Analysis of Probabilistic Systems via Generating Functions and Padé Approximation

Michele Boreale[✉]

Università di Firenze, Firenze, Italy
michele.boreale@unifi.it

Abstract. We investigate the use of generating functions in the analysis of discrete Markov chains. Generating functions are introduced as power series whose coefficients are certain hitting probabilities. Being able to compute such functions implies that the calculation of a number of quantities of interest, including absorption probabilities, expected hitting time and number of visits, and variances thereof, becomes straightforward. We show that it is often possible to recover this information, either exactly or within excellent approximation, via the construction of Padé approximations of the involved generating function. The presented algorithms are based on projective methods from linear algebra, which can be made to work with limited computational resources. In particular, only a black-box, on-the-fly access to the transition function is presupposed, and the necessity of storing the whole model is eliminated. A few numerical experiments conducted with this technique give encouraging results.

1 Introduction

Our goal is to understand if the concept of generating function [9] can play a useful role in the analysis of Markov chains. In the present paper, we focus on the reachability properties of time-homogeneous, finite Markov chains. The generating function of such a system is a power series in the variable z, $g(z) = \sum_{j\geq 0} a_j z^j$, whose coefficients, or *moments*, a_j are just the probabilities of hitting a state of interest exactly at time $j = 0, 1, 2, \cdots$. With $g(z)$, a whole host of information about the system is packed into a single mathematical object, including: the probability of the event itself - which is of course $g(1)$ - and various statistics, such as the expected hitting time and its variance. We will demonstrate that, by building a rational representation of $g(z)$, in a number of interesting situations it is possible to extract this information, either exactly or within excellent approximation, using limited computational resources. These limitations are mainly the fact that one can access the system's transition relation only in a black-box, on-the-fly[1] fashion, and can only store a small portion of its state space at time. We give a more detailed account of our approach and of our paper below.

Author's address: Michele Boreale, Università di Firenze, Dipartimento di Statistica, Informatica, Applicazioni (DiSIA) "G. Parenti", Viale Morgagni 65, I-50134 Firenze, Italy. E-mail: michele.boreale@unifi.it. Work partially supported by MIUR funded project CINA.

[1] That is, via a function that given a state returns the list of its successors together with their probabilities.

M.M. Halldórsson et al. (Eds.): ICALP 2015, Part II, LNCS 9135, pp. 82–94, 2015.
DOI: 10.1007/978-3-662-47666-6_7

We first introduce and motivate the system's generating function $g(z)$, then establish some of its important properties, like its radius convergence and its rationality (Section 2). We then show (Section 3) that, via *Padé approximants* [3], an *exact* rational representation of $g(z)$ can be recovered from the knowledge of its first $2N$ moments, where N is the number of system's states. These moments can in principle be computed relying solely on a black-box, on-the-fly access to the transition relation, and do not require storing the entire model. Yet, limited resources imply that one is often forced to consider approximations. We argue (Section 4) that polynomial approximations, derived from truncating $g(z)$, may not be a good idea, and that rational ones should rather be preferred. This is especially true in the presence of clusters of states that are nearly uncoupled with other states in the chain. We then discuss a method to compute one such approximation effectively (Section 5). The basic idea here is to view the matrix P that represents the transition relation as a linear application on $\mathbb{R}^N$; then to take its projection $\hat{P}$ onto a small, m-dimensional subspace $\mathcal{K}_m$, with $m \ll N$. The generating function of the projected application, $\hat{g}(z)$, is a rational *Padé-type* approximation of the original $g(z)$. Accuracy is often very good already for small m: this somewhat surprising effectiveness is a consequence of the tendency of $\hat{P}$'s eigenvalues to be excellent approximations of P's ones. We show that the Arnoldi algorithm [2,8] can be used to effectively compute $\hat{g}(z)$, in a way that is compatible with an on-the-fly access to the transition relation and with limited computational resources. Notably, transitions need not be stored at all with this method. Error control and steady-state distributions are discussed in the full version [4]. We then present a few numerical experiments that have been conducted with a preliminary Matlab implementation of this idea (Section 6), and which give very encouraging results. For comparison, the results obtained on the same systems with a state-of-the-art probabilistic model checker are also reported. We conclude the paper with a discussion of future venues of research and related work (Section 7). All the proofs, some numerical examples and additional technical material can be found in the full version available online [4].

2 The System Generating Function $g(z)$

Consider a time-homogeneous Markov chain $\{X_j\}_{j\geq 0}$ over a finite set of states $\mathcal{S} = \{1, ..., N\}$ with $N > 0$ and initial state $X_0 = 1$. To avoid uninteresting special cases, we will assume that all states of the chain are reachable from 1 (but need not assume the vice-versa, so the chain might well be reducible.) We want to study the event corresponding to reaching a (typically, 'bad') state $s_{bad} = N$, that is $Reach \stackrel{\triangle}{=} \{X_j = N \text{ for some } j \geq 0\}$ and denote by $p_{reach} \stackrel{\triangle}{=} \Pr(Reach)$ the probability of this event. Without loss of generality, we will assume that state N is absorbing that is $\Pr(X_{j+1} = N | X_j = N) = 1$. Later on in this section we will also consider another type of statistics, concerning the number of visits, where we will not assume N is absorbing. We will also be interested in statistics concerning the *hitting time* random variable, defined as: $T \stackrel{\triangle}{=} \inf\{j \geq 0 : X_j = N\}$. Let us define the j-th ($j \geq 0$) *moment* of the system as the probability of hitting state N for the first time exactly at time j, that is

$$a_j \triangleq \Pr(X_j = N \text{ and } X_i \neq N \text{ for } i < j)\,. \tag{1}$$

Clearly, the probability of eventually reaching N is just the sum of the moments: $p_{reach} = \sum_{j\geq 0} a_j$. Our main object of study is defined below.

Definition 1 (Generating function). *The generating function of the system is the power series, in the complex variable z, $g(z) \triangleq \sum_{j\geq 0} a_j z^j$.*

Note that the $g(z)/p_{reach}$ is just the probability generating function of the random variable T conditioned on the event *Reach*. As such, with $g(z)$ a whole host of information about T and its moments is packed into a single mathematical object. For instance, indicating with $g', g'', ...$ the derivatives of g, easy calculations show that (provided the mentioned quantities are all defined).

$$p_{reach} = g(1) \qquad E[T|Reach] = g'(1)/g(1) \qquad \text{var}[T|Reach] = (g''(1) + g'(1))\,/g(1) - (g'(1)/g(1))^2\,. \tag{2}$$

More generally, information on higher moments of T can be extracted using the identity $E[X(X-1)\cdots(X-k)|Reach] = g^{(k+1)}(1)/g(1)$, although usually the first two moments are enough for a satisfactory analysis of the system. In practice, we will be able to extract this information only provided we are able to build an efficient representation of $g(z)$. As a first step towards this, we shall see in a moment that $g(z)$ can be represented as a rational function, that is, as the ratio of two polynomials in z. At this point it is convenient to introduce some notation.

Notation In the rest of the paper, some basic knowledge of linear algebra is presupposed. We let P denote the $N \times N$ stochastic matrix that defines the transition function of the chain. Departing from the usual convention, we will work with *column*-stochastic matrices, that is we let the element of row i and column j of P, denoted by p_{ij}, be $\Pr(X_{t+1} = i | X_t = j)$. In what follows, vectors are considered as column-vectors; in particular, e_i denotes the i-th canonical column vector of $\mathbb{R}^N$. So the vector $P^i e_1$ is just the probability distribution of the variable X_i of the chain. A vector is stochastic if its components are nonnegative and sum to 1. For a matrix or vector A, we let A^T denote its transpose. I_k will denote the $k \times k$ identity matrix; the index k will be omitted when clear from the context. A rational function in z of type $[h, k]$, for $k \geq 0$ and $h \geq 0$ or $h = -\infty$, is a ratio of two polynomials in z, $r(z)/t(z)$, such that $\deg(r) \leq h$ and $\deg(t) \leq k$. We will let z range over complex numbers and x on reals.

In the rest of the paper, we let $\tilde{e}_1 \triangleq (P - I)e_1$. Note that $e_N^T P^j e_1$ is just the probability of being in state N at time j. Exploiting the fact that N is absorbing, it is easy to see that, for $j \geq 1$, $a_j = e_N^T P^j e_1 - e_N^T P^{j-1} e_1 \;=\; e_N^T P^{j-1} \tilde{e}_1$. Ignoring for a moment issues of convergence and singularity, we can then reason as follows. We first note that the following equality can be readily checked.

$$(I - zP)(I + zP^1 + z^2P^2 + \cdots) = I$$

which implies that $(I - zP)^{-1} = (I + zP^1 + z^2P^2 + \cdots)$. We then can write

$$\begin{aligned} g(z) &= a_0 + \sum_{j\geq 1} a_j z^j & &= a_0 + \sum_{j\geq 1} e_N^T z^j P^{j-1}\tilde{e}_1 \\ &= a_0 + z\cdot e_N^T(\sum_{j\geq 0} z^j P^j)\tilde{e}_1 & &= a_0 + z\cdot e_N^T(I-zP)^{-1}\tilde{e}_1 \\ &= a_0 + \frac{z\cdot e_N^T \mathrm{Adj}(I-zP)\tilde{e}_1}{\det(I-zP)} \end{aligned} \tag{3}$$

where, in the last step, we have exploited Cramer's rule for the computation of the inverse (recall that Adj(A) denotes the *adjoint matrix* of a matrix A.) In the last expression, the denominator and numerator of the fraction are polynomials in z. This shows that $g(z)$ is a rational function. This informal reasoning can be made into a rigorous proof – a nontrivial point of which is related to the singularity of the matrix $(I - zP)$ at $z = 1$. Another important point is where the power series $g(z)$ is defined, that is, what is its radius of convergence. The the next theorem records these facts about $g(z)$.

Theorem 1 (convergence and rationality of g**).** *There is a real $R > 1$ such that the power series $g(z)$ in Definition 1 converges for all $|z| < R$. Moreover, there is a rational function $r(z)/t(z)$ such that for all such z's*

$$\textit{(a)}\;\; t(z) \neq 0 \qquad \textit{(b)}\;\; \deg(r), \deg(t) \leq N-1 \qquad \textit{(c)}\;\; g(z) = r(z)/t(z)\,. \tag{4}$$

The proof of the above theorem provides us also with an explicit expression for $g(z)$, that is (3). In what follows, $(I - zP)^{-1}$ will be used as an abbreviation for the matrix of rational expressions $\frac{\mathrm{Adj}(I-zP)}{\det(I-zP)}$, where it is understood that common factors are canceled out. Concerning this expression, note that $\det(I - zP) = z^N \det((1/z)I - P)$ is just the characteristic polynomial of P with coefficients reversed. That is, the relation between the characteristic polynomial and our $\det(I - zP)$ is as follows:

$$\begin{aligned} \det(zI-P) &= \beta_N z^N + \beta_{N-1}z^{N-1} + \cdots + \beta_0 \qquad \text{and} \\ \det(I-zP) &= \beta_N + \beta_{N-1}z + \cdots + \beta_0 z^N \end{aligned}$$

(In passing, note that $\beta_N = 1$ and $\beta_0 = -\det(P)$.) In particular, from $\det(I - zP) = z^N \det((1/z)I - P)$ it is clear that the roots of the polynomial $\det(I - zP)$ are just the reciprocals of the nonzero roots of P: that is, the reciprocals of the nonzero eigenvalues of P. This fact can be exploited to give more precise information about R. We record these facts below.

Corollary 1. *There is $R > 1$ such that for $|z| < R$ and for $\tilde{e}_1 = (P - I)e_1$*

$$g(z) = a_0 + z\cdot e_N^T(I-zP)^{-1}\tilde{e}_1\,. \tag{5}$$

In particular, $g(z)$ has a radius of convergence either $R = |1/\lambda|$ for some eigenvalue $0 < |\lambda| < 1$ of P, or $R = +\infty$.

Let us now drop the assumption that N is absorbing. We are interested in counting the visits to state N, starting from $X_0 = 1$. To this purpose we introduce a different generating function: $f(z) \triangleq \sum_{j\geq 0} c_j z^j$, where $c_j \triangleq \Pr(X_j = N) = e_N^T P^j e_1$. By definition,

$f(1)$ is the expected number of visits of the chain to state N. Recall that $f(1) < +\infty$ iff N is transient. By paralleling the above development for $g(z)$, we can prove the following.

Theorem 2. *Let N be transient. The power series $f(z)$ has radius of convergence $R > 1$. Moreover, for $|z| < R$, one has $f(z) = e_N^T(I - zP)^{-1}e_1$. The expression on the right of the last equality gives rise to a rational function $r(z)/t(z)$ of type $[N-1, N]$ such that $t(z) \neq 0$ for $|z| < R$.*

From now on, we will consider $g(z)$ only; statements and proofs for $f(z)$ can be obtained by obvious modifications. All the quantities of interest about the system, (2), will be easy to compute, provided we can recover the rational representation $r(z)/t(z)$ of $g(z)$ promised by Theorem 1. The expression provided by Corollary 1 can be useful for small values of N and provided one knows P explicitly. Here is a small example to illustrate.

Example 1. We consider a chain with $N \geq 3$ states 1,2,..., N, where, for $1 \leq i \leq N-3$ and a (small) $0 < \delta < 1$, there is a transition from i to 1 with probability $1 - \delta/i$, and to each of $i+1, N-1, N$ with probability $\delta/3i$; for $i = N-2$, there is a transition from i to 1 with probability $1-\delta/i$, and to each of $N-1, N$ with probability $\delta/2i$; $N-1$ and N are absorbing. For reasons that will become evident later on, we call this chain $Nasty(N, \delta)$.

The transition matrix P of $Nasty(6, \delta)$ is given on the right (recall that we work with column-stochastic matrices.) From symmetry considerations, it is clear that the probability of reaching either of the two absorbing states is 1/2, thus $p_{reach} = 1/2$. Let us check this out via $g(z)$. With the help of a computer algebra system, we apply (5) and, taking into account that $a_0 = 0$, find:

$$\begin{bmatrix} 1-\delta & 1-\delta/2 & 1-\delta/3 & 1-\delta/4 & 0 & 0 \\ \delta/3 & 0 & 0 & 0 & 0 & 0 \\ 0 & \delta/6 & 0 & 0 & 0 & 0 \\ 0 & 0 & \delta/9 & 0 & 0 & 0 \\ \delta/3 & \delta/6 & \delta/9 & \delta/8 & 1 & 0 \\ \delta/3 & \delta/6 & \delta/9 & \delta/8 & 0 & 1 \end{bmatrix}$$

$$g(z) = \frac{1}{2}\,\frac{\delta^4 z^4 + 8\delta^3 z^3 + 72\delta^2 z^2 + 432\delta z}{(\delta^4 - 4\delta^3)z^4 + (12\delta^3 - 36\delta^2)z^3 + (108\delta^2 - 216\delta)z^2 + 648(\delta - 1)z + 648}\,.$$

When evaluating this at $z = 1$ we get $g(1) = p_{reach} = 1/2$. By differentiating the above expression of $g(z)$ and then evaluating the result at $z = 1$, we get $g'(1) = 2\left(\delta^3 + 9\delta^2 + 54\delta + 162\right) / \left(\delta^4 + 8\delta^3 + 72\delta^2 + 432\delta\right)$, which can be evaluated for instance at $\delta = 10^{-3}$ to compute $E[T|Reach] = g(1)^{-1}g'(1) \approx 1500.25$.

Computing an expression for $g(z)$ based on a direct application of Corollary 1 requires the explicit knowledge of the matrix P. Moreover, the computation relies on costly symbolic operations involving matrices whose entries are rational functions in z, rather than scalars. For these reasons, this method can only be practical for small values of N. The next section explains how to numerically calculate a rational representation of $g(z)$ out of the first $2N$ moments of the system, without having to know P explicitly.

3 Exact Reconstruction of $g(z)$

We first review Padé approximants and then explain how to employ them to exactly reconstruct $g(z)$. The exposition of Padé approximants in this section is standard. For an in depth treatment, see e.g. [3]. Let $f(z) = \sum_{i\geq 0} c_i z^i$ be a generic a power series in the complex variable z, with a nonzero radius of convergence. For any $n \geq 0$, we let the truncation of f at the n-th term be the polynomial $f_n(z) \overset{\triangle}{=} \sum_{i=0}^{n} c_i z^i$. Let us indicate by $o(z^k)$ a generic power series divisible by z^k. Given two power series $f(z)$ and $d(z)$ we write $f(z) = d(z) \bmod z^{n+1}$ iff $f_n = d_n$, or, in other words, if $f(z) - d(z) = o(z^{n+1})$. Polynomials are of course considered as power series with only finitely many nonzero coefficients.

Definition 2 (Padé approximants). *Let $f(z) = \sum_{i\geq 0} c_i z^i$ be a power series in the complex variable z with a nonzero radius of convergence. Given integers $h, k \geq 0$, we say a pair of polynomials in z, say (r, t), is a $[h, k]$-*Padé approximant of $f(z)$ *if the following holds true, where $n = h + k$.*

$$\textbf{\textit{(a)}}\ z \nmid t(z) \quad \textbf{\textit{(b)}}\ \deg(r) \leq h,\ \deg(t) \leq k \quad \textbf{\textit{(c)}}\ f(z)t(z) = r(z) \bmod z^{n+1}\,. \tag{6}$$

Seen as a *real* function $r(x)/t(x)$, a Padé approximant is a rational approximation of the function $f(x)$, up to and including the term of degree n of its Taylor of expansion. To see this, first note that equation (6)(c) is equivalent to saying that there exists a power series $k(x)$ such that $f(x)t(x) = r(x) + k(x)x^{n+1}$. As $t(0) \neq 0$, the last equation is equivalent to saying that, in a neighborhood of 0: $f(x) = r(x)/t(x) + x^{n+1}k(x)/t(x)$. This equation is equivalent to saying that the Taylor expansion of $r(x)/t(x)$ from $x = 0$, truncated at the n-th term, coincides with[2] $f_n(x)$, that is: $r(x)/t(x) = \sum_{i=0}^{n} c_i x^i + o(x^{n+1})$. In case f is itself a rational function, Padé approximants provide us with a method to actually find an *exact* representation of it, given only sufficiently many coefficients c_i, as we will see shortly. We first state a result about uniqueness of Padé approximants; its proof follows from easy manipulations on rational functions (or see [3, Th.1.1].)

Proposition 1. *Let (r, t) and (p, q) be two $[h, k]$-Padé approximants of f. Then they are the same as a function, in the sense that $r(z)/t(z) = p(z)/q(z)$.*

Given any power series $f(z)$, it is possible to compute a $[h, k]$-Padé approximant of it as follows. Here we assume for simplicity that $h \leq k$; the case $h > k$ does not interest us, and can be anyway treated with minor notational changes. Also, in view of condition (6)(a) of the definition of Padé approximant, without loss of generality we will restrict ourselves to the case where $t(0) = 1$, that is, the constant coefficient of t is always taken to be 1. Assume $n + 1$ ($n = h + k$) coefficients $c_0, c_1, ..., c_n$ of f are given. Arrange the coefficients from h through $h+k-1 = n-1$ to form a $k \times k$ matrix C as described next,

[2] To see this, observe that, for $0 \leq j \leq n$, by equating the j-th derivatives of the left and right hand side of (6)(c), one obtains that $f^{(j)}(x) = (r(x)/t(x))^{(j)} + o(x^{n-j+1})$, so that $c_j = f^{(j)}(0)/j! = (1/j!)(r(0)/t(0))^{(j)}$. Also note that the Taylor expansion of $r(x)/t(x)$ at $x = 0$ exists, as $t(0) \neq 0$.

where $c_l \triangleq 0$ for indices $l < 0$. Let $r(z) = \alpha_h z^h + \cdots + \alpha_0$ and $t(z) = \beta_k z^k + \cdots + \beta_1 z + 1$ be two polynomials, for generic vectors of coefficients $\alpha = (\alpha_0, ..., \alpha_h)^T$ and $\beta = (\beta_1, ..., \beta_k)^T$. Assume (r, t) is a $[h, k]$-Padé approximant of f. Then we can equate coefficients of like powers on the left- and right-hand side of (6)(c).

$$C = \begin{bmatrix} c_h & c_{h-1} & \cdots & c_{h-k+1} \\ c_{h+1} & c_h & \cdots & c_{h-k+2} \\ & & \vdots & \\ c_{h+k-1} & c_{h+k-2} & \cdots & c_h \end{bmatrix}.$$

In particular, coefficients from $h+1$ through $n = h+k$ are 0 on the right, thus coefficients on the left must satisfy the following, for $\gamma \triangleq (-c_{h+1}, ..., -c_{h+k})^T$:

$$C\beta = \gamma\,. \tag{7}$$

On the other hand, assume (7), seen as a system of equations in the unknowns β, has a solution. Then by taking α given by:

$$\alpha = \tilde{C}\beta' \tag{8}$$

$$\tilde{C} = \begin{bmatrix} c_0 & & & \\ c_1 & c_0 & & \\ & & \vdots & \\ c_h & c_{h-1} & \cdots & c_1 \; c_0 \end{bmatrix}.$$

where $\beta' = (1, \beta_1, ..., \beta_h)^T$ and $\tilde{C}$ is the $(h+1) \times (h+1)$ lower-triangular matrix $\tilde{C}$ given on the right. We see that (6)(c) is satisfied by (r, t).

Of course also (6)(a) and (6)(b) are satisfied. Therefore (r, t), as given by α and β, is a $[h, k]$-Padé approximant of f. In other words, we have shown the following.

Proposition 2. *A $[h, k]$-Padé approximant for f exists if and only if the system of equations* (7) *in the unknowns β has a solution. If it exists, the coefficients β and α for t and r are given, respectively, by a solution of* (7) *and by* (8).

Note that the procedure outlined above to reconstruct a $[h, k]$-Padé approximant takes $O(k^3 + h^2) = O(n^3)$ operations and $O(n^2)$ storage. Let us now come back to our Markov chain. Theorem 1 and Proposition 1 ensure that $g(z)$ coincides, as a function, with its $[N-1, N-1]$-Padé approximant. Using the above outlined method, one can reconstruct the rational form of $g(z)$, provided one knows (an upper bound on) N and the coefficients $a_0, ..., a_{2N-2}$. The latter can in principle be computed by a power iteration method: $a_0 = e_N^T e_1$ and $a_i = e_N^T P^{i-1} \tilde{e}_1$ for $i \geq 1$. For this, it is sufficient to obtain a black-box access to the function $u \mapsto Pu$, which is compatible with an on-the-fly implementation. A numerical example illustrating the method is reported in the full paper [4].

While dispensing with symbolic computations and the explicit knowledge of P, the method described in this section still suffers from drawbacks that confine its application to small-to-moderate values of N. Indeed, the solution of the linear system (7) has a time complexity of $O(N^3)$. Although this can be improved using known techniques from numerical linear algebra (see [4]), the real problem here is that the explicit computation of $2N$ moments a_j is in practice very costly and numerically instable, and should be avoided. Moreover, it is clear that a consistent gain in efficiency can only be obtained by accepting some degree of approximation in the computed solution.

4 Discussion: Approximating $g(z)$

Suppose that, possibly due to limited computational resources, we have access only to a limited number, say $m + 1$, of g's moments a_i. Or, more generally, we can access the transition relation of the Markov chain - the mapping $u \mapsto Pu$ - only a limited number m of times. Typically, we can only afford $m \ll N$. The resulting information may be not sufficient to recover an exact representation of $g(z)$. Then the problem becomes finding a a good approximating function $\hat{g}(z)$ of $g(z)$. "Good" here means at least that $g(z) - \hat{g}(z) = o(z^{m+1})$; but, even more than that, $\hat{g}(z)$ should approximate well $g(z)$ near $z = 1$, as we are mainly interested in evaluating $g(1), g'(1)$ and so on, see (2). The first, obvious attempt is to consider a polynomial approximation: $\hat{g}(z) = g_m(z) = \sum_{i=0}^{m} a_i z^i$. This is the truncation of $g(z)$ at the term of degree m.

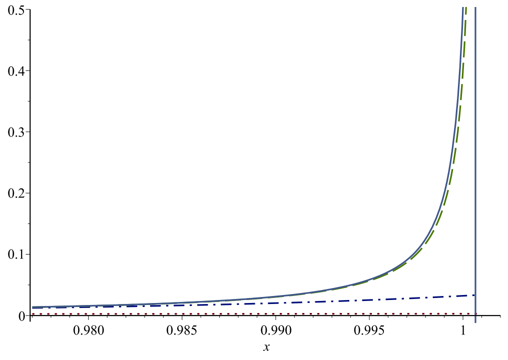

Fig. 1. Plots of $g_{10}(x)$ (dotted), of $g_{100}(x)$ (dash-dotted), of the [1, 1]-Padé approximant $\hat{g}(x) = 2x/(-5995x + 6000)$ (dashed) and of $g(x)$ (solid), for $Nasty(6, 10^{-3})$, near $x = 1$. Here $g(z)$ has a pole at $z = 1/\lambda$ with $\lambda \approx 0.9993$.

Unfortunately, such a $\hat{g}(z)$ might be a very bad approximation of $g(z)$. The reason is that the rational representation $r(z)/t(z)$ of $g(z)$ may have a pole near[3] $z = 1$: that is, there can be a $z_0 \in \mathbb{C}$ such that $|z_0 - 1| \approx 0$ and $\lim_{z \to z_0} |r(z)/t(z)| = +\infty$. Then, as z approaches 1 from its convergence zone, $g(z)$ becomes extremely fast growing, and essentially impossible to approximate by means of a polynomial function, as polynomials have no finite poles.

As stated by Corollary 1, the pole of smallest modulus of $g(z)$, which determines its radius of convergence R, is of the form $z_0 = 1/\lambda$, for some subdominant eigenvalue λ of P, that is an eigenvalue with $|\lambda| < 1$. If 1 is "badly separated" from λ, that is if $|\lambda - 1| \approx 0$, the truncated sums $\sum_{i \le m} a_i$ will converge very slowly to p_{reach}, as m grows. In this respect, a rational approximation $\hat{g}(z) = \frac{\hat{r}(z)}{\hat{t}(z)}$ can perform much better. Indeed, $\hat{t}(z)$ can be chosen so as to have a root near z_0. This in essence is what Padé approximation achieves. When building a $[h, k]$-Padé approximant with $h + k \le m$, the same amount of information used to build the polynomial $g_m(z)$ above - the first $m + 1$ moments of $g(z)$ - is used to "guess" an approximation of z_0, that becomes a root of $\hat{t}(z)$ (this aspect will be further discussed in the next section, see Remark 1.) The benefit of rational over polynomial approximation is qualitatively illustrated by the plots in Fig. 1.

[3] Note that the rational function $r(z)/t(z)$, while coinciding with $g(z)$ within the disk $|z| < R$, will also be defined outside this disk.

It is well-known [5] that that the bad separation phenomenon (subdominant eigenvalues close to 1) occurs if there is a cluster of states that are strongly coupled with one another, but nearly uncoupled with other states in the chain, like $1, ..., n-2$ in $Nasty(n, \delta)$ (see [4].) In the next section we explore an effective way of building rational approximations $\hat{g}(z) = \hat{r}(z)/\hat{t}(z)$.

5 Approximation of $g(z)$ via a Projection Method

The general idea of a projection method is as follows. Consider P as a linear map acting on the N-dimensional space $\mathbb{R}^N$. We identify a m-dimensional subspace, $\mathcal{K}_m$, and then consider the projection of P onto this space, say $\hat{P}$: this is our low-dimensional approximation of the original system. Here, $m \ll N$: practically m will be of the order of tens or hundreds. Formally, consider an integer $m \geq 1$ and the *Krylov subspace* of $\mathbb{R}^N$

$$\mathcal{K}_m(P, \tilde{e}_1) \triangleq \text{span}\{\tilde{e}_1, P\tilde{e}_1, P^2\tilde{e}_1, ..., P^{m-1}\tilde{e}_1\}$$

abbreviated as $\mathcal{K}_m$ in what follows. Now take any orthonormal basis of $\mathcal{K}_m$ and arrange the corresponding column vectors into a $N \times m$ matrix, $V_m = [v_1, ..., v_m]$. Note that orthonormality means that $V_m^T V_m = I_m$. We can consider the projection of P, seen as an application $\mathbb{R}^N \to \mathbb{R}^N$, onto $\mathcal{K}_m$. The representation of this application restricted to $\mathcal{K}_m$, in the basis V_m, is given by the $m \times m$ matrix

$$H_m = V_m^T P V_m \,. \tag{9}$$

(The matrix H_m will play the role played by $\hat{P}$ in the above informal description.) Note that if m is large enough then $\mathcal{K}_m$ will be a P-invariant subspace of $\mathbb{R}^N$, that is $P\mathcal{K}_m \subseteq \mathcal{K}_m$[4].

Theorem 3. *Let $m \geq 1$. Consider the function, defined in a neighborhood of the origin*

$$\hat{g}(z) \triangleq a_0 + z \cdot (e_N^T V_m)(I_m - zH_m)^{-1}(V_m^T \tilde{e}_1)\,. \tag{10}$$

Then $\hat{g}(z)$ is a rational function of type $[m, m]$ and $g(z) - \hat{g}(z) = o(z^{m+1})$. Moreover, if 1 is not an eigenvalue of H_m, then $\hat{g}(z)$ is defined in a neighborhood of $z = 1$. Finally, if $\mathcal{K}_m$ is P-invariant, then $g(z) = \hat{g}(z)$.

Remark 1. Note from (10) that, while $\hat{g}(z)$ is a rational function of type $[m, m]$, it is not guaranteed that $g(z) - \hat{g}(z) = o(z^{2m+1})$. Thus $\hat{g}(z)$ is not, in general, a Padé approximant, but only a weaker *Padé-type* approximant. Comparing (5) and (10), we further see that how well $\hat{g}(z)$ approximates $g(z)$ depends on how well the polynomial $\det(I_m - zH_m)$ approximates the polynomial $\det(I - zP)$. We have already noted that the roots of these polynomials are the reciprocals of nonzero eigenvalues of H_m and P, respectively. It is known for general matrices P that, already for small values of m, H_m's eigenvalues - known as *Ritz values* in the literature - tend to be excellent approximations of the eigenvalues of P that are at the extreme of the spectrum, that is, those of either large or small modulus. The details and nature of such approximation are not yet fully understood,

[4] In particular, it is sufficient to take any $m \geq \nu$, where $\nu \leq N$ is the degree of the minimal polynomial of P, that is, the monic polynomial p of minimal degree such that $p(P) = 0_{N\times N}$: this is a consequence of the Cayley-Hamilton theorem.

Algorithm 1. Arnoldi based calculation of $\hat{g}(z)$

```
    Input: $m \geq 1$; a black-box mechanism for computing the function $u \mapsto Pu$
    Output: a triple $(a_0, V_m, H_m)$
 1:    $a_0 = e_N^T e_1$
 2:    $v_1 = \tilde{e}_1 / \|\tilde{e}_1\|_2$
 3:    for $j = 1, 2, ..., m$ do
 4:        for $i = 1, 2, ..., j$ do $h_{ij} = (Pv_j, v_i)$
 5:        $w_j = Pv_j - \sum_{i=1}^{j} h_{ij} v_i$
 6:        $h_{j+1,j} = \|w_j\|_2$
 7:        if $(h_{j+1,j} = 0) \vee (j = m)$ then break else $v_{j+1} = w_j / h_{j+1,j}$
 8:    $m = j$
 9:    $V_m = [v_1, ..., v_m]$
10:    return $(a_0, V_m, H_m)$
```

but in the specialized literature there is abundant experimental and theoretical evidence about it, see e.g. [8]. In any case, this fact retrospectively sheds light on the somewhat surprising effectiveness of Padé approximation.

Formulae for the derivatives $\hat{g}^{(j)}(x) = \frac{d^j}{dx^j}\hat{g}(x)$ follow from (10) by the familiar rules of derivation, see [4]. We now show that the matrices V_m and H_m can be computed via an effective and numerically stable procedure known as the *Arnoldi process* [2,8]. Arnoldi does not require knowledge of the full matrix P, but only a black-box access to the matrix-vector multiplication function $u \mapsto Pu$, which makes it compatible with an on-the-fly approach. Algorithm 1 works incrementally and, for $j = 1, ..., m$, builds an orthormal basis of $\mathcal{K}_j$, $V_j = [v_1, ..., v_j]$, and the corresponding projected version of P onto $\mathcal{K}_j$, H_j. The next vector v_{j+1} is built by orthonormalizing Pv_j against the available basis V_j (lines 4–7, which are essentially the Gram-Schmidt orthonormalization.) If this process results in the null vector ($h_{j+1,j} = 0$), then Pv_j is linearly dependent from vectors in V_j, thus the space $\mathcal{K}_j$ is P-invariant, and the main iteration stops.

The algorithm makes use of the following variables, for $j = 1, ..., m$: the scalars $a_0, a_j \in \mathbb{R}$; vectors $v_j, w_j \in \mathbb{R}^N$; the matrix $H_m \in \mathbb{R}^{m \times m}$ whose nonzero elements are the reals $h_{l,l'}$ for $1 \leq l \leq l' + 1$ and $l \leq l' \leq m$; the matrix $V_m \in \mathbb{R}^{N \times m}$. In line 4, $(\cdot, \cdot)$ denotes inner product. Of course, Pv_j needs to be computed only once per each j-iteration. Nesting of blocks is defined by indentation. The algorithm can take advantage of a sparse storage scheme. In what follows, we let W be the maximal number of nonzero elements in $P^j \tilde{e}_1$, for any $0 \leq j \leq m$, and B the maximal number of outgoing transitions from any state. Note that $W \cdot B$ is upper bounded by the overall number of transitions. Recall that a square matrix is in upper *Hessenberg* form if all its entries below the main subdiagonal are 0.

Theorem 4. *Let $m \geq 1$ and let (a_0, V_m, H_m) be the output returned by Algorithm 1. Then V_m is an orthonormal basis of $\mathcal{K}_m$ and (9) is satisfied. As a consequence, $\hat{g}(z)$ satisfies (10) given this choice of a_0, V_m, H_m. Moreover, H_m is in upper Hessenberg form and if $h_{m+1,m} = 0$ then $\hat{g}(z) = g(z)$. Assuming $u \mapsto Pu$ can be computed in $O(WB)$ operations* *and $O(W)$ storage, the algorithm takes $O(mWB)$ operations and $O(mW)$ storage to complete.*

A numerical example illustrating Algorithm 1 is reported in the full version [4]. Note that in principle we can calculate $\hat{g}(1)$, and more generally $\hat{g}(x)$ whenever defined for x, directly using the definition (10). However, it is computationally much better to proceed as follows: $\hat{g}(x) = a_0 + e_N^T V_m y$, where y is the (unique) solution of the system $(I_m - xH_m)y = V_m^T \tilde{e}_1 = e_1^{(m)} \cdot ||\tilde{e}_1||_2$ (here $e_1^{(m)}$ is the first canonical vector of $\mathbb{R}^m$.) Since $(I_m - xH_m)$ is still quasi-triangular (upper Hessenberg), the system above can be solved with $O(m^2)$. The derivatives of g can be computed similarly, see [4]. In the end, via the Arnoldi algorithm 1 (cost $O(mWB)$), we have reduced the computation of all the important properties of the system to the resolution of small quasi-triangular systems, which cost approximately $O(m^2)$, for a small, "affordable" m. Computation of steady-state probabilities and of error control is also discussed in [4].

6 Experiments

We have put an on-the-fly implementation (in Matlab) of Algorithm 1 at work on a few simple probabilistic systems. With two exceptions, the chosen systems are relatively small, but, due to the presence of nearly uncoupled strongly connected components, they exhibit the bad separation phenomenon discussed in Section 4. In each case, the reachability probability of interest is easy to compute analytically, due to the symmetry of the system. The Matlab implementation, as well as the Matlab and Prism specifications of the considered examples, are available at [4]. We give below an informal description of these systems.

The $Nasty(n, \delta)$ systems have been introduced in Example 1; here we have fixed $\delta = 10^{-3}$ and considered $n = 10^5, 10^6$. A $Queue(n)$ system consists of n queueing processes, each of capacity four, running in parallel. At each time, either an enqueue (prob. 0.1) or a dequeue (prob. 0.9) request arrives, and is served by any process that is able to serve it. In case of global overflow, each process chooses either of two indeterminate error states, and remains there forever. This gives rise to 2^n possible overflow configurations, which are absorbing. The event of interest is that the system eventually reaches one specific such configuration; here the cases $n = 2, 3$ are considered. An $Ising(n)$ system consists of n particles, each of which can take on, at each time, either the *up* or the *down* spin value, with a probability depending on how many *up*'s are in system and on a temperature parameter. The *all-up* and *all-down* configurations are absorbing, and the event of interest is that all-up is eventually reached, starting from the equilibrium configuration with $n/2$ particles down and $n/2$ up; here, the cases $n = 6, 8$ are considered. In *Chemical*(n), a solution is initially composed by $n/2$ reactant pairs of type 1 and $n/2$ reactant pairs of type 2. A reactant pair of one type can get transformed into a pair of the other type, according to a chemical rule obeying the law of mass action – the probability that the reaction occurs depends on the concentration of the reactants. A solution consisting of reactants pairs all of the same type is absorbing. The probability of eventually reaching one specific such solution is seeked for; here, the cases $n = 24, 26$ are considered.

Table 1. Results of some experiments. N = number of states; p_{reach} = exact reachability probability. $\hat{p}_{reach}$ = probability returned by the algorithms, truncated at the 4-th digit after decimal point; $\%error = 100 \times |p_{reach} - \hat{p}_{reach}|/p_{reach}$ is the relative error percentage; m = number of iterations; $time$ = time in seconds. For Prism, default options have been employed except that for m (tweaking other options did not lead to significant improvements); model building time is included in $time$.

System			**Algorithm 1**				**Prism**			
Name	N	p_{reach}	$\hat{p}_{reach}$	$\%error$	m	$time$	$\hat{p}_{reach}$	$\%error$	m	$time$
Nasty	10^5	0.5	0.5000	$< 10^{-11}$	5	0.08	0.4999	$< 10^{-4}$	27	5524
	10^6	0.5	0.5000	$< 10^{-10}$	5	0.06	-	-	-	-
Queue	49	0.25	0.2500	$< 10^{-5}$	15	0.92	0.2454	1.81	133176	0.34
	231	0.125	0.1248	0.11	37	11.21	0.0013	98.91	1983765	22.63
Ising	64	0.5	0.5000	$< 10^{-3}$	9	2.06	0.4982	0.34	26433	0.05
	256	0.5	0.4998	<0.10	18	20.58	0.0578	88.42	1257073	8.16
Chemical	25	0.5	0.4994	0.10	18	0.40	1.2×10^{-7}	99.99	2000030	1.55
	27	0.5	0.4937	1.25	20	0.45	7.8×10^{-9}	99.99	2000034	1.96

Table 1 displays the outcomes of these experiments. In all the considered cases, Algorithm 1 returned, in reasonable time, quite accurate results in terms of relative error. For comparison, results obtained with Prism, a state-of-the-art probabilistic model checker [6], are also included. Results provided by Prism were reasonably accurate only in three out of eight cases. In one case (*Nasty* with 10^6 states), Prism was not able to build the model after about one hour. In five out of eight cases, the Matlab implementation of Algorithm 1 run anyway faster than Prism.

7 Conclusion, Further and Related Work

We have demonstrated that, in the analysis of Markov chains, generating functions provide a bridge toward Padé approximation theory that is useful both at a conceptual and at a technical level. Direct extensions of the method to full temporal logics, such as LTL, seem worth studying, as well as extensions to richer models, like continuous Markov chains or Markov Decision Processes. Another potential field of application is the time-bounded analysis of infinite-state system.

Methods and tools based on a symbolic representations of the entire state-space, such as Prism [6], can take great advantage of the presence of system regularities, as e.g. induced by massive interleaving: in those cases, an on-the-fly approach cannot be expected to match the performance of these tools. Nevertheless, as indicated by our small-scale experiment, the presented methodology turns out to be helpful in situations of bad eigenvalues separation, and/or when, for whatever reason, building the entire system's model turns out to be not feasible. In numerical linear algebra, numerous worksare devoted to the experimental evaluation of projective methods applied to Markov chains, see e.g. [7] and references therein. These works focus on the calculation of steady-state probabilities and no connection to generating functions is made. Further discussion on related work, including use of Padé approximation in Engineering [1], can be found in [4].

References

1. Antoulas, A.C.: Approximation of Large-scale Dynamical Systems. SIAM (2005)
2. Arnoldi, W.E.: The principle of minimized iterations in the solution of the matrix eigenvalue problem. Quarterly of Applied Mathematics **9**, 17–29 (1951)
3. Baker Jr., G.: Essentials of Padé Approximants. Academic Press (1975)
4. Boreale, M.: Full version of the present paper, Matlab and Prism code. http://rap.dsi.unifi.it/~boreale/papers/GFviaKrylov.rar
5. Hartfiel, D.J., Meyer, C.D.: On the structure of stochastic matrices with a subdominant eigenvalue near 1. Linear Algebra Appl. **272**, 193–203 (1998)
6. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011)
7. Philippe, B., Saad, Y., Stewart, W.J.: Numerical Methods in Markov Chain Modelling. Operations Research **40**, 1156–1179 (1996)
8. Saad, Y.: Iterative methods for sparse linear systems. SIAM (2003)
9. Wilf, H.S.: Generatingfunctionology, 2/e. Academic Press (1994)