05287\ 10^{-21}$
1/2	$9.05287\ 10^{-21}$	$9.05287\ 10^{-21}$	$9.05287\ 10^{-21}$	$9.05287\ 10^{-21}$		$9.05287\ 10^{-21}$
1	$9.05287\ 10^{-21}$	$9.05287\ 10^{-21}$	$9.05287\ 10^{-21}$	$9.05287\ 10^{-21}$		$9.05287\ 10^{-21}$
2	$5.29341\ 10^{-17}$	$5.76799\ 10^{-17}$	$9.05287\ 10^{-21}$			$9.05287\ 10^{-21}$
4						$9.05287\ 10^{-21}$
8						$9.05287\ 10^{-21}$
16						$9.05287\ 10^{-21}$
32						$9.05287\ 10^{-21}$
64						

Table 5.2.1. Test case 1 with $n=10$: probability that all components are failed at $t=8760$.

The following observations can be made from the above experiment.

- Multi-step methods AB2 and AB4 perform rather weak. They more costly and more instable than FEM. They must be avoided.
- FEM is as stable as RK2 and RK4, even on stiff models.
- $\rho=1$ is "the" good value to start with for FEM, RK2 and RK4 (i.e. that largest value of *dt* that ensures that no element in the probability matrix is outside [0,1]). Larger values lead to instability and we didn't observed realistic problems form which $\rho=1$ leads to instability.
- EXP has a great auto-corrective capacity, thanks to its inner loop.
- All methods show a very sharp phenomenon: up to a certain value of ρ, results are precise. Then, for a value only a bit larger, the instability takes place and results go outside of [0,1]. For stiff problems, the phenomenon is even sharper.

Test case 2: In order to confirm the above observations, we considered the family of the networks pictured figure 5.2.2. The source *I* and the target *O* are assumed to be perfectly reliable. *I* supplies an output flow. Components *Ui*'s are repairable units with a failure rate λ_{Ui} and a repair rate μ_{Ui}. They supply an output flow if they are working and they are supplied with an input flow. Components *Si*'s are spare units. They are started when the corresponding *Di* is unable to supply an output flow. They are stopped as soon as *Di* is able again to supply it. Components *Si*'s are characterized by their probability γ_{Si} to fail on demand, their failure rate λ_{Si} and their repair rate μ_{Si}. Therefore, the state of *Si* depends on the states of *U1*, *S1*, …, *Ui-1*, *Si-1*. Finally, at most *r* components can be repaired simultaneously, which makes all components pairwisely dependent. The structure of this test case is thus very different from the structure of the previous one.

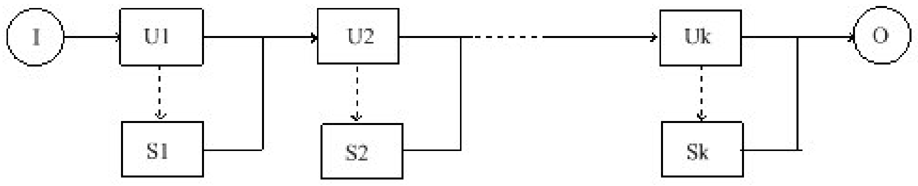

Figure 5.2.2. A network of regular and spare units.

ρ	FEM	RK2	RK4	AB2	AB4	EXP
1/8	*1.02948 10^{-7}*	*1.02948 10^{-7}*	*1.02948 10^{-7}*	*1.02948 10^{-7}*	*1.02948 10^{-7}*	*1.02948 10^{-7}*
1/4	*1.02948 10^{-7}*	*1.02948 10^{-7}*	*1.02948 10^{-7}*	*1.02948 10^{-7}*	*1.02948 10^{-7}*	*1.02948 10^{-7}*
1/2	*1.02948 10^{-7}*	*1.02948 10^{-7}*	*1.02948 10^{-7}*	*1.02948 10^{-7}*		*1.02948 10^{-7}*
1	*1.02948 10^{-7}*	*1.02948 10^{-7}*	*1.02948 10^{-7}*	*1.02948 10^{-7}*		*1.02948 10^{-7}*
2	*1.02948 10^{-7}*	*1.02948 10^{-7}*	*1.02948 10^{-7}*			*1.02948 10^{-7}*
4						*1.02948 10^{-7}*
8						*1.02948 10^{-7}*
16						*1.02948 10^{-7}*
32						*1.02948 10^{-7}*
64						

Table 5.2.3. Test case 2 with $n=5$: probability that all components are failed at $t=8760$.

ρ	FEM	RK2	RK4	AB2	AB4	EXP
1/16	*9.82868 10^{-5}*	*9.82868 10^{-5}*	*9.82868 10^{-5}*	*9.83113 10^{-5}*	*9.83113 10^{-5}*	*9.82865 10^{-5}*
1/8	*9.82903 10^{-5}*	*9.82903 10^{-5}*	*9.82903 10^{-5}*	*9.83393 10^{-5}*	*9.83393 10^{-5}*	*9.829 10^{-5}*
1/4	*9.82973 10^{-5}*	*9.82973 10^{-5}*	*9.82973 10^{-5}*	*9.83952 10^{-5}*	*9.83952 10^{-5}*	*9.8297 10^{-5}*
1/2	*9.83113 10^{-5}*	*9.83113 10^{-5}*	*9.83113 10^{-5}*	*9.85072 10^{-5}*		*9.8311 10^{-5}*
1	*9.83393 10^{-5}*	*9.83393 10^{-5}*	*9.83393 10^{-5}*	*9.87311 10^{-5}*		*9.8311 10^{-5}*
2	*9.83952 10^{-5}*	*9.83952 10^{-5}*	*9.83952 10^{-5}*			*9.8311 10^{-5}*
4						*9.8311 10^{-5}*
8						*9.8535 10^{-5}*
16						*9.8535 10^{-5}*
32						*9.8535 10^{-5}*
64						

Table 5.2.4. Test case 2 with $n=5$: reliability of the system at $t=8760$.

Table 5.2.3 gives the probability that all components are failed, computed with the different methods, different values of ρ and for $n=5$, $r=2$, $t=8760$, $\lambda_{Ui} = 10^{-7}$, $\mu_{Ui} = 10^{-1}$, $\gamma_{Si} = 0.1$, $\lambda_{Si} = 10^{-4}$ and $\mu_{si} = 10^{-2}$, for all $i=1...5$. The corresponding graph has *1507* states and *8992* transitions. Table 5.2.4 gives the reliability of the system.

Tables 5.2.3 and 5.2.4 show the same picture as table 5.2.1, although the problems and the numerical values are very different! We observed the same phenomenon for different values of the transition rates, different values of the mission time…

These results show that, at least on examples close to those one encounters in reliability and dependability studies, all of the methods are accurate if *dt* is selected correctly. This observation contradicts many textbooks that consider FEM as too unstable to be used. The ratio r gives a convenient mean to select *dt*.

5.3. Which method to choose?

How do methods compare in term of efficiency? A first answer to this question is provided by tables 5.3.1, 5.3.2 and 5.3.3 that give the running times in seconds of the different methods for different values of ρ. Two observations can be made about these running times.

- FEM performs always better than the other methods. For the same *dt*, it is twice as fast as RK2 and four times faster than RK4. This fact means that a fair comparison of accuracies of FEM and RK2 (resp. RK4) should be made with a *dt* two times larger (resp. four times larger). Anyway, we observed a better accuracy of Runge-Kutta methods than in FEM in none of the test cases with dealt with.
- Running times for EXP decrease as *dt* increases up to a point were they start to increase again. In other words, the inner loop of EXP is able to correct a too large value of *dt*, but this is costly. Moreover, FEM with $\rho=1$ is always faster than EXP with any ρ. For the same accuracy, FEM is always faster than EXP (and more generally than any other method).

5.4. Where are the limits?

The previous experiments show that FEM, RK2, RK4 and EXP methods are efficient and accurate. An important question is how far we can go with these methods on nowadays computers. This question is twofold: we have to consider both space and time consumption. In our implementation, that uses double precision arithmetic, each non-zero entry of the matrix requires 28 bytes (the same structure is used to encode both the infinitesimal generator and the probability matrix) and each entry of a vector requires 8 bytes. The numbers of vectors required to run the different methods are as follows.

FEM	EXP	RK2	RK4	AB2	AB4
2	5	4	4	4	6

Let *N* be the number of states and $M = r.N$ be the number of non-zero entries of the matrix. The number of bytes required is thus $(28r+16).N$ for FEM, $(28r + 40).N$ for EXP, ... To make things concrete, assume that $N = 10^5$. If $r = 5$, i.e. $M = 5.10^5$, the execution of EXP requires about *16* megabytes. If $r=10$, it requires *32* megabytes, and so on. On a computer with 1 gigabyte memory, it should be theoretically possible to deal with Markov graphs with up to 10^6 states and 10^7 transitions. In practice, actual limits are slightly below these values.

As an illustration, consider again the test case 1. The largest number of components we were able to handle is *18*. The corresponding graph has *262,144* states and *4,718,592* transitions.

The table 5.4.1 gives the probability that all components are failed computed at $t=8760$ with the different methods for different values of ρ. Running times in seconds are given as well.

The largest number of blocs we were able to deal with for test case 2 is 7 for both assessment of the availability and the assessment of the reliability. The corresponding graphs have respectively *264,711* states and *2,525,164* transitions and *203,997* states and *1,187,857* transitions. Tables 5.4.2 and 5.4.3 give the results for this test case.

These results show that very large graphs can be handled within few minutes on a nowadays laptop computer.

As a concluding remark, we observe that the generation of Markov graphs from high level description is often more space consuming than their assessment. The limits we gave above are actually those of our compiler (from AltaRica descriptions to Markov graphs), not those of our assessment tool.

6. More realistic examples

6.1. A Power Supply Unit

Test case 3: this test case comes from reference [PG80a] and a personal communication by M. Bouissou [Bou03]. Fig. 6.1.1 pictures an electric power supply. The regular power supply of boards LHA and LHB comes the transformer TS. TS itself is supplied by the plant PLT. PLT works in two modes, either the regular mode when the net NET is available or a standalone mode, rather unstable, when the net is lost. When PLT is failed or when the transformer TP is down, TS is supplied by the net (through the line GEV). When TS cannot be alimented and the net is available, boards are alimented by the transformer TA, through the line LGR. Finally, diesel engines DA and DB are used as cold spare units when neither TS nor TA is able to supply boards. Boards LGD and LGF may fail.

ρ	FEM	RK2	RK4	AB2	AB4	EXP
1/16	*0.42*	*0.861*	*1.702*	*0.46*	*0.51*	*1.692*
1/8	*0.22*	*0.44*	*0.881*	*0.23*	*0.25*	*0.951*
1/4	*0.12*	*0.23*	*0.44*	*0.12*	*0.13*	*0.52*
1/2	*0.06*	*0.12*	*0.24*	*0.06*		*0.3*
1	*0.03*	*0.06*	*0.13*	*0.03*		*0.19*
2			*0.06*			*0.11*
4						*0.09*
8						*0.1*
16						*0.11*
32						*0.13*

Table 5.3.1. Test case 1 with *n=10* (availability): running times in seconds.

ρ	FEM	RK2	RK4	AB2	AB4	EXP
1/16	*0.781*	*1.572*	*3.004*	*0.841*	*0.941*	*3.094*
1/8	*0.37*	*0.781*	*1.562*	*0.43*	*0.49*	*1.732*
1/4	*0.19*	*0.41*	*0.791*	*0.21*	*0.25*	*0.971*
1/2	*0.1*	*0.21*	*0.42*	*0.12*		*0.54*
1	*0.05*	*0.11*	*0.21*	*0.25*		*0.31*
2			*0.12*			*0.19*
4						*0.16*
8						*0.18*
16						*0.2*
32						*0.23*

Table 5.3.2. Test case 2 with *n=5* (availability): running times in seconds.

ρ	FEM	RK2	RK4	AB2	AB4	EXP
1/16	*1.722*	*3.495*	*6.589*	*1.932*	*2.313*	*6.168*
1/8	*0.841*	*1.702*	*3.304*	*0.971*	*1.151*	*3.164*
1/4	*0.43*	*0.851*	*1.642*	*0.48*	*0.58*	*1.672*
1/2	*0.21*	*0.42*	*0.811*	*0.25*		*0.841*
1	*0.11*	*0.21*	*0.41*	*0.12*		*0.44*
2	*0.06*	*0.11*	*0.21*			*0.24*
4						*0.3*
8						*0.34*
16						*0.36*
32						*0.39*

Table 5.3.3. Test case 2 with *n=5* (reliability): running times in seconds.

	FEM		EXP	
ρ	probability	running time	probability	Running time
1/2	*8.36017* 10^{-37}	*165*	*8.36017* 10^{-37}	*752*
1	*8.36017* 10^{-37}	*84*	*8.36017* 10^{-37}	*442*
2			*8.36017* 10^{-37}	*267*
4			*8.36017* 10^{-37}	*217*
8			*8.36017* 10^{-37}	*258*
16			*8.36017* 10^{-37}	*297*

Table 5.4.1. Results for test case 1 (availability) with *n=18*.

	FEM		EXP	
ρ	probability	running time	probability	running time
1/2	*1.02959 10^{-7}*	*135*	*1.02959 10^{-7}*	*655*
1	*1.02959 10^{-7}*	*68*	*1.02959 10^{-7}*	*380*
2			*1.02959 10^{-7}*	*227*
4			*1.02959 10^{-7}*	*211*
8			*1.02959 10^{-7}*	*249*
16			*1.02959 10^{-7}*	*275*

Table 5.4.2. Results for test case 2 (availability) with $n=7$.

	FEM		EXP	
ρ	probability	running time	probability	running time
1/8	*9.83004 10^{-5}*	*1037*	*9.83 10^{-5}*	*3138*
1/4	*9.83074 10^{-5}*	*510*	*9.83 10^{-5}*	*1652*
1/2	*9.83214 10^{-5}*	*257*	*9.8314 10^{-5}*	*852*
1	*9.83493 10^{-5}*	*132*	*9.83419 10^{-5}*	*461*
2			*9.83979 10^{-5}*	*275*
4			*9.83979 10^{-5}*	*363*
8			*9.83979 10^{-5}*	*411*
16			*9.83979 10^{-5}*	*443*

Table 5.4.3. Results for test case 2 (reliability) with $n=7$.

All components are repairable with a failure rate λ and a repair rate μ. Plant PLT has a probability γ to fail on demand to switch in standalone mode. Finally, Diesel DA and DB are spare units with a probability γ to fail on demand and a repair rate μ_d that corresponds to this failure. Moreover, DA and DB have a common cause failure with a rate λ_{cc} and a repair rate μ_{cc}. These rates and probabilities are given Table 6.1.2.

The problem is to assess the availability and the reliability of the system for a given mission time.

The Markov graphs for the availability has *30,720* states and *459,560* transitions. The graph for reliability has *25,337* states and *152,679* transitions.

Table 6.1.3 gives results obtained on test case 3 with the different methods and different values of the ratio ρ.

We can make the following remarks from these experiments.

- This rather large realistic test case is solved within few minutes with a good accuracy by the methods under study in this article.
- We can observe the same behaviour of the methods for this realistic test case than for artificial examples of the previous section.

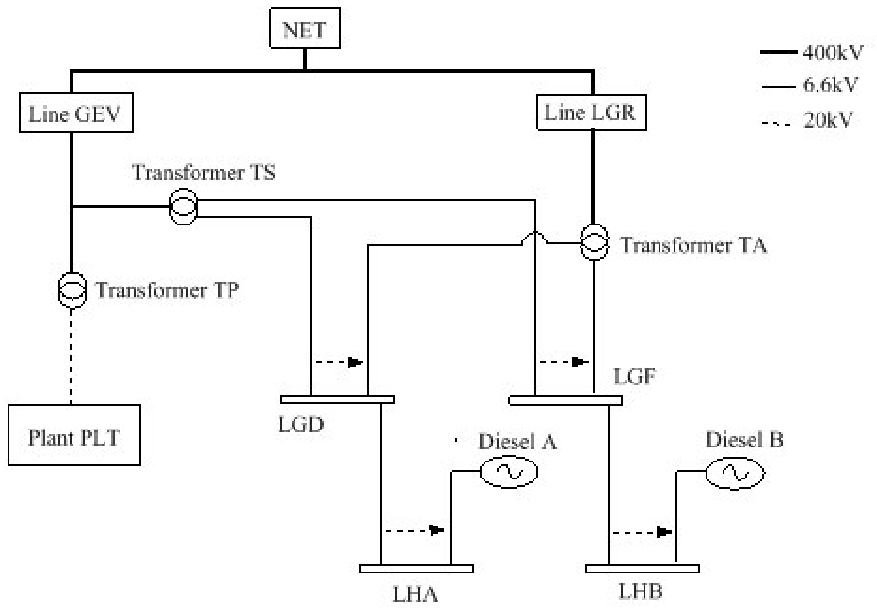

Figure 6.1.1. Test case 3. An electric power supply.

	λ	μ	γ	μ_d	λ_{cc}	μ_{cc}
NET	$1.0\ 10^{-6}$	$8.0\ 10^{-3}$				
GEV, LGR	$5.0\ 10^{-6}$	$1.0\ 10^{-2}$				
TP, TS, TA	$2.0\ 10^{-6}$	$1.0\ 10^{-3}$				
LGD, LGF, LHA, LHB	$2.0\ 10^{-7}$	$1.0\ 10^{-1}$				
PLT (regular mode)	$1.0\ 10^{-4}$	$1.0\ 10^{-1}$				
PLT (standalone)	$1.0\ 10^{-1}$	$1.0\ 10^{-3}$	0.5			
DA, DB	$1.0\ 10^{-4}$	$2.0\ 10^{-2}$	0.001	$1.0\ 10^{-1}$	$1.0\ 10^{-4}$	$1.0\ 10^{-2}$

Table 6.1.2. Test case 3: reliability parameters.

		(un)availability		(un)reliabilility	
method	ρ	value	running time	value	running time
FEM	*1/8*	*3.25201 10^{-7}*	*849*	*5.1138 10^{-5}*	*231*
FEM	*1/4*	*3.25201 10^{-7}*	*428*	*5.11394 10^{-5}*	*116*
FEM	*1/2*	*3.25201 10^{-7}*	*215*	*5.11422 10^{-5}*	*58.044*
FEM	*1*	*3.25201 10^{-7}*	*107*	*5.11478 10^{-5}*	*29.362*
FEM	*2*	*3.25201 10^{-7}*	*53.467*	*5.1159 10^{-5}*	*14.371*
RK2	*1/8*	*3.25201 10^{-7}*	*1800*	*5.1138 10^{-5}*	*530*
RK2	*1/4*	*3.25201 10^{-7}*	*897*	*5.11394 10^{-5}*	*266*
RK2	*1/2*	*3.25201 10^{-7}*	*451*	*5.11422 10^{-5}*	*134*
RK2	*1*	*3.25201 10^{-7}*	*228*	*5.11478 10^{-5}*	*68*
RK2	*2*	*3.25201 10^{-7}*	*117*	*5.1159 10^{-5}*	*34.77*
RK4	*1/8*	*3.25201 10^{-7}*	*3642*	*5.1138 10^{-5}*	*1053*
RK4	*1/4*	*3.25201 10^{-7}*	*1809*	*5.11394 10^{-5}*	*522*
RK4	*1/2*	*3.25201 10^{-7}*	*897*	*5.11422 10^{-5}*	*257*
RK4	*1*	*3.25201 10^{-7}*	*443*	*5.11478 10^{-5}*	*128*
RK4	*2*	*3.25201 10^{-7}*	*222*	*5.1159 10^{-5}*	*65*
EXP	*1/8*	*3.25201 10^{-7}*	*2534*	*5.11369 10^{-5}*	*857*
EXP	*1/4*	*3.25201 10^{-7}*	*1383*	*5.11383 10^{-5}*	*468*
EXP	*1/2*	*3.25201 10^{-7}*	*750*	*5.11383 10^{-5}*	*252*
EXP	*1*	*3.25201 10^{-7}*	*395*	*5.11439 10^{-5}*	*129*
EXP	*2*	*3.25201 10^{-7}*	*205*	*5.11551 10^{-5}*	*68*
EXP	*4*	*3.25201 10^{-7}*	*213*	*5.11775 10^{-5}*	*68*
EXP	*8*	*3.25201 10^{-7}*	*261*	*5.11775 10^{-5}*	*81*
EXP	*16*	*3.25201 10^{-7}*	*286*	*5.11775 10^{-5}*	*89*
EXP	*32*	*3.25201 10^{-7}*	*300*	*5.13567 10^{-5}*	*93*

Table 6.1.3. Test case 3: results and running times in seconds.

6.2. A Part of an Oil Production System

<u>Test case 4:</u> The system pictured Fig. 6.2.1 represents a part of an oil extraction installation. This test case has been designed to concentrate most of the modelling difficulties of the assessment of production availability (see reference [KR02] for a presentation of these topics).

- *W1* and *W2* are wells. Their production capacities are respectively *160* and *70* (10^k barrels/day). When they are failed, the production is stopped.
- *T1* and *T2* are tanks. They are assumed to be perfectly reliable. Their storage capacities are respectively *110* and *100* (barrels/day).
- *A*, *B* are two treatment units. They have two failure modes: a failure that decreases their capacities from *170* to *100* (resp. *120* to *70*), and a severe failure that stops the treatment. A severe failure may occur either when the unit is working correctly or when it is in a degraded mode.
- Components *Ci*'s, *Di*'s and *Ei*'s are treatment units. Their capacities are respectively *120*, *80* and *50*. For all of these units, a failure stops the treatment. Components of bloc *E* are in hot redundancy.
- The line *D* is in cold redundancy with the line *C*. As soon as the line *C* is repaired, the line *D* is stopped. If *C1* fails, then *C2* is stopped and vice-versa.
- *R* is a pool of two repairers (i.e. that at most two components can be repaired simultaneously). A component is not able to treat the production during a repair.
- The production entering in component *A* must be split into two: a fraction goes to the tank *T1* through the top line, the remainder goes to the tank *T2* through the bottom line. This requires defining a splitting policy. We adopt the following one. The production of *W1* goes preferably to the top line. The production of *W2* goes preferably to the bottom line. If needed, what remains available from *W1* goes to the bottom line.

The table 6.2.1. gives failure rates λ (λ_s for severe failures of components *A* and *B*), repair rates μ (μ_s for repairs of severe failures), and probability to fail on demand γ for components *D1* and *D2*.

The problem is to assess the average production of the two wells and the average storage in the two tanks, i.e. the mathematical expectation of these quantities through the mission time. These quantities can be assessed by means of sojourn times, as explained section 4.

The Markov graph that encodes this test case has *110,768* states and *861,232* transitions. The table 6.2.3 gives results obtained with the different methods for different values of ρ.

We can make the same remarks for this test case as for the previous one: first, it is handled within few minutes although very large. Second, methods show once again the same behaviour as on artificial examples of the previous section.

7. Conclusion

In this article, we reported results of an experimental study on six iterative methods to compute transient probabilities of large Markov models. We suggest a very simple, however very efficient, implementation of these methods (that take less than 100 lines of C code). We discussed how to tune their parameters. We show, by means of several test cases, that methods under study give consistently accurate results. Markov graphs with up to hundreds of thousands states and millions of transitions can be handled within few minutes on a nowadays laptop computer.

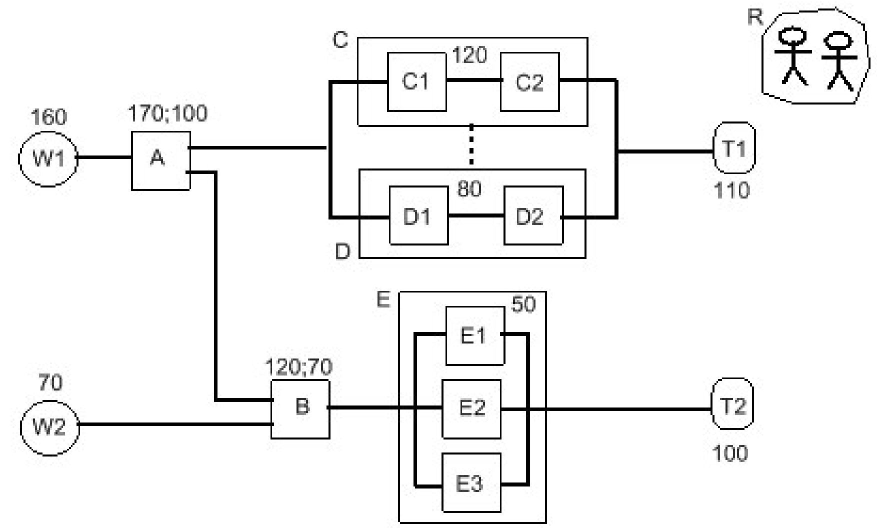

Figure 6.2.1. An Oil Production System.

	λ	μ	λ_s	μ_s	γ
W1, W2	*2 10^{-5}*	*5 10^{-3}*			
A, B	*9 10^{-3}*	*2 10^{-2}*	*3 10^{-5}*	*4 10^{-3}*	
C1, C2	*3 10^{-3}*	*4 10^{-2}*			
D1, D2	*8 10^{-3}*	*4 10^{-2}*			*2 10^{-2}*
E1, E2, E3	*4 10^{-3}*	*5 10^{-2}*			

Table 6.2.2. Transition rates for the system depicted Fig. 6.2.1.

method	ρ	production of well W1	production of well W2	storage of tank T1	storage of tank T2	Running time
FEM	*1/8*	*83.5621*	*47.6059*	*70.0157*	*61.1523*	*264*
FEM	*1/4*	*83.5677*	*47.6095*	*70.0207*	*61.1565*	*139*
FEM	*1/2*	*83.5788*	*47.6168*	*70.0308*	*61.1648*	*69*
FEM	*1*	*83.601*	*47.6315*	*70.0509*	*61.1816*	*34.58*
RK2	*1/8*	*83.565*	*47.607*	*70.0177*	*61.1543*	*568*
RK2	*1/4*	*83.5736*	*47.6119*	*70.0249*	*61.1605*	*294*
RK2	*1/2*	*83.5909*	*47.6216*	*70.0393*	*61.1732*	*151*
RK2	*1*	*83.6266*	*47.6416*	*70.0689*	*61.1992*	*77*
RK4	*1/8*	*83.565*	*47.607*	*70.0177*	*61.1542*	*1112*
RK4	*1/4*	*83.5734*	*47.6118*	*70.0248*	*61.1604*	*568*
RK4	*1/4*	*83.5734*	*47.6118*	*70.0248*	*61.1604*	*574*
RK4	*1/2*	*83.5904*	*47.6215*	*70.039*	*61.1729*	*297*
RK4	*1*	*83.6248*	*47.6409*	*70.0677*	*61.198*	*154*
RK4	*2*	*80.6009*	*46.239*	*67.7647*	*59.0752*	*132*
EXP	*1/8*	*83.565*	*47.607*	*70.0177*	*61.1542*	*966*
EXP	*1/4*	*83.5734*	*47.6118*	*70.0248*	*61.1604*	*547*
EXP	*1/2*	*83.5904*	*47.6215*	*70.039*	*61.1729*	*308*
EXP	*1*	*83.6248*	*47.6409*	*70.0677*	*61.198*	*175*
EXP	*2*	*83.6946*	*47.6803*	*71/89*	*61.249*	*115*
EXP	*4*	*83.8392*	*47.7607*	*70.2456*	*61.3543*	*133*
EXP	*8*	*84.1458*	*47.9283*	*70.497*	*61.577*	*159*
EXP	*16*	*84.8085*	*48.2828*	*71.035*	*62.0564*	*174*

Table 6.2.3. Test case 4: results and running times in seconds.

The most surprising result of our study is certainly that the very simple Forward Euler Method, which is considered as too rough in the literature, gives accurate results and over performed the other algorithms (including Runge-Kutta methods).

The limits in the size of models we are able to deal with stand in their generation, not their assessment. Efficient generation of Markov graphs from high level descriptions (including truncation, state merging and symmetry breaking techniques) is certainly the focus point for future improvements of the Markov technology.

8. Bibliography

[Bou03] M. Bouissou, Compte-rendu la journée de démonstration d'outils d'analyse de sûreté de fonctionnement des systèmes. 2003. http://www-math.univ-mlv.fr/users/bouissou.

[KR02] Y. Kawauchi and M. Rausand, A new approach to production regularity assessment in the oil and chemical industries. *Reliability Engineering and System Safety*,.n°75 pp 379–388, 2002.

[PG80a] A. Pagès and M. Gondrand. Fiabilité des systèmes, volume 39 of Collection de la Direction des Etudes et Recherches d'EdF. Eyrolles, 1980. ISSN 0399-4198.

[PG80b] I. A. Papazoglou and E. P. Gyftopoulos. Markovian Reliability Analysis Under Uncertainty with an Application on Shutdown System of Clinch River Reactor. *Nuclear Science and Engineering*. vol. 73, pp 1-18. 1980.

[PTVF95] W. H. Press, S. A. Teukolski, W. T. Vettering and B. P. Flannery, *Numerical Recipes in C*. Cambridge University Press, 1995. ISBN 0-521-43108-5.

[QSS00] A. Quateroni, R. Sacco and F. Saleri. *Méthodes numériques pour le calcul scientifique*. Springer Verlag, 2000. ISBN 2-287-59701.

[Rau02] A. Rauzy. "Mode automata and their compilation into fault trees". *Reliability Engineering and System Safety*. Vol. 78, pages 1-12. 2002. Elsevier Publishing.

[RT88] A. Reibman and K. Trivedi. "Numerical transient analysis of Markov models". *Computers And Operation Research*. vol 15, num 1, 1998. pp 19-36.

[SAP97] L.F. Shampine, R.C. Allen and Jr. S. Pruess. Fundamentals of Numerical Computing. John Wiley & Sons. 1997. ISBN 0-471-16363-5.

[Ste94] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1994. ISBN 0-691-03699-3.