# A Nearly-Sublinear Method for Approximating a Column of the Matrix Exponential for Matrices from Large, Sparse Networks

Kyle Kloster[1,⋆] and David F. Gleich[2,⋆]

[1] Purdue University, Mathematics Department
[2] Purdue University, Computer Science Department
{kkloste,dgleich}@purdue.edu
http://www.cs.purdue.edu/homes/dgleich/codes/nexpokit

**Abstract.** We consider random-walk transition matrices from large social and information networks. For these matrices, we describe and evaluate a fast method to estimate one column of the matrix exponential. Our method runs in sublinear time on networks where the maximum degree grows doubly logarithmic with respect to the number of nodes. For collaboration networks with over 5 million edges, we find it runs in less than a second on a standard desktop machine.

**Keywords:** Matrix exponential, Gauss-Southwell, local algorithms.

## 1 Introduction

The matrix exponential is a standard tool in network analysis. Its uses include node centrality [9,11,10], link-prediction [15], graph kernels [14], and clustering [8]. For the particular problems of node centrality, graph kernels, and clustering, what is most valuable is a coarse estimate of a column of the matrix exponential. In this paper, we consider computing $\exp\{\mathbf{P}\}\mathbf{e}_c$ where $\mathbf{P}$ is the random-walk transition matrix for a directed or undirected graph and $\mathbf{e}_c$ is the $c$th column of the identity matrix. More precisely, and to establish some notation for the paper, let $\mathbf{G}$ be an $n \times n$ adjacency matrix for a *directed or undirected* graph, let $\mathbf{e} = [1, \cdots, 1]^T$ the vector (of appropriate dimensions) of all 1s, and let $\mathbf{D} = \mathrm{diag}(\mathbf{Ge})$ so that $\mathbf{D}_{ii} = d_i$ is the degree of node $i$ (and $d = \max_i\{d_i\}$ is the maximum degree). We consider $\mathbf{P} = \mathbf{GD}^{-1}$. This case suffices for many of the problems studied in the literature and allows us to compute exponentials of the negative normalized Laplacian $-\hat{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{GD}^{-1/2} - \mathbf{I}$ as well. Observe that

$$\begin{aligned}\exp\{\mathbf{D}^{-1/2}\mathbf{GD}^{-1/2} - \mathbf{I}\}\mathbf{e}_c &= e^{-1}\mathbf{D}^{-1/2}\exp\{\mathbf{GD}^{-1}\}\mathbf{D}^{1/2}\mathbf{e}_c \\ &= \sqrt{d_c}e^{-1}\mathbf{D}^{-1/2}\exp\{\mathbf{GD}^{-1}\}\mathbf{e}_c,\end{aligned}$$

so computing a column of either $-\hat{\mathbf{L}}$ or $\mathbf{P}$ allows computation of the other at the cost of scaling the solution vector.

⋆ Supported by NSF CAREER award 1149756-CCF.

A. Bonato, M. Mitzenmacher, and P. Prałat (Eds.): WAW 2013, LNCS 8305, pp. 68–79, 2013.
© Springer International Publishing Switzerland 2013

Computing accurate matrix exponentials has a lengthy and "dubious" history [17]. Let $\mathbf{A}$ be a general $n \times n$ matrix that is large and sparse and let $\mathbf{b}$ be a general vector. A popular method for computing $\exp\{\mathbf{A}\}\mathbf{b}$ involves using an $m$-step Krylov approximation of the matrix $\mathbf{A}$ yielding $\mathbf{A} \approx \mathbf{V}_m \mathbf{H}_m \mathbf{V}_m^T$. If we use this form, we can approximate $\exp\{\mathbf{A}\}\mathbf{b} \approx \mathbf{V}_m \exp\{\mathbf{H}\}\mathbf{e}_1$. For $m \ll n$, the computation is reduced to the much smaller $\exp\{\mathbf{H}\}$ at the cost of using $m$ matrix-vector products to generate $\mathbf{V}_m$. Such an approximation works well for computing both the entire exponential $\exp\{\mathbf{A}\}$ and its action on a vector: $\exp\{\mathbf{A}\}\mathbf{b}$. This idea underlies the implementation of the Matlab package ExpoKit [21], which has been a standard for computing $\exp\{\mathbf{A}\}\mathbf{b}$ for some time.

While these algorithms are fast and accurate (see references [13], [12], and [2], for the numerical analysis), they depend on matrix-vector products with the matrix $\mathbf{P}$ and orthogonalization steps between successive vectors. When a Krylov method approximates a sparse matrix arising from a graph with a small diameter, then the vectors involved in the matrix-vector products become dense after two or three steps, even if the vector starting the Krylov approximation has only a single non-zero entry. Subsequent matrix-vector products take $O(|E|)$ work where $|E|$ is the number of edges in the graph. For networks with billions of edges, we want an alternative to Krylov-based methods that prioritizes speed and sparsity over accuracy. In particular, we would like an algorithm to estimate a column of the matrix exponential in less work than a single matrix-vector product.

Local methods perform a computation by accessing a small region of a matrix or graph. These are a practical alternative to Krylov methods for solving massive linear systems from network problems that have sparse right hand sides; see, for instance, references [3,7]. Rather than matrix-vector products, these procedures use steps that access only a single row or column of the matrix. We design a local algorithm for computing $\exp\{\mathbf{P}\}\mathbf{e}_c$ by translating the problem of computing the exponential into solving a linear system, and then using a local algorithm.

We present an algorithm that approximates a specified column of $\exp\{\mathbf{P}\}$ for column stochastic $\mathbf{P}$ (Section 4, Figure 1). The algorithm uses the Gauss-Southwell method (Section 2) for solving a linear system to approximate a degree $N$ Taylor polynomial (Section 3). The error after $l$ iterations of the algorithm is bounded by $\frac{1}{N!N} + l^{-1/(2d)}$ as shown in Theorem 2, and the runtime is $O(ld + ld\log(ld))$ (Section 5.3). Given an input error $\varepsilon$, the runtime to produce a solution vector with error less than $\varepsilon$ is sublinear in $n$ for graphs with $d \leq O(\log\log n)$. We acknowledge that this doubly logarithmic scaling of the maximum degree is unrealistic for social and information networks where the maximum degree typically scales almost linearly with $n$. Nevertheless, the existence of a bound suggests that it may be possible to improve or establish a matching lower-bound.

## 2 Local Computations and the Gauss-Southwell Method

The Gauss-Southwell (GS) iteration is a classic stationary method for solving a linear system related to the Gauss-Seidel and coordinate descent methods [16]. It is especially efficient when the desired solution vector is sparse or localized [7,5]