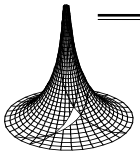

9/1 (2011), 45–75
tmcs@math.klte.hu
http://tmcs.math.klte.hu

Teaching
Mathematics and
Computer Science

Teaching Reliability Theory with the Computer Algebra System MAXIMA

ATTILA CSENKI

Abstract. The use of the Computer Algebra System MAXIMA as a teaching aid in an MSc module in Reliability Theory is described here. Extracts from student handouts are used to show how the ideas in Reliability Theory are developed and how they are intertwined with their applications implemented in MAXIMA. Three themes from the lectures are used to illustrate this: (1) Normal Approximations, (2) Markov Modelling, (3) Laplace Transform Techniques.

It is argued that MAXIMA is a good tool for the task, since: it is fairly easy to learn & use; it is well documented; it has extensive facilities; it is available for any operating system; and, finally, it can be freely downloaded from the Web. MAXIMA proves to be a useful tool even for Reliability *research* for certain tasks. This latter feature provides a seamless link from teaching to research – an important feature in postgraduate education.

Key words and phrases: reliability theory, computer algebra, MAXIMA, engineering education, mathematical education.

ZDM Subject Classification: M50, N80.

1. Introduction

The author has taught over successive years a one-semester module entitled *Reliability Modelling and Analysis* (RMA) to MSc students. In this paper it is described how the Computer Algebra System (CAS) MAXIMA is used in the module as a teaching aid.

In the next section, Sect. 2, we describe the relevant MSc programmes and how RMA forms part of it. We discuss the module structure, the prerequisites

and give an account of how MAXIMA found its way into the module as a teaching aid. Furthermore, we explain the reasons for adopting MAXIMA and consider a brief summary of its attributes.

The main emphasis in the paper is on the student handouts, an edited version of some of the pertinent parts of which is given in Appendix A. They show how material in Reliability Theory can be combined with and illustrated by results produced in MAXIMA. We concentrate in Appendix A on three *themes*: (a) Approximations, (b) Markov Modelling, and, (c) Laplace Transforms. This will be discussed in detail in Section 3. In the concluding section, Section 4, we reflect on some possible future action.

2. Teaching Framework

2.1. Taught MSc Courses

2.1.1. Course Structures

Out of the nine taught MSc causes offered by our *Department of Computer Science and Applied Mathematics*, RMA is an optional module which students on the following two courses may choose,

(a) *MSc in Mobile Computing* (MC),

(b) *MSc in Networks and Performance Engineering* (NPE).

According to published material, these courses are for "graduates from science, maths, computing, engineering and related degree programmes." The postgraduate courses relevant for these two groups of students are shown in Table 1. For an MSc degree, students on these two courses have to accumulate 180 credits over three semesters, 60 credits in each. The first two of these semesters fall into the usual teaching and examination periods whereas the third one extends over the summer months.

In Table 1, compulsory and optional modules are respectively indicated by a tick (✓) and a letter '**o**'. It is seen that RMA at the most amounts to one ninth of the overall student effort.

2.1.2. Accreditation

The two MSc courses NPE and MC are approved for accreditation by the British Computer Society (BCS). More precisely, they carry 30 academic points towards membership in the BCS and exempt the successful candidate from taking

Table 1. Overview of the Postgraduate Courses NPE and MC for 2007/8.

Module Title	*Semester*	*Credit*	*Course*	
			NPE	*MC*
Mobile Applications	1	20	–	✓
Networks & Protocols	1	20	✓	✓
Performance Modelling & Telecommunications Systems	1	20	✓	–
Software Development (Postgraduate)	1	20	✓	–
Wireless Networks	1	20	–	✓
Advanced Simulation Modelling	2	20	o	✓
Artificial Intelligence with Applications	2	20	–	o
Mobile Networks Performance Modelling	2	20	–	✓
Real Time Systems (Postgraduate)	2	20	o	o
Reliability Modelling & Analysis	2	20	o	o
Software Performance Engineering Group Project	2	20	✓	–
Dissertation	3	60	✓	✓

certain examinations with the BCS. After accreditation, the member becomes a chartered engineer and a member of the BCS.

2.2. The Module RMA

2.2.1. The Module

In the module descriptor for RMA, computer skills are not specifically mentioned as a prerequisite but it may be assumed in the lectures, because of the admissions policy, that students taking this module will have the requisite background in programming. Furthermore, as RMA is delivered in the second semester, NPE students will have been learnt programming (at the latest) in semester 1 in their Software Development module. In general, the MC students come with good programming skills, some from Industry. (Some of the student on these two courses are graduates from this University and will have been taught programming in Java.)

RMA is designed to prepare students for solving reliability related problems in their own fields of specialism. It is assumed that students have some prior knowledge of Probability and Statistics (P&S). (A common initial standard in the basics is achieved by helping the weaker student on a one-to-one basis with reference to a book like [18].)

The total number of hours students are expected to spend studying for this module is around 200. There are 24 hours of timetabled sessions with lecture attendance and tutorial classes. There are in addition 16 contact hours envisaged for

individual tutorial help or revision. These sessions are arranged individually. The remaining time will be spent by students with private study: bookwork, solving tutorial exercises, the preparation of the coursework and exam preparation.

The module specifies a maximum of 20% coursework marks, and a maximum of 80% examination marks.

2.2.2. Outline Syllabus

The following topics are discussed in the present form of the module:[1]

- A revision of the relevant concepts from P&S,
- Reliability characteristics of lifetime distributions,
- Parametric classes of lifetime distributions with applications,
- The normal distribution as a limit & its applications in Quality Control,
- Reliability block diagrams,
- Maintenance modelling,
- Ordering models,
- Markov modelling (incl. a rudimentary treatment of Laplace transforms),
- Parameter estimation: Method of Moments, Maximum Likelihood, Graphical Methods,
- Censoring.

The above are broad headlines and there is scope for changing the emphasis from year to year.

2.2.3. Computer Usage

Module Evolution

There is no programming activity specified in the module descriptor but it is *possible* to set coursework which requires computer programming. In the initial years, students were asked to implement in their coursework a simulation study of an ordering model in a programming language of their choice; most of them chose Java. During the same time period, examples and graphs produced in MAXIMA were used in the *lectures* and *tutorials* in support of teaching. This was well received by students even though no quantitative assessment of the effect of using CAS has ever been performed; this so happened for the following three reasons,

[1] A selected set of topics covered in the module are discussed *in detail* in the companion paper [3].

- Student numbers are small and wildly fluctuating from year to year. Typically, there are five to ten people taking the module each year. (The present cohort is a mere three students!)
- The instructional material based on MAXIMA grew gradually and it has therefore never been a well defined point in time marking the transition from the 'without CAS' to the 'with CAS' phase.
- Finally, the module notes based on CAS are subject to constant revision and enlargement.

Why Use a CAS?

As reasoned above, no quantitative evidence can be produced to prove the utility of a CAS in teaching the module RMA. It was, however, *plausible* that employing a CAS in teaching would prove beneficial; the following points supported this thesis.

- Students have a common interest and background in computers.
- The syllabus allows in part (more precisely, in the coursework component) for computer programming to form part of the coursework. The programming could be numerical and/or symbolic; a CAS can do both.
- In the author's experience, computer generated course material enlivens the presentation and makes 'dry' (i.e. formal, Mathematics based) material more acceptable to non-mathematicians. In particular, the possibility of showing computer generated graphs and illustrations during lectures, has proved a great asset as this method is superior to the traditional 'chalk-and-talk' approach.
- There are many reported cases in the literature of successful computer usage in Engineering education. Examples thereof are [9], [5] and [8]. Some textbooks use Computer Algebra Systems (CAS) for teaching cognate subjects such as P&S (e.g. [7]). (In all these reported cases commercial software is used.)

Why Use MAXIMA*?*

Initially, the use of MATLAB was envisaged as many of our students have met it before. It transpired soon, however, that for lack of licensing, a full version of MATLAB (incl. its Symbolic Math Toolbox) was not available. After some research, it has been decided to build up a portfolio of teaching material written in MAXIMA, a *non-commercial* CAS. The emphasis here is truly on *free availability*. Furthermore, MAXIMA was identified, after some experimentation, as a viable alternative to MATLAB's Symbolic Math Toolbox. It is well documented ([6],

[15], [10]) and comprehensive. In favour of MAXIMA was also the fact that the author had experience with it in the reliability context ([1], [2]).

Over the years, a large amount of MAXIMA based teaching material, in the form of lecture notes, exercises and coursework sheets, has been accumulated.

An interesting account of the history of CAS is in [11], with a prominent role given to MAXIMA. We admit freely that the decision to use MAXIMA was made at the time without knowing [11]. But, with hindsight, we feel vindicated: MAXIMA is still one of the best known non-commercial CAS around which suited our requirements.

Finally, the article [13] (accessible for readers in German) is yet another recent account of the sustained interest in MAXIMA.

It has been brought to my attention by one of the referees that the free software Sage [14] supports computation with objects in many different computer algebra systems in a unified fashion using a common interface.

MAXIMA

MAXIMA is a large, mature CAS which seems to have served as a model (a predecessor) for many of today's commercial systems [11]. It was written in LISP, the first functional programming language. (In addition to its own programming language, MAXIMA accepts LISP code.[2]) The main criticism directed at MAXIMA is the lack of a graphical user interface. This means that the output is normally returned in a typewriter font, perhaps not a pretty sight, but of no importance as far as its computational power goes. Furthermore, the function `tex` can be used to make MAXIMA to return the output in a format ready for inclusion later in a LaTeX document. MAXIMA is available for every operating system and it is well documented in the literature (e.g. [6]) as well as through an online manual.

There is no attempt in the module to teach MAXIMA to the students in a systematic fashion. Instead, a collection of pertinent examples (as demonstrated here) is given to them with the expectation that competence will be attained by studying these examples in conjunction with the literature on MAXIMA. Students may, if they choose to do so, ignore the available MAXIMA material alltogether without affecting their measured 'success' in the module as MAXIMA knowledge will not be assessed. It is felt, however, that such a course of action would be immature and, indeed, the majority of our MSc students are curious enough to be interested also in non-assessed material.

[2]This is yet another point which some of our students may find of interest as some with a first degree from here have a background in the fuctional language HASKELL, [17].

3. Discussion

We concentrate here on three selected topics: *Normal Approximations*, *Markov Modelling* and *Laplace Transforms*, addressed in Section 3.1, Section 3.2 and Section 3.3, respectively.

The full report [4] with many additional examples is available from the author.

3.1. Normal Approximations

The Normal Approximation is introduced in Section A.1.1 with a view to discussing later questions in Quality Control. We start by stating an appropriate form of the Central Limit Theorem (CLT), followed by a series of examples.

The first three examples are designed to illustrate the CLT for exponentially distributed summands. A MAXIMA implementation of the convolution operation is shown in Example A.1, code M.A.1. Noteworthy is that `convn_all` *recursively* defines a *list* of self-convolutions of a given function. Lists are an important data structure in MAXIMA as well as in LISP, the programming language in which MAXIMA was written. Our students do not have direct knowledge of LISP, the first *functional* programming language, but those who did our first degree in Computer Science will have been taught HASKELL [17], a modern descendant of LISP. Therefore, code like that shown in M.A.1 is readily accessible for a great number of students on our course. In M.A.2, we draw by MAXIMA the first few convolution densities by using `plot2d`. We want to give to students a 'template' for plotting graphs rather than a complete, reasoned set of instructions for doing so. It is hoped that this and subsequent examples will enable them to use [6] and [15] independently. After standardizing in Example A.2, a visual impression is given in Example A.3 of the quality of the normal approximation afforded by the CLT.

Example A.4 is designed to illustrate the CLT for a wildly oscillating density, perhaps not seen in practice, which, however, is interesting to students precisely because of its unusual shape and also because it allows certain numerical features of MAXIMA (such as `numer` and `quad_qagi`) to be demonstrated.

3.2. Markov Modelling

For the Markov material reviewed here, the students may wish to consult also Ramakumar's textbook [12].

Kolmogorov Equations

Access to Markov models is through the Komogorov equations in Sect. A.2.1.

Example A.5 is the smallest, two-state system to model a process with exponential holding times. It serves to introduce the terminology and defines the Kolmogorov equations which then are solved symbolically in MAXIMA with M.A.8. The example is concluded with a short digression on the numerical way using SCILAB.

In Example A.6, we consider an $(n+1)$-state absorbing Markov model of the n-stage Erlang distribution. Noteworthy is here in M.A.9 the implementation of the Kolmogorov equations by `stages`. Several MAXIMA features are exemplified there: list manipulation, symbolic differentiation, recursive definition, conditional. The example is concluded with a practical application calling for MAXIMA's numerical capabilities.

In the last item in this section, Exercise A.1, the reader is asked to apply the techniques learnt to a more complex, 6-state repair model. The student is asked in part (d) to apply the MAXIMA function `linsolve`, not seen heretofore to obtain the long term solution of the system symbolically. Two alternatives for defining the auxiliary function `take` are shown in the model solution, one by iteration, in M.A.12, and one by recursion, in M.A.13. It is good educational practice to consider both approaches.

3.3. Laplace Transforms

We indicate in Section A.3.1 the reason for the limitations of the approaches taken in Section A.2 for analyzing Markov models. Laplace transforms are introduced briefly in Section A.3.2 and the exponential density is taken to illustrate some of the corresponding MAXIMA functions. The next and only example in this section, Example A.8, is a continuation of work started in Exercise A.1. The modified model is now absorbing and Laplace transforms are used for obtaining moments of the time to system failure.

4. Conclusions

Aspects of an MSc module in Reliability Theory supported by the CAS MAXIMA have been described here.

Are there alternatives to MAXIMA? If one insists on a non-commercial system, there aren't many. The only other, more recent one is YACAS ('Yet Another

Computer Algebra System', [19]) which to the best of the author's knowledge has not been used in conjunction with a similar lecture course. There is scope for future work here.

Not all module topics have as yet been covered by applications and examples written in MAXIMA. Implementation in MAXIMA of pertinent material for this module is an ongoing task.

Acknowledgments

Financial assistance by my former Head of the Department, Professor Mike Woodward, made it possible for me to participate in the IMA/SEFI Conference *Mathematical Education of Engineers* in Loughborough in April 2008. This paper is based on material presented in that conference.

The help of Miss Andrea Ryan is appreciated in compiling material on the courses using RMA.

Two referees have commented on the original version [4] of this contribution. As a result, I have drastically shortened the paper, still hoping to have retained its original features. I have also more carefully reflected upon alternatives to MAXIMA and on some (free) software supplementing it.

References

[1] A. Csenki, Joint interval reliability for Markov systems with an application in transmission line reliability, *Reliability Engineering and System Safety* **92**, 2007, 685–696.

[2] A. Csenki, On the three-state weather model of transmission line failures, Proceedings of the Institution of Mechanical Engineers, Part O, *Journal of Risk and Reliability* **221**, 2007, 217–228.

[3] A. Csenki, Salient features of a lecture course in Reliability Theory, *International Journal of Mechanical Engineering Education* **36**, 2008, 339–365.

[4] A. Csenki, Teaching reliability theory with the computer algebra system MAXIMA, *Technical Report, School of Computing, Informatics, Media, The University of Bradford*, 2008.

[5] K. M. Dempsey, J. H. Kane and J. P. Kurtz, BEAMTOOL: Interactive beam analysis for today's student and engineer, *Computer Applications in Engineering Education* **13**, 2005, 293–305.

[6] B. Heller, *MACSYMA for Statisticians*, NY: Wiley-Interscience, New York, 1991.

[7] J. J. Kinney, *Probability – An Introduction with Statistical Applications*, NY: Wiley, New York, 1997.

[8] D. G. Knight, Revisiting Newtonian and non-Newtonian fluid mechanics using computer algebra, *International Journal of Mathematical Education in Science and Technology* **37**, 2006, 573–592.

[9] J. H. Mathews, Using a computer algebra system to teach double integration, *International Journal of Mathematical Education in Science and Technology* **21**, 1990, 723–732.

[10] MAXIMA website, `http://maxima.sourceforge.net/`.

[11] F. Pohlmann, Doing the Sums: Linux, GNU amd Mathematics, *Linux User & Developer* **53**, 2005, 52–55.

[12] R. Ramakumar, *Engineering Reliability – Fundamentals and Applications*, NJ: Prentice-Hall, Englewood Cliffs, 1993.

[13] T. Romeyke, Algebra mit Maxima, *Linux Magazin* **09/08**, 2008, 32–35 (in German).

[14] The SAGE website, `http://www.sagemath.org/`.

[15] W. F. Schelter, MAXIMA Manual, Version 5.9.3, 2006, `http://maxima.sourceforge.net/docs/manual/en/maxima.html`.

[16] Introduction to SCILAB, version 4.0, User Guide, 2006, `http://www.scilab.org/ftp://ftp.inria.fr/INRIA/Scilab/documentation/pdf/intro.pdf`.

[17] S. Thompson, *Haskell – The Craft of Functional Programming*, Second edition, Addison-Wesley, Harlow, England, 1999.

[18] R. E. Walpole and R. H. Myers, *Probability and Statistics for Engineers and Scientists*, NY: Macmillan, New York, 1978.

[19] YACAS website, `http://yacas.sourceforge.net/homepage.html`.

Appendix A. Edited Excerpts from Student Handouts

A.1. Approximations

A.1.1. The Normal Approximation

We start by stating the *Central Limit Theorem* and then consider several examples eventually leading to an illustration of the quality of approximation afforded thereby.

The *Central Limit Theorem* tells us that many random variables have a distribution which is well approximated by a normal distribution. This theorem is stated in class informally and a mathematically oriented textbook on statistics is recommended for the exact conditions under which it holds.

Central Limit Theorem. Let $T_1, \ldots, T_n$ be random variables with respective means $a_1, \ldots, a_n$ and variances $\sigma_1^2, \ldots, \sigma_n^2$. Then, if the individual standardized

random contributions are small, the distribution of the standardized sum is approximately standard normal, i.e.

$$P\left(\frac{\sum_{i=1}^{n}(T_i - a_i)}{\sqrt{\sum_{i=1}^{n}\sigma_i^2}} \leq x\right) \approx \Phi(x). \tag{1}$$

□

Example A.1. (*Erlang Densities.*) Let us assume that a system comprises of n consecutively run units with respective lifetimes $T_1, \ldots, T_n$ each of which is exponentially distributed with rate λ. The system lifetime, $S_n = T_1 + \ldots + T_n$ is then Erlang$-(\lambda, n)$ distributed. To find with MAXIMA the pdf of S_n, we define `conv` and `conv_all` in M.A.1.

Maxima Code M.A.1: *Implementing convolutions*

```
1 conv(fU, fV) :=
2  (fU: subst(t-y, t, fU), fV: subst(y, t, fV), integrate(fU*fV, y, 0, t));
3 convn_all(fun, n) :=
4  (if n = 1 then
5    [fun]
6   else
7    (convs : convn_all(fun, n - 1), cons(conv(fun, first(convs)), convs)));
```

The MAXIMA function `conv` convolves the functions (densities) f_U and f_V according to

$$(f_U * f_V)(t) = \int_0^t f_U(t-y) f_V(y) dy, \tag{2}$$

whereas `convn_all` returns the *list* of the first n convolutions of a given density with itself. For example, the densities of the first five partial sums $S_1, \ldots, S_5$ are obtained thus

```
(%i3) convn_all(lambda * %e^(- lambda * t), 5);
Is  t  positive, negative, or zero?

pos;
        4       5   - t lambda   3       4   - t lambda
       t  lambda  %e            t  lambda  %e
(%o3) [----------------------, ----------------------,
                24                        6
         2       3   - t lambda
        t  lambda  %e                    2   - t lambda           - t lambda
          ----------------------, t lambda  %e          , lambda %e          ]
                     2
```

The five stage Erlang density (the first entry of the above list) is extracted for later reference as follows.

```
(%i4) first(%);
                              4       5   - t lambda
                             t  lambda  %e
(%o4)                        ----------------------
                                      24
```

Assuming unit rate ($\lambda = 1$), the densities of $S_1, \ldots, S_5$ are plotted in Fig. 1 using the MAXIMA code M.A.2.

Maxima Code M.A.2: *Drawing Fig. 1*

```
plot2d(convn_all(%e^(-t),5),
         [t, 0, 10],
         [gnuplot_curve_styles, [``with lines 7"]],
         [nticks, 50],
         [gnuplot_term, ps],
         [gnuplot_out_file, ``conv_exp.ps"],
         [gnuplot_preamble, ``set title 'Convolution densities';
                             set grid;
                             set nokey;
                             set xlabel 'time';
                             set ylabel 'pdf'"]);
```

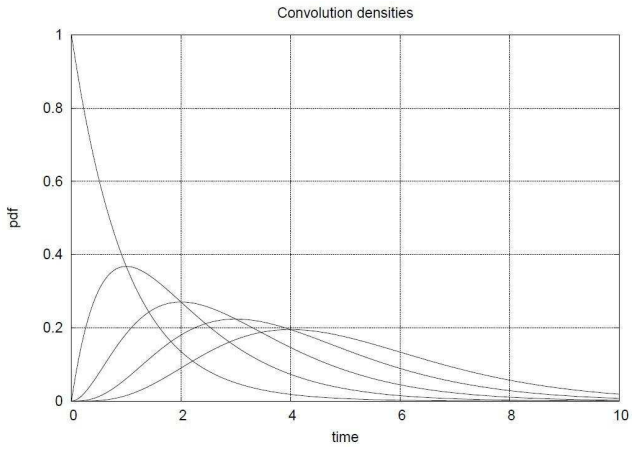

Figure 1. Convolving exponentials with unit rates

Example A.2. (*Standardization.*) Next we introduce the notion of *standardization.* Let U be a random variable whose pdf, mean and standard deviation are f_U, a_U and σ_U, respectively. Then the random variable $V = (U - a_U)/\sigma_U$

is referred to as the *standardized* version of U. It has zero mean, unit standard deviation and pdf

$$f_V(v) = \sigma_U f_U(\sigma_U v + a_U). \tag{3}$$

Using (3), we implement in M.A.3 the standardization of a pdf (which is written as a function of t) by the function `standard`.

Maxima Code M.A.3: *Definition of* `standard`

```
1 standard(pdf,mean,std) := std * subst(std * t + mean, t, pdf);
```

As an example, we standardize a normal pdf

$$\text{pdf}(t) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(t-a)^2}{2\sigma^2}},$$

thus

```
(%i14) standard(1/(sigma*sqrt(2*%pi))*%e^(-(t-a)^2/(2*sigma^2)), a,sigma);
                                          2
                                         t
                                       - --
                                         2
                                    %e
(%o14)                        -----------------
                              sqrt(2) sqrt(%pi)
```

and, as expected, we thereby obtain the standard normal density. □

Example A.3. (*Standardized Erlangs.*) Fig. 2 shows the standardized convolution densities for $n = 10$ and $n = 50$, as well as the limiting density ϕ. To draw the Fig. 2, the list of the first fifty standardized convolution densities was produced; this was accomplished by the MAXIMA function `dens_list`, defined in M.A.4.

Maxima Code M.A.4: *Definition of* `dens_list`

```
1 dens_list(pdf, mean, std, n) := maplist(standard,
2                                         reverse(convn_all(pdf, n)),
3                                         mean * makelist(i, i, 1, n),
4                                         std * makelist(sqrt(i), i, 1, n));
```

Fig. 2 was plotted by M.A.5.

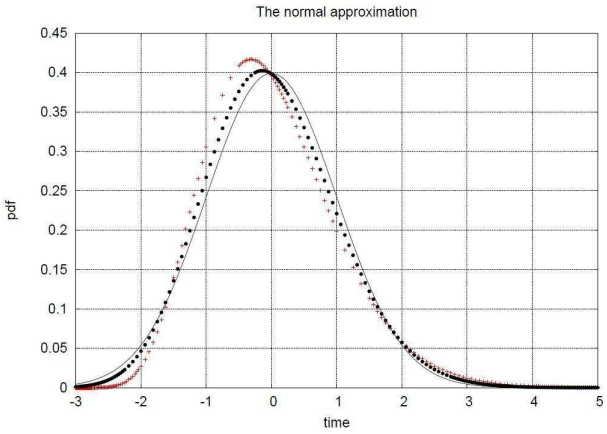

Figure 2. The normal approximation: $n = 10$ (+) and $n = 50$ (•)

Maxima Code M.A.5: *Drawing Fig. 2*

```
plot2d([tenth(densities), last(densities), phi],
       [t, -3, 5],
       [gnuplot_curve_styles, [``with points 1",
                               ``with points 7",
                               ``with lines 7"]],
       [nticks, 1],
       [gnuplot_term, ps],
       [gnuplot_out_file, ``normal_approx.ps"],
       [gnuplot_preamble, ``set title 'The normal approximation';
                           set grid;
                           set nokey;
                           set xlabel 'time';
                           set ylabel 'pdf'"]);
```

The code in M.A.5 uses the MAXIMA functions `phi` (the standard normal density ϕ) and `densities`; they are defined in M.A.6.

Maxima Code M.A.6: *Definitions for M.A.5*

```
phi       : %e^(-t^2 / 2) / sqrt(2*%pi);
densities : dens_list(%e^(-t), 1, 1, 50);
```

(As a by-product, the latter also displays on the terminal the list of the first fifty symbolic expressions for the convolution densities. This was suppressed by setting `ttyoff : true`.) □

Example A.4. (*An Oscillating Density.*) The density of the life distribution we are going to consider here is assumed to have the shape of an attenuating cosine,

$$f(t) \sim (1 + cos(10t))e^{-t}.$$

We use MAXIMA to find out what the normalizing constant should be in order for f to be a pdf.

```
(%i1) integrate ((1 + cos(10*t)) * %e^(-t),t,0,inf);
Principal Value
                                    102
(%o1)                               ---
                                    101
```

Thus,

$$f(t) = \frac{101}{102}\left(1 + cos\left(10t\right)\right)e^{-t}, \qquad t > 0,$$

is a lifetime pdf. M.A.7 draws the graph of f in MAXIMA (Fig. 3).

Maxima Code M.A.7: *Drawing Fig. 3*

```
1  f : (101/102) * (1 + cos(10*t)) * %e^(-t);
2  plot2d(f, [t, 0, 5],
3            [gnuplot_curve_styles, [``with lines 7"]],
4            [nticks, 50],
5            [gnuplot_term, ps],
6            [gnuplot_out_file, ``osc_dens.ps"],
7            [gnuplot_preamble, ``set title 'An oscillating density';
8                               set grid;
9                               set nokey;
10                              set xlabel 't';
11                              set ylabel 'f(t)'"]);
```

f appears rather artificial and it indeed does *not* correspond to any known real situation. It is, however, suitable for illustrating the applicability of the Central Limit Theorem even for a small number of summands.

To find out more about our distribution, we determine its mean: we use MAXIMA to calculate the indefinite integral of $tf(t)$ and substitute $t = 0$ (the lower limit of integration).

```
(%i4) subst(0,t,integrate(t*f,t));

                                    5051
(%o4)                             - ----
                                    5151
```

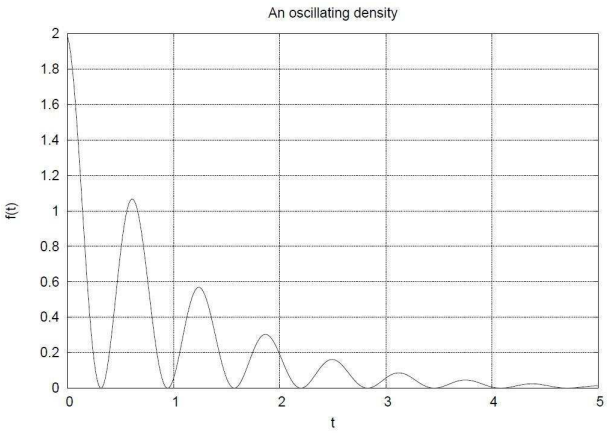

Figure 3. An oscillating density

As the integral vanishes for $t = +\infty$ (this we know by visual inspection of the explicit form of the antiderivative produced by MAXIMA, not shown here), the *exact* value of the mean is $\frac{5051}{5151}$. A decimal approximation thereof we obtain by

```
(%i5) - %, numer;
(%o5)                          0.98058629392351
```

We may also use one of the numerical integration routines of MAXIMA to find out an approximate value for the mean.

```
(%i6) quad_qagi(''f*t,t,0,inf);
...
(%o6)          [0.9805862939463, 6.514429589477498E-9, 1155, 0]
```

The integral is the first entry in this list.

```
(%i7) mean : % [1];
(%o7)                            0.9805862939463
```

(The above is seen to be for practical purposes identical to the previous value.) In a similar fashion we compute the second central moment.

```
(%i8) quad_qagi(''f*t*t,t,0,inf) [1];
...
(%o8)                          1.979817434269797
```

The standard deviation is then obtained by

```
(%i9) std : sqrt(% - mean^2);
(%o9)                          1.009092639153837
```

Similarly to the earlier example, we now create the list of all convolution densities up to degree ten by

```
(%i10) densities : dens_list(f, mean, std, 10);
...
```

and finally plot the 5th, the 10th normalized densities and the normal pdf ϕ by

```
(%i11) plot2d ([fifth(densities), last(densities), phi], [t, -2, 5], ...
```

The result is shown in Fig. 4.

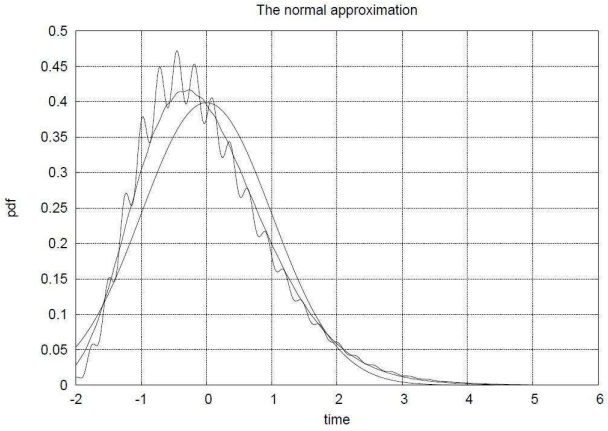

Figure 4. Normal approximations for an oscillating density for $n = 5, 10$

It is seen that in spite of the oscillating behaviour of the density f, after ten convolutions it becomes 'smooth'. And, even after convolving the density five times only, it is reasonably well approximated by the standard normal pdf. □

A.2. Markov Modelling

A.2.1. Kolmogorov Equations

Example A.5. (*Two-state System.*) We consider a two-state system with *transition rate diagram* (or *Markov diagram*) as shown in Fig. 5.

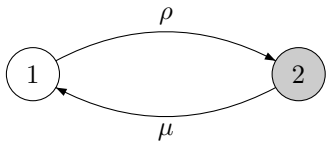

Figure 5. The two-state system

It comprises a single machine which can be in one of two states: the working state ① and the repair state ② (shaded). Failure and repair rates are ρ and μ, respectively. Denoting by $\mathbf{\Lambda} = [\lambda_{ij}]_{i,j=1,2}$ the system's transition rate matrix, the Kolmogorov equations are

$$\frac{dp_1(t)}{dt} = \lambda_{11}p_1(t) + \lambda_{21}p_2(t),$$
$$\frac{dp_2(t)}{dt} = \lambda_{12}p_1(t) + \lambda_{22}p_2(t),$$

or, more explicitly,

$$\frac{dp_1(t)}{dt} = -\rho p_1(t) + \mu p_2(t), \tag{4}$$
$$\frac{dp_2(t)}{dt} = \quad \rho p_1(t) - \mu p_2(t). \tag{5}$$

Both (4) and (5) are readily written down with reference to the system's Markov diagram in Fig. 5:

- On the left hand side, $\frac{dp_i(t)}{dt}$ is the rate of flow of probability *into* state i.
- On the right hand side,
  - The coefficient of $p_i(t)$ is the negative sum of all transition rates labelling *outgoing* edges from state i. This is the ith diagonal entry of the rate matrix. It is the *rate of loss* of probability of state i.
  - The coefficient of $p_j(t)$, $j \neq i$, is the rate labelling the *incoming* edge pointing to state i from state j. It is the *rate of gain* of probability by state i, received from state j.

Furthermore,

- Once the Kolmogorov equations are set up, we may write down the transpose of the transition rate matrix by simply copying the pattern of coefficients as they appear in the equations.

$$\mathbf{\Lambda}^t = \begin{pmatrix} -\rho & \mu \\ \rho & -\mu \end{pmatrix} \tag{6}$$

These rules can be used to obtain the Kolmogorov equations and the transition rate matrix for systems of any size.

To solve the Kolmogorov equations, we need some *initial conditions*. Let us assume that at time zero the machine is working, i.e. it is in state ①:

$$p_1(0) = 1, \qquad p_2(0) = 0. \tag{7}$$

Equations (4)–(7) are solved in MAXIMA by M.A.8.

Maxima Code M.A.8: *Solving (4)–(7)*

```
eq1 : diff(p1(t), t) = - rho * p1(t) + mu * p2(t);
eq2 : diff(p2(t), t) = rho * p1(t) - mu * p2(t);
atvalue(p1(t), t = 0, 1);
atvalue(p2(t), t = 0, 0);
desolve([eq1, eq2], [p1(t), p2(t)]);
```

Running M.A.8 produces the output

```
                      - (rho + mu) t
               rho %e                       mu
(%o10) [p1(t) = -------------------- + --------,
                      rho + mu          rho + mu
                                                          - (rho + mu) t
                                             rho      rho %e
                                  p2(t) = -------- - --------------------]
                                          rho + mu          rho + mu
```

A visually more appealing output can be achieved by running the `tex` command.

```
(%i11) tex(%);
$$\left[ {\it p_1}\left(t\right)={{\rho\,e^{- \left(\rho+\mu\right)
 \,t }}\over{\rho+\mu}}+{{\mu}\over{\rho+\mu}} , {\it p_2}\left(t
 \right)={{\rho}\over{\rho+\mu}}-{{\rho\,e^{- \left(\rho+\mu\right)
 \,t }}\over{\rho+\mu}} \right] $$
(%o11)                                false
```

The output thus produced is displayed in LaTeX as shown below.

$$\left[p_1\left(t\right)=\frac{\rho\,e^{-\left(\rho+\mu\right)\,t}}{\rho+\mu}+\frac{\mu}{\rho+\mu},p_2\left(t\right)=\frac{\rho}{\rho+\mu}-\frac{\rho\,e^{-\left(\rho+\mu\right)\,t}}{\rho+\mu}\right]$$

This shows that the long term *point availability*, which is defined as $\lim_{t\to\infty} p_1(t)$, is $\mu/(\rho+\mu)$, whereas the long term *point unavailability*, defined as $\lim_{t\to\infty} p_2(t)$, is $\rho/(\rho+\mu)$.

Thus far we have a list of symbolic solutions. We may use it for generating *numerical* solutions. As an example, we calculate the point availability $p_1(t)$ for $t = 100$ with the failure rate $\rho = 0.01$ and repair rate $\mu = 0.1$ by

```
(%i12) ev(rhs(first(%o10)), rho = 0.01, mu = 0.1, t = 100);
(%o12)                        .9090924274273446
```

An alternative to the above method of solving the Kolmogorov equations is by using *matrix exponentials*. A closed form expression for the solution of the Kolmogorov equations (4)–(5) is given by

$$\mathbf{p}(t) = \mathbf{p}(0)e^{t\mathbf{\Lambda}}, \tag{8}$$

where the rate matrix $\mathbf{\Lambda}$ is defined in (6). If the prime interest is in a numerical rather than a symbolic solution then we may evaluate the right hand side of (8) with a system where the matrix exponential is implemented. SCILAB[3] confirms our earlier result thus

```
-->p = [1 0] * expm(100 * [-0.01 0.01; 0.1 -0.1]); disp(p(1));

   0.9090924
```

□

Example A.6. (*Markov chain model of an* $(n+1)$*-stage Erlang distribution.*) Consider the $(n+1)$-state Markov chain with transition rate diagram as shown in Fig. 6.

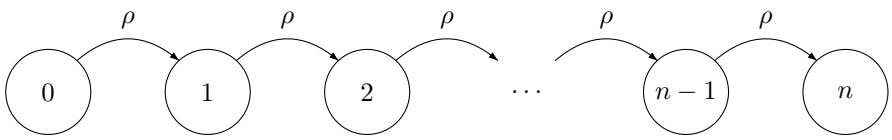

Figure 6. Modelling the Erlang distribution.

Let the system start in state 0 at time $t = 0$. The time of arrival in state n, the absorbing state, is then Erlang-n distributed with some parameter $\rho > 0$; let us call its cdf $p_n(t)$. The system's Kolmogorov equations are

$$\begin{aligned}
\frac{dp_0(t)}{dt} &= -\rho p_0(t) \\
\frac{dp_1(t)}{dt} &= \rho p_0(t) - \rho p_1(t) \\
&\vdots \\
\frac{dp_{n-1}(t)}{dt} &= \rho p_{n-2}(t) - \rho p_{n-1}(t) \\
\frac{dp_n(t)}{dt} &= \rho p_{n-1}(t)
\end{aligned} \tag{9}$$

Equation (9) is implemented by the MAXIMA function `stages`, defined in M.A.9.

[3]SCILAB is a MATLAB-like free programming environment of French origin [16]. Its use won't be pursued here further.

Maxima Code M.A.9: *Implementation of (9)*

```
stages(n) :=
 (if n = 1 then
   block(eq0 : diff(p0(t), t) = - rho * p0(t),
      eq1 : diff(p1(t), t) =   rho * p0(t),
      atvalue(p0(t), t = 0, 1),
      atvalue(p1(t), t = 0, 0),
      eqs   : [eq0, eq1],
      probs : [p0(t), p1(t)])
  else
   block(stages(n - 1),
      concat(eq, (n-1)) :: diff(concat(p, (n-1))(t), t) =
                           rho * concat(p, (n-2))(t) - rho * concat(p, (n-1))(t),
      concat(eq, n)     :: diff(concat(p, n)(t), t) = rho * concat(p, (n-1))(t),
      atvalue(concat(p, n)(t), t = 0, 0),
      eqs   : endcons(concat(eq, n), endcons(concat(eq, (n-1)), init(eqs))),
      probs : endcons(concat(p, n)(t), probs)));
```

Noteworthy are the following features of `stages`.

- It is defined by *recursion*, exploiting the patterned nature of the coefficients on the right hand side of (9).
- It creates the *global* variables `eqs` and `probs`, to be used later in the arguments of MAXIMA's differential equations solver `desolve`.
- Finally, it uses the *auxiliary* function `init` which returns the initial section of the input list. Before defining `stages` by M.A.9, declare `init` by

  ```
  init(list) := reverse(rest(reverse(list)));
  ```

Once the Kolmogorov equations are available, the Erlang cdf for a given value of n is returned by `ecdf`, defined in M.A.10.

Maxima Code M.A.10: *Implementation of the Erlang cdf & pdf*

```
ecdf(n) := (kill(eqs, probs), stages(n), rhs(last(desolve(ev(eqs), ev(probs)))));
epdf(n) := diff(ecdf(n), t);
```

For $n = 5$, for example, we get respectively the cdf and pdf of the five stage Erlang distribution by

```
(%i5) tex(ecdf(5));
```

$$-\frac{\rho^4\,t^4\,e^{-\rho\,t}}{24}-\frac{\rho^3\,t^3\,e^{-\rho\,t}}{6}-\frac{\rho^2\,t^2\,e^{-\rho\,t}}{2}-\rho\,t\,e^{-\rho\,t}-e^{-\rho\,t}+1$$

```
(%i6) tex(epdf(5));
```

$$\frac{\rho^5\, t^4\, e^{-\rho\, t}}{24}. \tag{10}$$

We have met the Erlang density (10) in Example A.1 before.

Application. A component's lifetime is known to follow an exponential distribution with mean unity. (The expected lifetime is taken as the unit of time.) A submarine carries on board five such components, one original and four spares. The system's lifetime is the *total* of the five component lifetimes. What is the probability that the system will survive a mission of length 2?

This question is answered by MAXIMA simply by

```
(%i7) 1 - float(subst([rho = 1, t = 2], ecdf(5)));
(%o7)                          0.94734698265629
```

□

Exercise A.1. (*Complex System, Asymptotic Behaviour.*) A system comprises two machines, A and B, and one repairman. The system can be in one of the following six states.

① Machine A is working and machine B is on standby, ready to operate as soon as A fails.

② This state is analogous to state 1 with the labels A and B interchanged.

③ Machine A is working and machine B is being repaired.

④ This state is analogous to state 3 with the labels A and B interchanged.

⑤ Both machines are down and the repairman is busy repairing machine A.

⑥ Both machines are down and the repairman is busy repairing machine B.

Failure and repair rates are ρ_A, ρ_B and μ_A, μ_B, respectively. It is assumed that a machine on standby won't deteriorate and that the repairman completes a repair already started, before possibly attending the next repair. The system is deemed operational if one of the machines is working. (The other may be on standby or it may be in the repair shop.) Initiall, the system is in state ①.

(a) Draw the system's transition rate diagram.

(b) Write down the Kolmogorov differential equations for the state probabilities.

(c) Write down the system's transition rate matrix.

(d) Assume now that the two machines are nominally identical and therefore the failure and repair rates are identical each, i.e. $\rho_A = \rho_B = \rho$ and $\mu_A = \mu_B = \mu$, say.

(i) Write down the equations which the *long term* state probability vector $\mathbf{p} = (p_1, \ldots, p_6)$ satisfies.

(ii) Obtain $\mathbf{p}$ by MAXIMA's function `linsolve`.

(iii) Write down an expression for the system's long term availability in terms of components' failure and repair rates. Compute the value of the long term availability for the case when the mean time to repair is one tenth of the mean time to failure.

Solution A.1.

(a) In Fig. 7, the system states are labelled with four-tuples whose entries mean in turn: *working*, *standby*, under *repair*, *waiting for repair*.

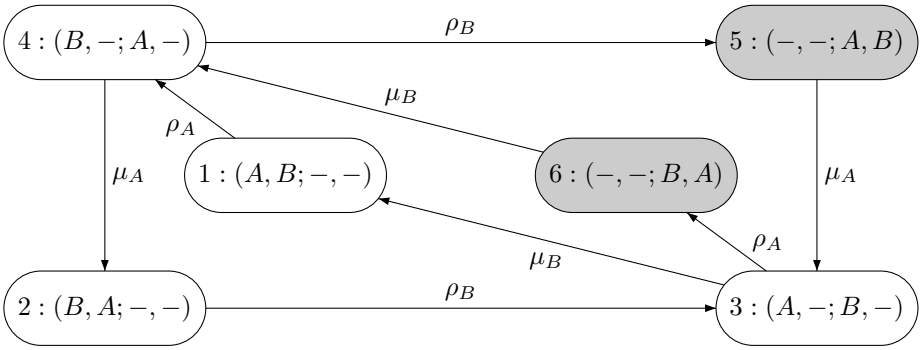

Figure 7. Markov model of a two-machine-one-repairman system.

(b)

$$\frac{dp_1(t)}{dt} = -\rho_A p_1(t) + \mu_B p_3(t) \tag{11}$$

$$\frac{dp_2(t)}{dt} = -\rho_B p_2(t) + \mu_A p_4(t) \tag{12}$$

$$\frac{dp_3(t)}{dt} = -(\mu_B + \rho_A) p_3(t) + \rho_B p_2(t) + \mu_A p_5(t) \tag{13}$$

$$\frac{dp_4(t)}{dt} = -(\mu_A + \rho_B) p_4(t) + \rho_A p_1(t) + \mu_B p_6(t) \tag{14}$$

$$\frac{dp_5(t)}{dt} = -\mu_A p_5(t) + \rho_B p_4(t) \tag{15}$$

$$\frac{dp_6(t)}{dt} = -\mu_B p_6(t) + \rho_A p_3(t) \tag{16}$$

The initial conditions are $p_1(0) = 1$, $p_2(0) = \ldots = p_6(0) = 0$.

(c) From (11)–(16), the transition matrix $\mathbf{\Lambda}$ is

$$\mathbf{\Lambda} = \begin{pmatrix} -\rho_A & 0 & 0 & \rho_A & 0 & 0 \\ 0 & -\rho_B & \rho_B & 0 & 0 & 0 \\ \mu_B & 0 & -(\mu_B + \rho_A) & 0 & 0 & \rho_A \\ 0 & \mu_A & 0 & -(\mu_A + \rho_B) & \rho_B & 0 \\ 0 & 0 & \mu_A & 0 & -\mu_A & 0 \\ 0 & 0 & 0 & \mu_B & 0 & -\mu_B \end{pmatrix}$$

(d) (i) Set the left hand sides of (11)–(16) to zero to get

$$0 = -\rho p_1 + \mu p_3 \tag{17}$$
$$0 = -\rho p_2 + \mu p_4 \tag{18}$$
$$0 = -(\mu + \rho) p_3 + \rho p_2 + \mu p_5 \tag{19}$$
$$0 = -(\mu + \rho) p_4 + \rho p_1 + \mu p_6 \tag{20}$$
$$0 = -\mu p_5 + \rho p_4 \tag{21}$$
$$0 = -\mu p_6 + \rho p_3 \tag{22}$$

The seventh equation is the normalizing condition

$$p_1 + \ldots + p_6 = 1 \tag{23}$$

(ii) It is $\mathbf{p} = c^{-1}(\mu^2, \mu^2, \mu\rho, \mu\rho, \rho^2, \rho^2)$ with $c = 2(\mu^2 + \mu\rho + \rho^2)$. We obtain this in MAXIMA by M.A.11.

Maxima Code M.A.11: *Solving (17)–(22) & (23)*

```
soln : linsolve([- rho * p1 + mu * p3,
                 - rho * p2 + mu * p4,
                   rho * p2 - (mu + rho) * p3 + mu * p5,
                   rho * p1 - (mu + rho) * p4 + mu * p6,
                   rho * p4 - mu * p5,
                   rho * p3 - mu * p6,
                   p1 + p2 + p3 + p4 + p5 + p6 - 1],
                [p1, p2, p3, p4, p5, p6]);
```

(iii) The long term availability is

$$p_1 + p_2 + p_3 + p_4 = \frac{\mu^2 + \mu\rho}{\mu^2 + \rho^2 + \mu\rho} = \frac{1 + \frac{\rho}{\mu}}{1 + \frac{\rho}{\mu} + \left(\frac{\rho}{\mu}\right)^2} \approx 0.991$$

We confirm this by first running M.A.11 and then proceeding with

```
(%i3) probs : subst([mu = 10 * rho], maplist(rhs,soln));
        50   50   5    5    1    1
(%o3) [---, ---, ---, ---, ---, ---]
       111  111  111  111  222  222
(%i4) float(lsum(i,i,take(4,probs)));
(%o4) 0.99099099099099
```

A function `take` is defined here for creating a list consisting of a specified number of entries of a given list. Two definitions of `take` are shown below, one based on iteration (in M.A.12), the other based on recursion (in M.A.13).

Maxima Code M.A.12: *Definition of* `take` *by iteration*

```
1  take(n, list) :=
2    (if n < length(list) then
3        block([temp, accum],
4              accum : [],
5              temp  : list,
6              for i : 1 thru n do
7                  (accum : endcons(first(temp), accum),
8                   temp  : rest(temp)),
9                   accum)
10    else
11       list);
```

Maxima Code M.A.13: *Definition of* `take` *by recursion*

```
1  take(n, list) :=
2    (if n < length(list) then
3        if n = 0 then
4           []
5        else
6           cons(first(list),take(n - 1, rest(list)))
7    else
8       list);
```

□

A.3. Laplace Transforms

A.3.1. Why Use Laplace Transforms?

With `desolve` (e.g. M.A.8), MAXIMA solves the Kolmogorov equations by transforming them into the *Laplace Transform* domain where a system of *linear* equations is established for the Laplace transforms $p_1^*(s), p_2^*(s), \ldots$ of the

unknown functions $p_1(t), p_2(t), \ldots$. MAXIMA solves these equations for the transforms $p_1^*(s), p_2^*(s), \ldots$ and then tries to invert each of them to obtain the original functions $p_1(t), p_2(t), \ldots$. This is where problems can arise. As each of the Laplace transforms is a rational function whose denominator is a polynomial of degree n, the number of states, the system may not be able to carry out a partial fraction decomposition as there is no closed form expression for the roots of a polynomial of degree $n \geq 5$.[4] MAXIMA needs, however, a partial fraction decomposition of the Laplace transforms for termwise invertion.

The cause of the difficulties should not really concern us here too much as we have set out to rely entirely on MAXIMA. However, it turns out that knowing some facts about Laplace transforms and using MAXIMA will help us to get a *partial* solution also to larger problems, i.e. $n \geq 5$.

A.3.2. Laplace Transforms

We collect here for later reference some facts concerning Laplace transforms. For a function $f(t)$ defined for $t > 0$, its *Laplace Transform* is defined by

$$f^*(s) = \int_0^\infty e^{-st} f(t) dt. \tag{24}$$

Knowing f^* allows f to be 'reconstructed' and various operations in the 'time domain' (the t-domain) correspond to certain transformations in the the s-domain. We want to address here those related to integration.

Take the limit $s \downarrow 0$ in (24). This reads as

$$f^*(0+) = \int_0^\infty f(t) dt. \tag{25}$$

Differentiate (24) with respect to s, which we carry out 'under the integral sign', and take s to zero as before to get

$$\left.\frac{df^*(s)}{ds}\right|_{s\downarrow 0} = -\int_0^\infty t f(t) dt. \tag{26}$$

Repeated application of the argument leading to (26) shows that

$$\left.\frac{d^n f^*(s)}{ds^n}\right|_{s\downarrow 0} = (-1)^n \int_0^\infty t^n f(t) dt. \tag{27}$$

[4]This is a deep theorem rooted in Group Theory, established in the 19*th* century by the Norwegian mathematician Abel.

The notation $\mathcal{L}(f(t)|s)$ is also used at times to denote the Laplace transform of f.

Example A.7. (*Moments of the Exponential Distribution.*) Apply (25)–(27) to $f(t) = \lambda e^{-\lambda t}$, the exponential density with rate λ.

```
(%i1) f : lambda * %e^(- lambda * t);
                - t lambda
(%o1) lambda %e
(%i2) lap : laplace(f,t,s);
          lambda
(%o2) ----------
      lambda + s
(%i3) int : limit(lap,s,0,plus);
(%o3) 1
(%i4) mean : limit(-diff(lap,s),s,0,plus);
         1
(%o4) ------
      lambda
(%i5) sndmom : limit(diff(lap,s,2),s,0,plus);
        2
(%o5) -------
              2
      lambda
```

□

Example A.8. (*Complex System, Expected Lifetime.*) We want to illustrate the usefulness of (25) by employing it for finding out information about a system's first time to failure T. Let us take the six-state system from Exercise A.1. Its state space is partitioned into the set of up states $\mathcal{U} = \{1, 2, 3, 4\}$ and the set of down states $\mathcal{D} = \{5, 6\}$. We rearrange the transition rate diagram in Fig. 7 such that the states in $\mathcal{D}$ become absorbing, giving Fig. 8. The event $\{T > t\}$ is equivalent to the system in Fig. 8 being at time t in one of the up states; thus, the system reliability is

$$R(t) = 1 - P(T < t) = 1 - P(\text{system is in } \mathcal{D} \text{ at time } t) = 1 - (p_5(t) + p_6(t)). \quad (28)$$

Take Laplace transforms in (28) to get

$$\mathcal{L}(R(t)|s) = \mathcal{L}(1 - p_5(t) - p_6(t)|s). \quad (29)$$

We know that the integral of the reliability function is the expected system lifetime. Thus, by (25) and (29),

$$E(T) = \int_0^\infty R(t)dt = \lim_{s \downarrow 0} \mathcal{L}(R(t)|s) = \lim_{s \downarrow 0} \mathcal{L}(1 - p_5(t) - p_6(t)|s). \quad (30)$$

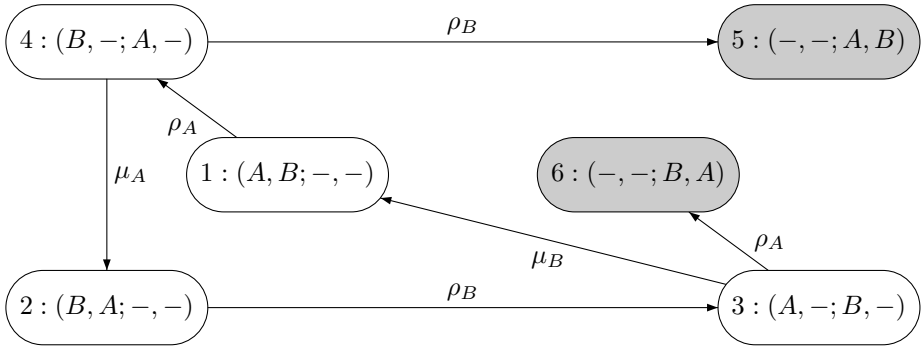

Figure 8. Absorbing Markov model, states in $\mathcal{D}$ are shaded.

The system in Fig 8 is implemented by M.A.14.

Maxima Code M.A.14: *Implementation of the system in Fig. 8*

```
eq1 : diff(p1(t), t) =   - rhoA * p1(t) + muB * p3(t);
eq2 : diff(p2(t), t) =   - rhoB * p2(t) + muA * p4(t);
eq3 : diff(p3(t), t) =   - (muB + rhoA) * p3(t) + rhoB * p2(t);
eq4 : diff(p4(t), t) =   - (muA + rhoB) * p4(t) + rhoA * p1(t);
eq5 : diff(p5(t), t) =   rhoB * p4(t);
eq6 : diff(p6(t), t) =   rhoA * p3(t);
atvalue(p1(t), t = 0, 1);
atvalue(p2(t), t = 0, 0);
atvalue(p3(t), t = 0, 0);
atvalue(p4(t), t = 0, 0);
atvalue(p5(t), t = 0, 0);
atvalue(p6(t), t = 0, 0);
soln : desolve([eq1, eq2, eq3, eq4, eq5, eq6],
               [p1(t), p2(t), p3(t), p4(t), p5(t), p6(t)]);
```

Run the code in M.A.14 and inspect `soln` to see that MAXIMA is unable to provide an explicit solution; the right hand sides of the entries of `soln` will be seen to be in the form `ilt(...)` where `ilt` stands for the inverse Laplace transform which in this case MAXIMA cannot evaluate further.

However, the *expected* system lifetime can be obtained in MAXIMA.

Maxima Code M.A.15: *Implementation of (30)*

```
lap      : laplace(1 - rhs(fifth(soln)) - rhs(sixth(soln)), t, s);
meanlife : limit(lap, s, 0, plus);
```

Running M.A.15, MAXIMA returns for $E(T)$ in `meanlife` the following:

$$\frac{(\rho_A+\mu_B)\,\rho_B^2+\left(\rho_A^2+(\mu_B+2\,\mu_A)\,\rho_A+\mu_A\,\mu_B\right)\,\rho_B+\mu_A\,\rho_A^2+\mu_A\,\mu_B\,\rho_A}{(\rho_A^2+\mu_B\,\rho_A)\,\rho_B^2+\mu_A\,\rho_A^2\,\rho_B}$$

Plausability checks for `meanlife`:

- `meanlife` has the *dimension* [TIME].
- If the repair rates are zero, the system *specializes* to two units being operated in succession, each having an exponential lifetime. The system's mean lifetime is, as expected, $1/\rho_A + 1/\rho_B$:

```
(%i18) e1 : expand(subst([muA = 0, muB = 0], meanlife));
        1      1
(%o18) ---- + ----
       rhoB   rhoA
```

By how much is the present system better than a one-machine system with failure rate ρ and no repair? To find a meaningful answer to this question, we assume that the two machines are nominally identical with failure rates ρ and repair rates $\mu = 10\rho$. Then, MAXIMA tells us that the expected lifetime of the system is twelve times that of a single machine:

```
(%i20) et : subst([muA=10*rho,muB=10*rho,rhoA=rho,rhoB=rho],meanlife);
       12
(%o20) ---
       rho
```

An analogue to (30) is the following formula for the *second* moment of the time to failure.[5]

$$\begin{aligned} E(T^2) &= \int_0^\infty P(T^2>u)du = \int_0^\infty P(T>\sqrt{u})du \\ &= 2\int_0^\infty P(T>t)tdt = 2\int_0^\infty tR(t)dt \\ &= -2\lim_{s\downarrow 0}\frac{d\mathcal{L}(1-p_5(t)-p_6(t)|s)}{ds}. \end{aligned} \tag{31}$$

We find the right hand side of (31) in MAXIMA by

```
(%i21) sndmoment : - 2 * tlimit(diff(lap,s),s,0,plus);[6]
(%o21) ...
```

MAXIMA responds with a large rational expression which we won't show here. Its numerator is found by

[5] To justify (31), use in turn: a substitution with $t = \sqrt{u}$, (26), and (29).

[6] The MAXIMA function `tlimit` may use Taylor series in finding the limit of its first argument. It is like `limit` from the user's point of view.

```
(%i22) num(sndmoment);
```

whereupon MAXIMA responds with

$$
\begin{aligned}
&2\left(\left(\rho_A^2+2\mu_B\rho_A+\mu_B^2\right)\rho_B^4\right.\\
&\quad+\left(\rho_A^3+(2\mu_B+4\mu_A)\rho_A^2+(\mu_B^2+6\mu_A\mu_B)\rho_A+2\mu_A\mu_B^2\right)\rho_B^3\\
&\quad+\left(\rho_A^4+(2\mu_B+4\mu_A)\rho_A^3+(\mu_B^2+9\mu_A\mu_B+3\mu_A^2)\rho_A^2\right.\\
&\qquad\qquad\left.+(4\mu_A\mu_B^2+4\mu_A^2\mu_B)\rho_A+\mu_A^2\mu_B^2\right)\rho_B^2\\
&\quad+\left(2\mu_A\rho_A^4+(5\mu_A\mu_B+2\mu_A^2)\rho_A^3+(3\mu_A\mu_B^2+5\mu_A^2\mu_B)\rho_A^2+2\mu_A^2\mu_B^2\rho_A\right)\rho_B\\
&\quad\left.+\mu_A^2\rho_A^4+2\mu_A^2\mu_B\rho_A^3+\mu_A^2\mu_B^2\rho_A^2\right)
\end{aligned}
\tag{32}
$$

Its denominator is obtained by

```
(%i23) denom(sndmoment);
```

with MAXIMA responding with

$$
\left(\rho_A^4+2\,\mu_B\,\rho_A^3+\mu_B^2\,\rho_A^2\right)\,\rho_B^4+\left(2\,\mu_A\,\rho_A^4+2\,\mu_A\,\mu_B\,\rho_A^3\right)\,\rho_B^3+\mu_A^2\,\rho_A^4\,\rho_B^2 \tag{33}
$$

Plausability checks for `sndmoment`:

- It is seen from (32) and (33) that `sndmoment` has the correct dimension, [TIME^2].
- If the repair rates are zero, the system *specializes*, as before, to the hypo-exponential case. The system's lifetime T has, as expected, the variance $1/\rho_A^2+1/\rho_B^2$:

```
(%i25) e2 : subst([muA = 0, muB = 0], sndmoment)$
(%i26) var : expand(e2 - e1^2);
         1       1
(%o26) ----- + -----
           2       2
       rhoB    rhoA
```

Let us now examine the second moment of the system lifetime under the same conditions as applied earlier when evaluating its first moment:

```
(%i27) e2t : subst([muA=10*rho,muB=10*rho,rhoA=rho,rhoB=rho],sndmoment);
       286
(%o27) ----
          2
       rho
```

Thus, the variance of T is

```
(%i28) var : e2t - et^2;
       142
(%o28) ----
          2
       rho
```

giving an approximate coefficient of variation of 0.993:

```
(%i29) float(sqrt(var)/et);
       0.99303127398442 rho
(%o29) --------------------
              abs(rho)
```

This is a very similar value to that what we would have obtained in the one-machine-no-repairman situation.

Higher moments of T may be obtained in a similar fashion, now using (27).□

ATTILA CSENKI
SCHOOL OF COMPUTING
INFORMATICS AND MEDIA
UNIVERSITY OF BRADFORD
BRADFORD, WEST YORKSHIRE
BD7 1DP, UK

E-mail: a.csenki@bradford.ac.uk

(Received July, 2010)